

Graduation Portfolio

Saxion Hogeschool, Deventer
Bsc in Software Engineering(HBO IT) 2019-2023



Student:
Company:

Keith I. (487130)
Voordeeluitjes.nl (Freetime Company)

Table of Contents

1. Project Proposal.....	5
2. Plan of Approach.....	5
2.1. Introduction.....	5
2.1.1. Goal.....	5
2.1.2. Target Group.....	5
2.1.3. Workspace.....	5
2.2. Project Description.....	5
2.3. Company Description.....	6
2.4. Project Details.....	6
2.4.1. Expected Research.....	7
2.4.2. Expected Solution.....	7
2.4.3. Expected Tasks.....	8
2.5. Functional Requirements.....	8
2.6. Non-Functional Requirement.....	9
2.7. User Stories.....	9
2.8. System Architecture Diagram.....	10
2.9. Project Management.....	11
2.9.1. Planning.....	12
2.9.2. FAQs for incoming team members.....	13
3. Research Report.....	15
3.1. Abstract.....	15
3.2. Introduction.....	15
3.3. Literature review.....	16
3.4. Data collection and Analysis.....	17
3.4.1. Data Preprocessing.....	19
3.5. Machine Learning model.....	24
3.5.1. Scikit-learn.....	24
3.5.2. TensorFlow and Keras.....	25
3.5.3. Spark MLlib.....	25
3.5.4. LightFM.....	25
3.5.5. Surprise.....	26
3.5.6. Fastai.....	26
3.6. Software development(Backend).....	27
3.6.1. Flask.....	29
3.6.2. Django.....	29
3.6.3. FastAPI.....	30
3.6.4. Node.js/Express.....	30
3.6.5. Spring Boot.....	30
3.6.6. Go.....	30
3.7. Database.....	31
3.7.1. SQLite.....	31

3.7.2. MySQL.....	32
3.7.3. PostgreSQL.....	32
3.8. Software development(Frontend).....	33
3.8.1. ReactJs.....	33
3.8.2. Vue.js.....	34
3.8.3. Angular.....	34
3.9. Recommendations.....	34
3.10. Conclusion.....	35
4. Functional Design.....	36
4.1. Introduction.....	36
4.2. Overview.....	36
4.3. Requirements.....	36
4.3.1. Functional requirements.....	36
4.3.2. Non-Functional Requirement.....	37
4.3.3. User Stories.....	37
4.3.4. Use Cases.....	38
4.3.5. User Interface.....	39
4.4. Data.....	39
4.5. Assumptions.....	39
4.6. Acceptance Criteria.....	40
5. Technical Design.....	41
5.1. Introduction.....	41
5.2. System Architecture.....	41
5.3. Database Design.....	41
5.4. Backend Design.....	43
5.4.1. Libraries and systems for backend development.....	44
5.5. Recommendation Model.....	45
5.5.1. Model Logic.....	45
5.5.2. Location(Hotel) Recommendation.....	45
5.5.3. Model management.....	46
5.5.4. Model Testing.....	46
5.6. Frontend Design.....	48
5.6.1. Libraries and systems for frontend development.....	50
5.7. Security.....	51
5.8. Performance.....	52
5.9. API Documentation.....	53
5.10. Testing.....	56
5.11. Deployment.....	57
5.12. Logging.....	58
5.13. Scalability.....	58
5.14. Recovery.....	59
5.15. Support.....	59
5.16. Conclusion.....	59

6. Systems Manual.....	60
6.1. System setup and Installation.....	60
6.2. API Documentation.....	60
6.3. SQL Queries.....	60
6.4. Known Issues.....	61
6.5. Unfinished parts.....	62
6.6. Troubleshooting.....	62
7. Graduation Report.....	64
7.1. Introduction.....	64
7.2. Problem definition.....	64
7.3. Assignment.....	64
7.4. Design decisions.....	64
7.5. Development environment.....	64
7.6. Results.....	65
7.7. Recommendation.....	66
7.8. Conclusion.....	67
7.9. Reflection.....	67
8. Video Demo Presentation.....	68
8.1. Draft.....	68
9. Appendix.....	69
9.1. Graduation proposal HBO-ICT.....	69
9.2. Research Questions.....	74
9.2.1. RQ1: (Recommendations).....	74
9.2.2. RQ2: (Data).....	75
9.2.3. RQ3: (Implementation).....	75
9.2.4. RQ4: (Interface).....	75
9.3. Contacts.....	75
9.4. Glossary.....	75
9.5. Readme.md.....	76
9.5.1. Main.....	76
9.5.2. Backend setup.....	77
9.5.3. Frontend setup.....	79
9.6. SQL Queries.....	81
9.7. CSV files format.....	85
9.8. API documentation preview.....	89
9.9. Concept design.....	90
9.10. Gitlab commit log.....	92
9.11. Graphs & Diagrams.....	92
9.12. Screenshots.....	93
9.13. References.....	97
9.14. Substantive Reflection.....	98
9.15. Acknowledgements.....	98

1. Project Proposal

The project proposal outlines the goals of the assignment. This was to be approved by the college and the company. See [project proposal](#).

2. Plan of Approach

2.1. Introduction

Chapter 1 gives general background about the project. Chapter 2 discusses points needed in case a new programmer joins the teams.

2.1.1. Goal

This document lays out goals and objects of the project.

2.1.2. Target Group

This document gives information for anyone who is interested to learn about the project and fellow team members.

2.1.3. Workspace

The project is to be carried in a hybrid form working remotely and on site at VDU address in Deventer, NL.

2.2. Project Description

For this project the company wants to show personalized search results for holiday destinations on their website. Currently, the company's system makes recommendations of search results based on deterministic queries on SQL. These search results offer matching results based on keywords. However, these results may not reflect the user's personal interests or preferences, and may miss out on similar or alternative destinations or hotels that the user may find more appealing.

VDU records all users' interaction with their website. This data contains history of user's purchases and liked destinations. They want to use this data to potentially recommend other destinations to their customers which could increase conversion(sales) rates.

The aim of this project is to design and implement a machine learning-based recommendation system that can generate personalized search results for users based on their past booking history, current website interaction, and other factors. The project will follow the software development lifecycle (SDLC) phases of planning, analysis, design, implementation, testing, and deployment. The project will also involve data processing, and analysis to build and evaluate machine learning models which leads to recommendations of high accuracy.

The main research question of this project is:

- RQ1: What ways can hotel search results be made more relevant to users?

The sub-questions are:

- RQ2: What data must be collected from visitors of the website, which could be used to provide personalized recommendations based on the user's personal preference?

- RQ3: How can this recommendation system be implemented into an existing website?
- RQ4: How can administrators see progress reports and do customization to the ML model?

2.3. Company Description

The company Voordeeluitjes.nl(VDU) is a leading online travel website in the Netherlands which offers a wide range of holiday destinations and hotels in the Netherlands and Germany. Their mission is to provide the best travel experience for its customers by offering them package deals which gives their clients convenience with possible discounts.

Voordeeluitjes.nl(VDU) is an online travel agency that specializes in offering affordable and attractive holiday packages in the Netherlands and Germany. The company was founded in 2007 and has become one of the leading websites among others such as booking.com. They have a network of over 6000 hotels and holiday parks, which it promotes. The company's vision is to make travel accessible and enjoyable for everyone by offering the best deals.

Their website is their main channel for reaching and serving its customers. The website allows users to search for holiday destinations and hotels based on various criteria, such as location, price, rating, popularity, theme, or facilities. The website also provides detailed information about each destination and hotel, such as photos, reviews, maps, activities, and booking availability information. The website lets users book their holiday packages online. Their website is powered by a SQL-based database system that stores and retrieves the data of the destinations and hotels. They also provide easy and fast customer service support.

2.4. Project Details

VDU's current search results system for its service is based on deterministic queries on SQL. This means that the system matches the user's search keywords with the data in the database and returns the results which match the search key phrase. This technique is quite outdated as it does not take into account the user's personal preferences or interests. These personal preferences vary for each user depending on their past booking history, interaction with the website, ratings etc. Therefore, the search results may not be relevant or appealing to the user if they are solely matched on keywords. There is missed opportunity to show alternative destinations or hotels that the user may find more attractive.

For example, suppose a user searches for "beach holiday" on the company's website. The current system will return a list of hotels that are located near a beach and have the word "beach" in their name or description. However, the user may have other preferences or interests that are not captured by the keyword "beach holiday" as the results do not take into account the user's history and booking behavior. This booking behavior may be that users who generally like to book on holiday near a beach also like to places that are sunny for example. This would suggest that users like to go on holiday in "tropical" places. If this is the case then there would be an invention to show such holiday places to this user. Otherwise, the user may not be satisfied with the search results and may not book any of the hotels.

The company wants to improve its search results system by showing holiday destinations/hotels based on users' interaction history to provide more relevant holiday destination results to make their offers more attractive. The company believes that by using machine learning techniques, it can discover the relationship

between a user's booking habits to generate personalized search results for each user based on their past booking history, current website interaction, and other factors.

2.4.1. Expected Research

The expected research for this project consists of the following activities:

- Literature reading: A review of the existing literature and research on machine learning-based recommendation systems as well as software development techniques for best practices.
- Data collection and analysis: A collection and analysis of the data that will be used to build and evaluate the machine learning model. The data will include the user interaction data that the company already collects, such as user clicks, purchase history, liked destinations, liked hotels, favorite pages, etc., as well as any additional data that may be useful for enhancing the recommendation system, such as user demographics, preferences, feedback, etc. The data collection and analysis will involve methods such as data extraction, transformation, cleaning, integration, exploration, visualization, and preprocessing.
- Machine learning model design and implementation: A design and implementation of the machine learning model that will generate personalized and relevant search results for users based on their past booking history, current website interaction, and other factors. The machine learning model design and implementation will involve methods such as feature engineering, model selection, model training, model testing, model evaluation, model optimization, and model deployment.
- Model integration and testing: An integration and testing of the machine learning model into the existing website system. The model integration and testing will involve methods such as system architecture design, system configuration, system testing, system debugging, system validation, and system documentation.
- Model administration and customization: An admin interface(website) that will allow administrators to see reports and do customization to the ML model. The admin website will involve methods such as web development, web design, web security, web testing, web debugging, web validation, and web documentation.

2.4.2. Expected Solution

In this section I layout what products I will be delivering to complete this assignment.

- A research report that documents the data analysis process, the machine learning model development process, the model evaluation and comparison results, and the recommendations for future work.
- A model implemented outside the VDU website that returns a personalized list of recommended hotel destinations. These results can be then displayed in the existing website search functionality and provides users with a new option to sort results by personalized recommendation.
- An internal admin website that allows the administrators to monitor and manage the machine learning model, view various reports and set custom properties or overrides for specific hotels or keywords.
- Documentations for the system.

2.4.3. Expected Tasks

- Data analysis: Review the existing data and identify the relevant features for building a personalized recommendation system. Perform data cleaning, preprocessing and transformation to prepare the data for machine learning. Create graphs to analyze data.
- Machine learning model development: Choose a machine learning algorithm for the recommendation system, such as collaborative filtering, content-based filtering or hybrid methods. Train, test and evaluate the model using suitable metrics and techniques. Optimize the model parameters and hyperparameters using grid search, random search or other methods. Compare different models and select the one with highest accuracy which can be trained in the shortest time.
- Model deployment: Integrate the machine learning model with the existing website system using API web services. Test that model can handle website requests and provide personalized recommendations in real time. Test the model functionality and performance on the website using various scenarios and user profiles. The model needs to have an update mechanism which will periodically update new user data to update its model with latest user information.
- Admin interface development: Design and implement an internal web interface for the administrators so that they can monitor and manage the machine learning model. This admin website will provide features such as viewing model output, user segments, user profiles and conversion rates; setting custom properties or overrides for specific hotels or keywords; activating or deactivating the model or its components; updating or retraining the model with new data and possibly other features which may be useful for admins.
- Documentation: Create documentation for the machine learning model and admin interface. The documentation will include motivation, process, test results, conclusions and recommendations. The documentation will also include some diagrams, screenshots.

2.5. Functional Requirements

These are the requirements that specify what the system should do for the users and the administrators.

Ref	Description
F1	The system should generate personalized recommendations based on the user's interests and preferences, which are derived from their past booking history, current website interaction and other relevant data sources.
F2	The personalized recommendation results should be fetched by VDU's existing booking website.
F3	The system should be able to update its recommendation ML model based on the user's new data.
F4	The system should allow administrators to view various reports on the model's performance.
F5	The system should allow administrators to set custom properties or overrides for specific hotels or keywords, such as boosting or suppressing them in the personalized recommendations. The system should allow administrators to activate or deactivate the model or its components.

2.6. Non-Functional Requirement

These are requirements that specify how the system should perform or behave in terms of quality, reliability, security, usability, etc

Ref	Description
NF1	The system should be able to handle a large number of requests without compromising performance or accuracy.
NF2	The system should be secure and protect the data and the model from unauthorized access or manipulation. Data is not added to the repo as all CSV files are on gitignore list.
NF3	The system should be user-friendly and intuitive for both users and administrators, providing clear instructions, feedback and error messages.
NF4	The system should work with the existing website system.
NF5	The user's data that will be processed by ML should be anonymized by removing any user's names and contact details.

2.7. User Stories

As a client, I want to...

US1	see personalized recommendations for hotels and destinations based on my interests and preferences, so that I can find the best options for my holiday.
US2	..have a new option to sort results by personalized recommendation along with other existing sorting functions (price, rating, popularity), so that I can easily compare and choose the most relevant results for me.
US3	..see images, ratings, prices and other details of the hotels or destinations in the personalized recommendations, so that I can get a clear and appealing overview of the options.
US4	..provide feedback on the personalized recommendations, such as liking, disliking or rating them, so that I can improve the quality and accuracy of the recommendations for me and other users.
US5	..monitor and manage the machine learning model that generates the personalized recommendations, so that I can ensure its optimal performance and functionality.

As an administrator, I want to...

US6	..view various reports and insights on the model output, user segments, user profiles and conversion rates, so that I can evaluate the effectiveness and impact of the personalized recommendation system.
US7	..set custom properties or overrides for specific hotels or keywords, such as boosting or suppressing them in the personalized recommendations, so that I can adjust the results according to business needs or preferences.
US8	..activate or deactivate the model or its components, such as using only collaborative filtering or content-based filtering or hybrid methods, so that I can experiment with different approaches and methods for the personalized recommendation system.
US9	..update or retrain the model with new data, so that I can keep the personalized recommendation system up-to-date and relevant.

2.8. System Architecture Diagram

Flow chart

This flowchart shows how the SSR system will fit into the VDU website overall. It shows that SSR is an independent system which acts as a feature adding component for the VDU website.

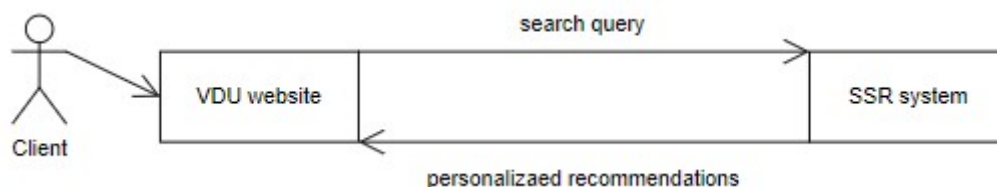


Fig PFG. Process flow graph

SSR system diagram

This diagram shows all components of the SSR system and their connection to each part therein.

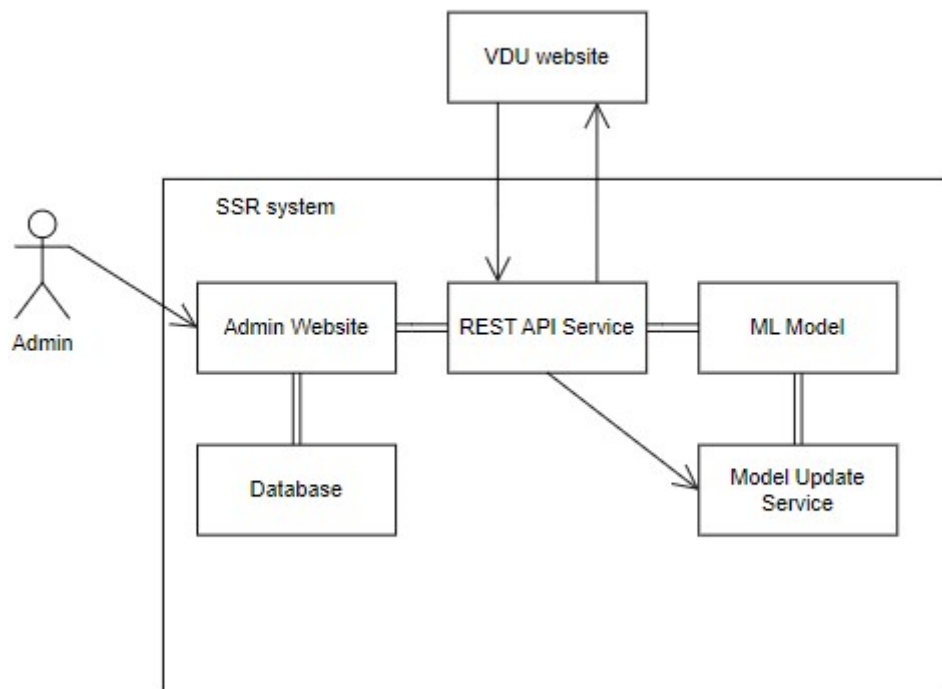


Fig SYSD. SSR system diagram

Link: <https://app.diagrams.net/?src=about#Amywebartist%2Fvdu%2Fmain%2Fdiagrams.drawio>

2.9. Project Management

The project started in Q4(May 2023) and should be completed by Q1(Oct 2023). The delivered products should address the main and sub research questions. The activities and tasks should be limited to the scope of the assignment. The project assumes there is sufficient user data available upon which the ML model would be created. Measures should be taken to secure data as per privacy act regulation. Risks such as spending too much time on just one deliverable should be avoided as it can lead to delays.

The software development includes various parts which are crucial for the success of this new addition to the system. These parts need to be carefully designed and created within the allocated time. The work done should be presented to company mentors and on a regular basis to feedback in order to assess whether the project is going as per company specifications. The project requires full time devotion from myself for which I need to read various articles, books and tutorials in order to create expected products.

In closure the system will be presented to the client to use and evaluate. This will be done by presenting a research report, the machine learning model, admin website and systems documentation. After which I will write a reflection report of the process. The project work will be showcased to college lecturers for

feedback, assess the strengths and weaknesses of the project and identify any lessons learned or recommendations for future likewise projects.

2.9.1. Planning

The project consists of four main phases: data analysis, machine learning model development, admin interface development and testing.

In the data analysis phase, I will explore and visualize the data to gain insights on user's booking behaviors.

In the machine learning model development phase, I will choose a suitable machine learning algorithm for the recommendation system and train, test and evaluate the model using the data.

In the software deployment phase, I will create an admin website which will administer the model using a RESTful API and JSON format. I will test the model functionality and performance. The model API service and admin website will be both created using Python and Flask.

I will document the work throughout the project. I will also write a brief description for the model created and more detailed documentation for the admin interface.

To plan the time effectively, I have made a table below that shows the start and end dates of each task and milestone, as well as their dependencies.

Below table shows the start and end dates in 2023 of each milestone along its dependency

Ref	Task	Start	Dependency
T1	Review existing data sources	07 May	None
T2	Identify relevant features	14 May	T1
T3	Perform data cleaning, preprocessing and transformation	21 May	T2
T4	Explore data using graph visualization	28 May	T3
T5	Choose machine learning algorithm	7 Jun	T4
T6	Train, test and evaluate models	28 Jun	T5
T7	Optimize model parameters and hyperparameters	7 Jul	T6
T8	Compare different models and select best one	14 Jul	T7
T9	Create REST API for fetching recommendations from model	7 Sep	T8
T10	Test model functionality and performance on website	14 Sep	T9
T11	Design website for administrators	21 Sep	None

T12	Implement website features for admins	28 Sep	T11
T13	Test web interface functionality and security	7 Oct	T12
T14	Write research report	14 Oct	None
T15	Write documentations	21 Oct	None
T16	Presentation	14 Nov	T16

2.9.2. FAQs for incoming team members

In this section there are some questions and answers in case another developer joins the project.

- Which software is used?
 - The company uses SQL for database and C# for programming. However this SSR system will be built using Python, SQLite and Flask. The reason for this is that most machine learning packages are only available in Python and Flask is a simple system on which website administration websites can be built. SQLite is a good choice because they don't want a large full setup and this small db will store input output values to return search recommendations based on incoming requests.
- What will be my tasks?
 - Existing and joining students project member will go through this document to familiarize him/her self and pickup what undone requirements
- What are we building?
 - We need to build a machine learning administration website and machine learning model that uses an API system.
- Can I find how my software integrates with other software?
- What's the first thing that needs to be done?
- Who do I turn to for help?
 - See contacts for appropriate person to contact
- Where is the repository for Items I need?
 - The repository is available over here <https://gitlab.com/mywebartist/vdu>
- Who's involved in this project?
- Do I have all the necessary contact addresses?
- What software-development method is used?
- What are the deadlines?
 - 08 May 2023 Q4 start
 - 28 May 2023 draft plan of approach
 - 11 Jun 2023 final plan of approach
 - 19 Jun 2023 monthly update meeting with teacher
 - 26 Jun 2023 Q4 end
 - 04 Sep 2023 Q1 start
 - 17 Nov 2023 Q1 end
 - 30 Sep 2023 draft graduation report
 - 15 Oct 2023 reflection report

- 15 Oct 2023 evaluation form
- 21 Oct 2023 presentation
- 29 Oct 2023 final graduation report
- 14 Nov 2023 presentation
- 17-24 Nov 2023 graduation session
- Weekly client/student meeting
- What's the expected quality of programming code?
 - The code should be simple to understand. As much as possible function size be kept less than 15 lines of code.
- What's the naming convention used for the coding?
 - Use camel case
- Who's the boss?
 - Everyone is in charge of their own destiny
- Who's the lead programmer?
 - Keith is the lead programmer of this project
- What kind of system do we use for managing/reporting progress and delays?
 - Tools used for project management are Google Calendar, Gitlabs user stories
- Where do I report the bugs?
 - Any bugs are to be logged on GitLab as an issue
- To who do I report about workload/working conditions/etc
 - Speak with client or college supervisor
- What are the working hours?
 - Working hours are 10am to 3pm, Monday to Friday
- To whom do I report when I'm late/to the doctor/ill
 - Speak with client or college supervisor

3. Research Report

3.1. Abstract

In this research report, I present my journey of developing and implementing a simple yet sophisticated recommendation system for a vacation booking website Voordeeluitjes.nl. My main goal was to create a personalized hotel recommendation system which shows visitors more attractive search results than the one currently being generated which uses conventional keyword-based search results.

My approach involved data collection and analysis, in which I looked at user's booking histories and similarities between hotels' facilities. Through machine learning techniques, I constructed a model capable of delivering tailored booking suggestions. I housed this system by way of a standard REST API system with adequate access protection.

My main research involved building a complete system which can be implemented into existing website systems. The system includes data and recommendation fetching, model results customizations, data updating methods and user administration.

Voordeeluitjes.nl is one of the well known hotel booking websites in the Netherlands. I joined their organization to help them achieve their vision of providing an affordable vacation within the country and neighboring countries.

My proposed solution was the development of a machine learning model which could generate recommendations for hotels via a REST API system along with a simple admin interface(secure website) and documentation. The model operated independently from the website, functioning as a recommendation engine, while the admin interface empowered administrators like me to oversee, fine-tune, and extract insights from the model.

My research involved various aspects such as literature review, data collection and analysis, model design and implementation, software and website development and lastly testing. I started off with the following Plan of Approach, and Functional and Technical Design. In every milestone completing user stories played a crucial role in the development process.

The system architecture diagram shows how the system which I developed connects with the Voordeeluitjes.nl system and their booking website. I adhered to a structured project management plan to accomplish tasks on set timelines.

Finally, after much dedication and hard work I was able to deliver artifacts, which includes research report, functional recommendation model, API backend, an admin website, and documentation. I believe these products are going to enrich the travel experience for Voordeeluitjes.nl's customers and that I have made a meaningful contribution to the idea of providing personalized hotel recommendations via so called SSR(Smart Search Results).

3.2. Introduction

In this research report, I document my journey of developing and implementing a sophisticated recommendation system for the vacation booking website Voordeeluitjes.nl. The main objective of this

project is to create a personalized hotel recommendation system that offers visitors more attractive search results based on existing keyword matching.

The rationale behind this project is rooted in the understanding that today's travelers seek tailored and relevant recommendations. The existing approach of keyword-based searches often falls short in capturing the nuances of individual preferences and booking histories. To address this gap, I undertook the task of constructing a recommendation system that would leverage user data and machine learning techniques to deliver personalized booking suggestions.

This project holds great promise for enhancing the user experience on Voordeeluitjes.nl platform as they are committed to providing affordable vacations. My proposed solution involves the development of a machine learning model capable of generating hotel recommendations through a REST API system. It comes with a user-friendly admin interface and comprehensive documentation.

The journey I embarked upon was a challenging one, involving extensive literature review, data collection and analysis, model design and implementation, and software design and website development. Throughout this progress I followed a structured project management plan outlined in my initial Plan of Approach.

I hope to provide insights into the methodologies used, challenges, and lessons learned during the project. I termed my system as "Smart Search Results", this project contributes to the ongoing evolution of hotel booking experience shaped by the recent advancement of machine learning and A.I.

3.3. Literature review

Recommendation systems have become a gold mine for many tech giants. It is revolutionizing how users interact with services on online platforms. These systems are influencing users' decision-making for example, showing similar products to buy or movies to watch. Netflix is a prime example where Netflix keeps history of watched movies and based on the watch history recommend other movies to watch. This area has been growing with the help of research in achieving even more personalized and relevant user experiences on different platforms.

In order to investigate my research questions [RQ3](#)(Implementation) and [RQ4](#)(Interface) I consulted Eric Evans, in "Domain-Driven Design: Tackling Complexity in the Heart of Software" (2014), emphasizing the importance of understanding the complexities of software systems. His perspective is particularly relevant in the area of recommendation systems, where complexity increases from the need to interpret user behavior using historical data. Domain-driven design principles can help in structuring the architecture of a system such as this one so that requirements are in line with needs of users. In this case the users are visitors of Voordeeluitjes.nl. This piece of resource came in particularly useful information for me in trying to answer [RQ3](#)(Implementation).

Erich Gamma et al., in "Design Patterns: Elements of Reusable Object-Oriented Software" (1994), introduced the concept of design patterns, which offer reusable solutions to common programming challenges. While their work focuses on software design, it has implications for the development of recommendation algorithms. Well thought out design patterns can enhance the modularity(separate components) and maintainability of recommendation systems, making them more adaptable to changing switching different components allowing easy upgrade throughout the lifetime of a software.

For the area of Machine Learning(ML), Aurelien Geron's "Hands-on Machine Learning with Scikit-Learn, Keras, and Tensorflow" (2022) provides valuable insights into the practical aspects of building ML models. This book is very helpful in understanding the techniques and tools necessary for constructing recommendation models capable of learning from user data. ML allows the uncovering of patterns and insights from datasets contributing to the generation of personalized recommendations.

Miguel Grinberg, in "Flask Web Development" (2018), writes about web development with Flask. It talks about implementation of RESTful API systems. This part is an essential component of the SSR systems. The integration of machine learning models through APIs is the meat and bone of this like most today's modern web applications. Flask, as a web framework, offers a sound base for developing software systems.

In order to create robust and efficient recommendation systems, keeping best practices in mind in software development is crucial for software that can be reliably used for coming years and not a few months. "Clean Code: A Handbook of Agile Software Craftsmanship" by Robert C. Martin (2010) and "Code Complete" by Steve McConnell (2004) underlines the significance of writing clean, maintainable, and efficient code. These principles are directly applicable to the development of software systems, making sure that the codebase remains manageable by keeping function sizes short as possible and adaptable as the system grows in data and usage. Detailed and complete documentation is also important so that development could be continued when primary developers are no longer working on it.

While these resources provide essential insights into the technical aspects of software systems, it is very important to explore the theoretical and industry norms aspects. "Artificial Intelligence in Recommender Systems" by Qian Zhang et al. (2020) provides a comprehensive overview of the role of artificial intelligence in enhancing recommender systems. This work highlights the integration of AI techniques, such as collaborative filtering and content-based filtering, to improve recommendation accuracy. Sci-kit Learn and pandas are amazing tools when trying to build systems as this and these cannot be overlooked.

Development and implementation of any system involves a multidisciplinary approach, taking insights from software design, machine learning, web development and database. By taking knowledge from these areas, I should be able to build at least a working prototype that offers personalized and relevant suggestions to users, thereby boosting bookings for Voordeeluitjes.nl.

3.4. Data collection and Analysis

The underlying success of this research project is about collection and analysis of data from Voordeeluitjes.nl's website. Data received by the data team was sufficient having relevant columns which provided access to booking history and user interaction records.

This part of research helped me in answering [RQ2\(Data\)](#). The data collection process involved checking out different tables in their database and trying to find out what could form the basis of user booking records, hotel details, and website interaction logs. The company started keeping records of all requests and responses as a way to collect data. User data protection is of utmost concern. Personal data is not needed to train the model so user names and contact details were removed, and so GDPR requirements were easily met.

After having received data from different tables, I started analyzing it trying to discover patterns, trends, and insights that could drive personalized hotel recommendations. I used techniques from exploratory data analysis (EDA) to gain an understanding of the dataset's structure and characteristics.

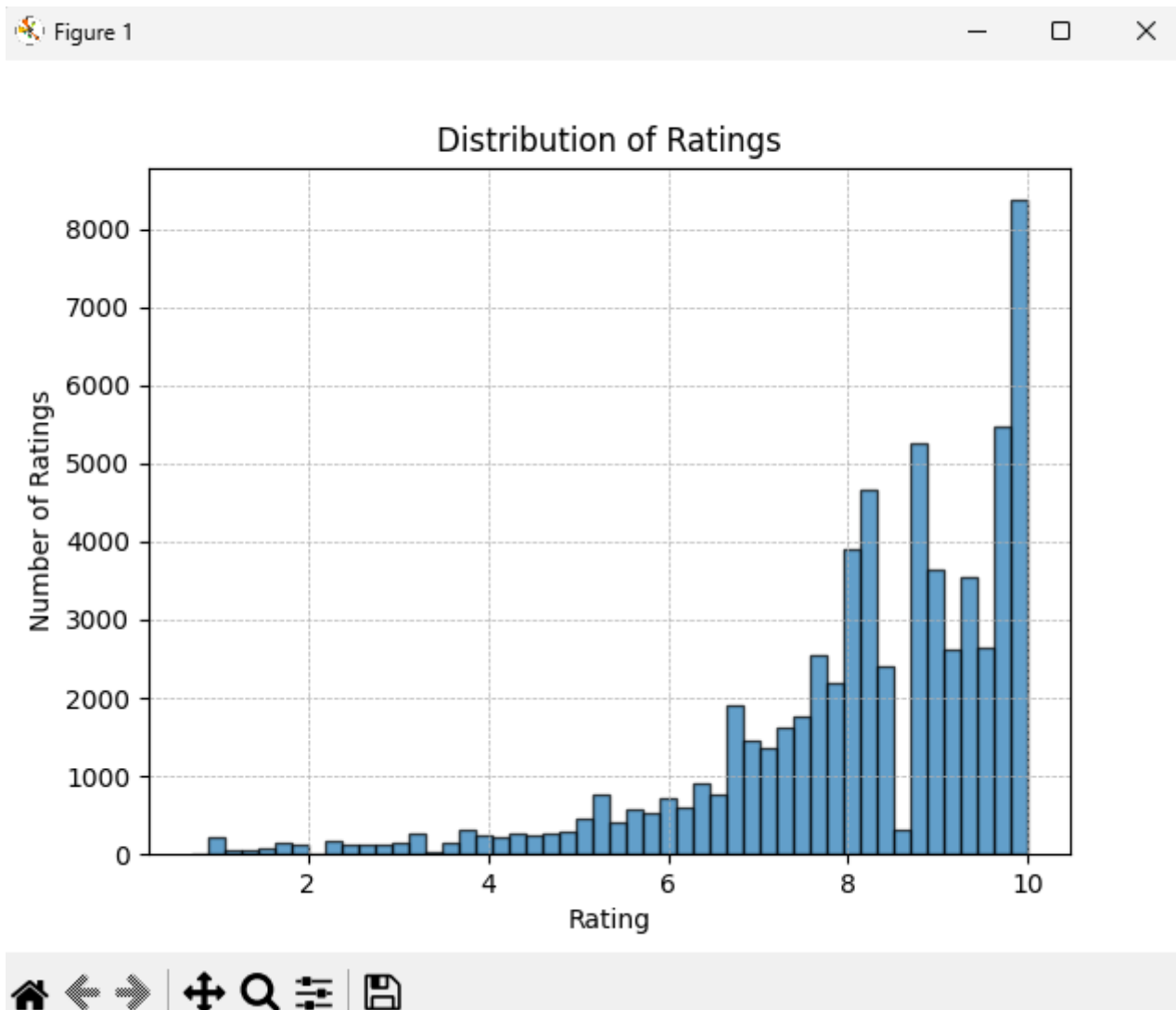


Fig. Histogram of ratings on bookings with dataset of 65,000 lines

In the end I found user booking histories to identify booking patterns. Users who had booked multiple hotels were found to show preferences forming the basis for collaborative filtering recommendation system.



Fig. Scikit-learn, TensorFlow and Apache Spark

By analysis I found attributes and features of hotels, with similarities and dissimilarities to create content-based filtering. Commonalities in facilities such as swimming pools, restaurants, and bars for example were considered when calculating the similarity between hotels.

These data-driven insights laid the groundwork for the design and implementation of the recommendation model. The data collection and analysis was very important for the project.

3.4.1. Data Preprocessing

The data provided by company and its preprocessing is as follows:

Data Lines	Lines
Bookings	455,366
Ratings	70,882
Bookings with Ratings	65,039

Data Lines

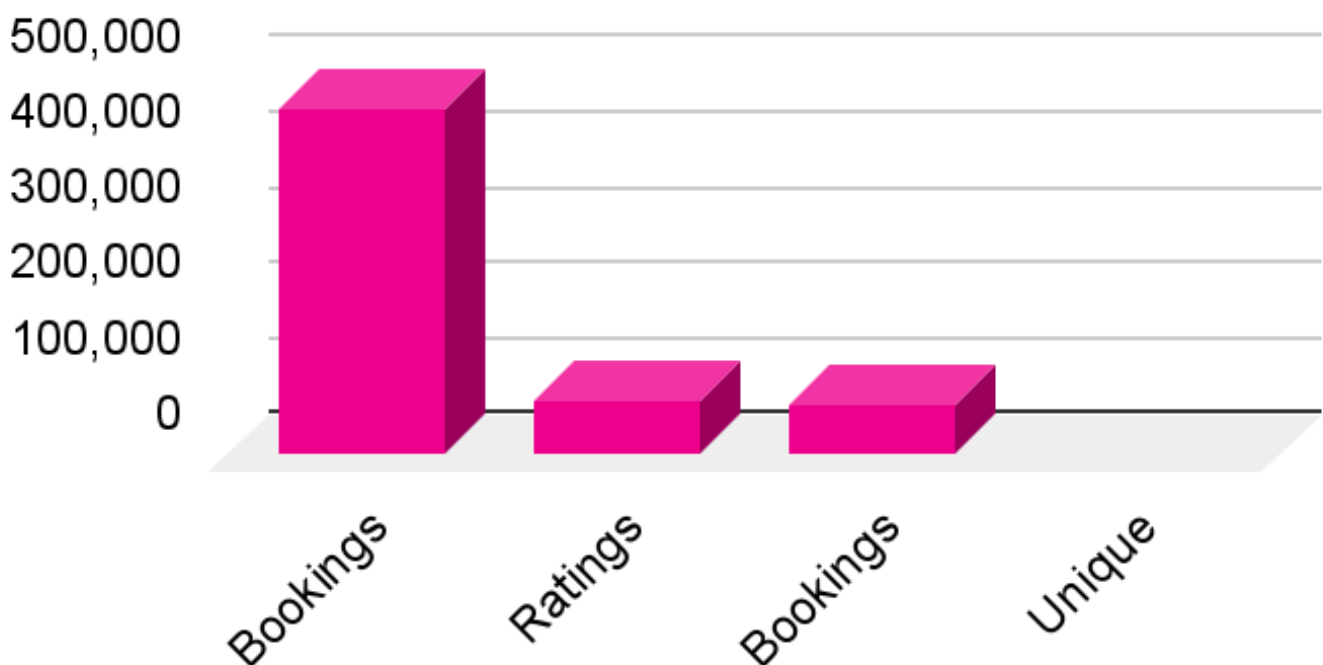


Fig. Data quantification

Countries where locations(hotels/parks) can be booked

Locations	Counts
Netherlands	2,452
Germany	1,448
Austria	610
Belgium	419
France	253
Italy	127
Luxemburg	51
Switzerland	49
Hungary	31
Croatia	30
Spain	28
Czech	20
Poland	17
Denmark	16
Portugal	14
England	12
Ireland	6
Sweden	5
Scotland	4
Slovakia	3
Greece	3
Lithuania	2
Norway	2
Morocco	2
Slovakia	2
Curacao	2
Andorra	1
Latvia	1
Montenegro	1
Total	5,611

Bookings by country

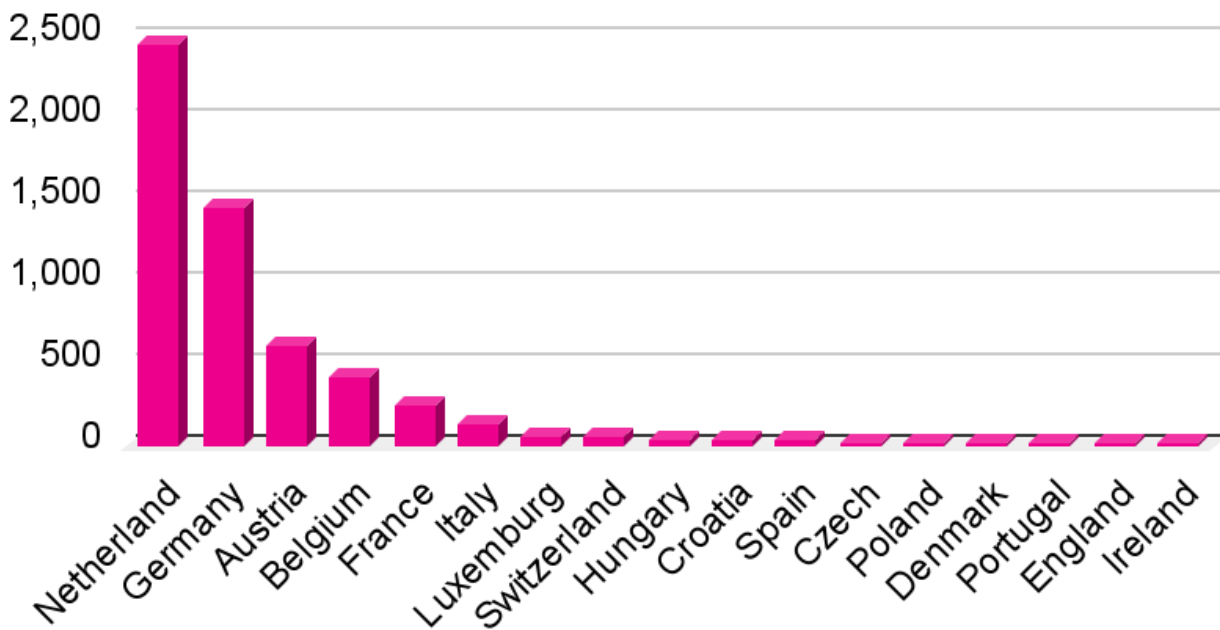


Fig.

Fig. Bookings by country

Frequency of Bookings per Person.
Total bookings 65036

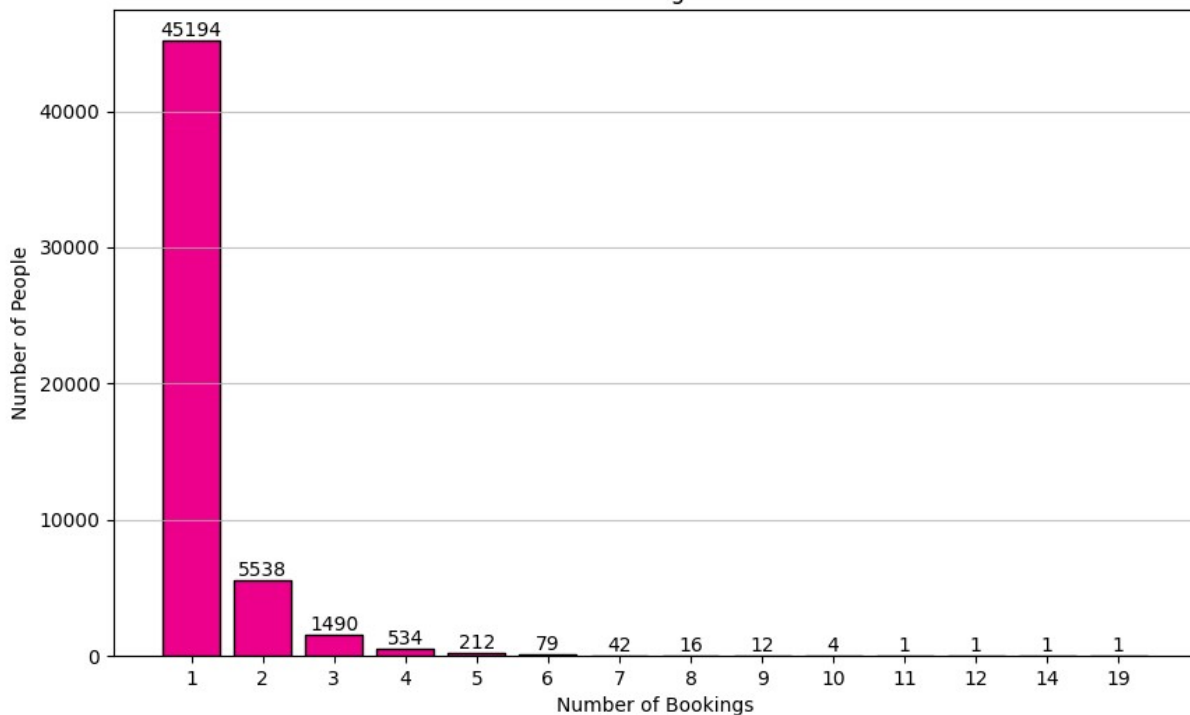


Fig. Frequency of bookings

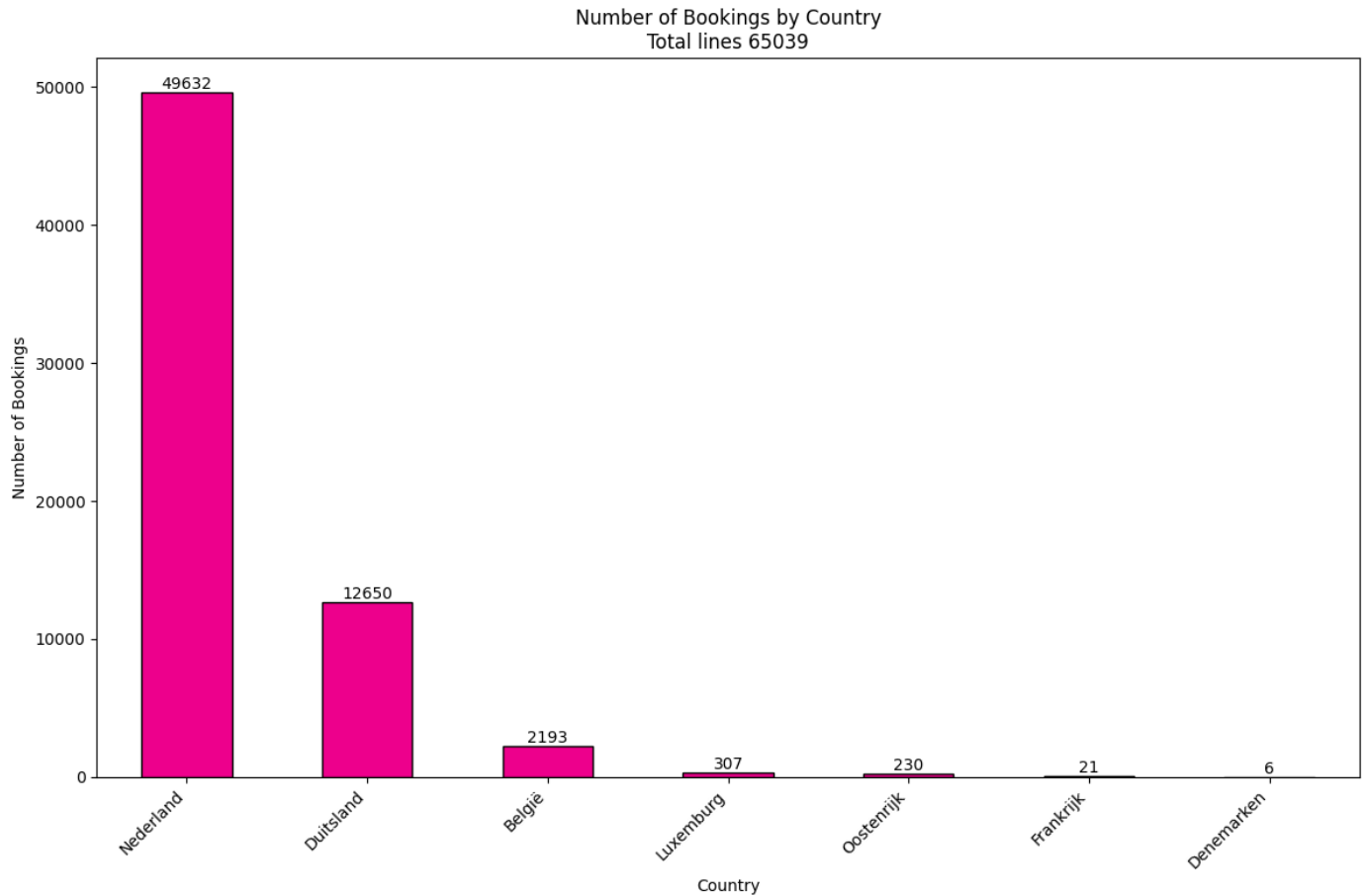


Fig. Booked countries

	Processing Time in Seconds	
Data Lines	Model 1	Model 2
100	0.01	0.01
1,000	0.81	0.04
2,000	3.94	0.14
2,500	6.84	0.23
5,000	43.48	0.95
10,000	250.35	3.83
20,000	1352.62	19.98

Fig. Processing time compare between Model 1 and Model 2

Processing Time in Seconds

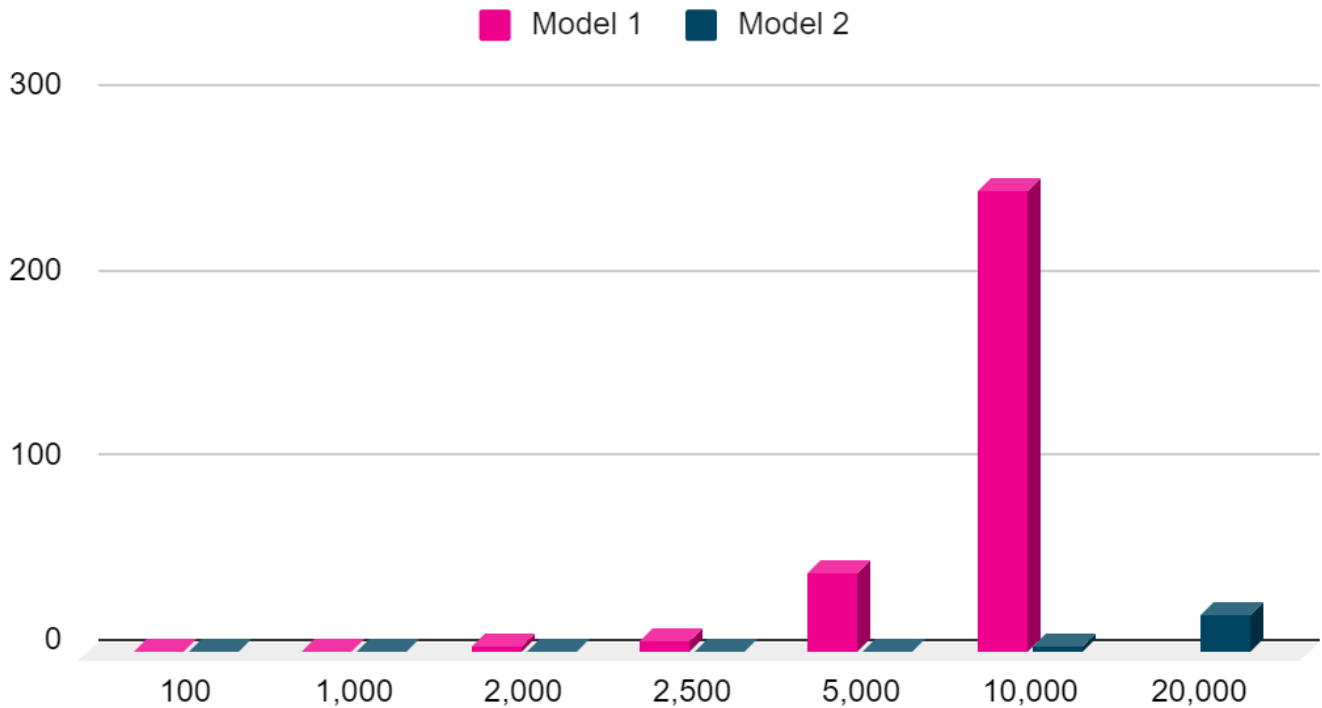


Fig. Processing time compare between Model 1 and Model 2

	Model Filesize in GB	
Data Lines	Model 1	Model 2
100	0.000129	0.000084
1,000	0.00975	0.00739
2,000	0.0359	0.0291
2,500	0.055	0.0453
5,000	0.203	0.177
10,000	0.766	0.693
20,000		2.55

Fig. Filesize compare between Model 1 and Model 2

Model Filesize in GB

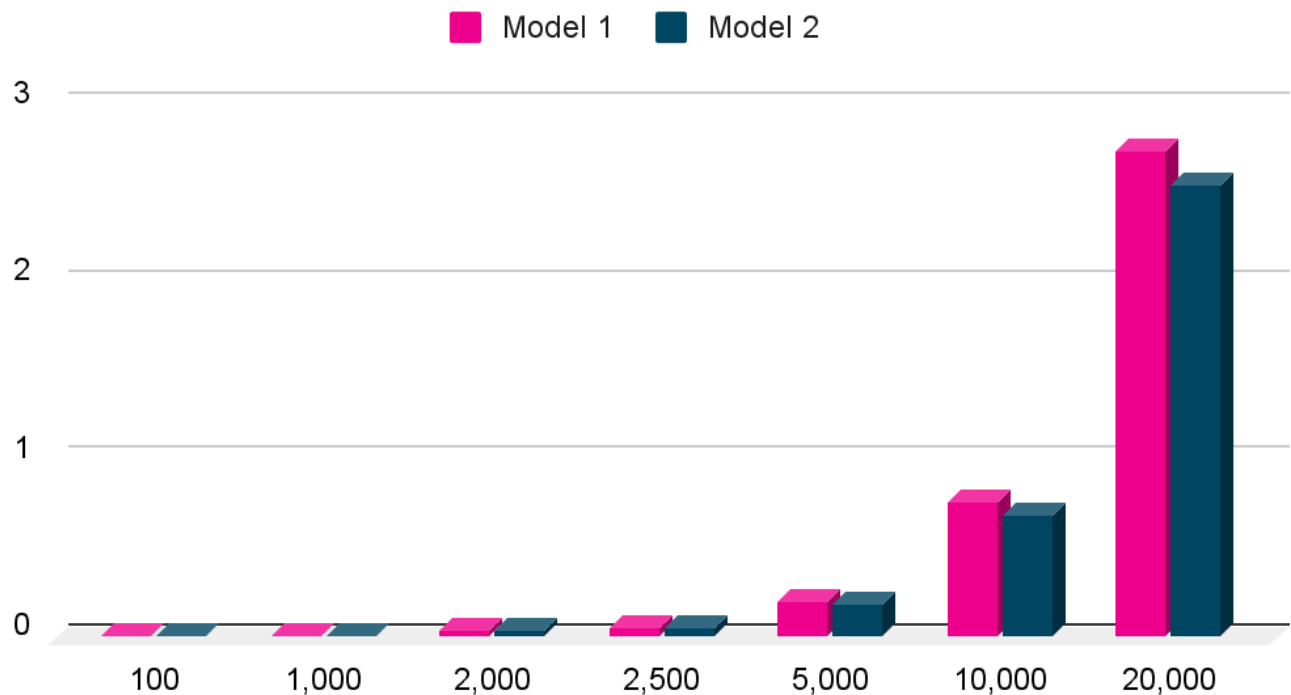


Fig. Filesize compare between Model 1 and Model 2

3.5. Machine Learning model

In order to answer [RQ1](#)(Recommendation) I referred to online sources to assess for a suitable candidate for building a machine learning model with, given available data from the company. Some of the technologies I investigated used today are as follows:

3.5.1. Scikit-learn

Scikit-learn(<https://scikit-learn.org>) is an open-source machine learning library in Python. It's renowned for its ease of use and its efficient implementations of a large number of classical machine learning algorithms. Scikit-learn is often the go-to tool for beginners and experts alike due to its simple interface and well-documented functions.

While Scikit-learn offers a vast array of machine learning algorithms, it doesn't have built-in specialized recommendation algorithms. For this hotel recommendation system such as this one, Scikit-learn can be used to develop basic models using techniques like classification or regression. However, for a more complex recommendation system that may involve collaborative filtering or matrix factorization, Scikit-learn may not be the most direct or efficient choice.

3.5.2. TensorFlow and Keras

TensorFlow(<https://www.tensorflow.org>) is a powerful and flexible open-source deep learning framework developed by Google for Python. It's designed to create and train custom machine learning models which can be used across a range of tasks.

Keras(<https://keras.io>), on the other hand, is a high-level API built on top of TensorFlow, allowing for easy and fast prototyping. Keras is an open-source library with a Python interface for creating artificial neural networks. It acts as an interface for the TensorFlow library. Additionally it supports convolutional and recurrent neural networks. Since this project is based on structured data this may not be a good choice.

When it comes to building sophisticated recommendation systems, deep learning models can capture intricate patterns in user behaviors, making TensorFlow a valuable tool. TensorFlow Recommenders, an extension of TensorFlow, offers tools explicitly designed for recommendation systems. Given a large enough dataset with hotel ratings and potentially other metadata, a deep learning-based recommendation system can yield more accurate results. However, this library is very complicated to use and develop and much harder to pinpoint issues in case of troubleshooting. There is also much higher demand for computational resources as opposed to other more basic modeling tools which work out of the box, such as Sci-kit Learn for an example.

3.5.3. Spark MLlib

Apache Spark's MLlib(<https://spark.apache.org/mllib>) available for, (Scala, Java, Python, R), is a distributed machine learning framework designed specifically for scalability and big data processing. With its robust ecosystem, Spark can handle vast amounts of data efficiently, making it a favorite for tasks that require processing large datasets in real-time or batch modes.

For a hotel recommendation system dealing with a large amount of data, Spark MLlib is a feasible option. It offers the Alternating Least Squares (ALS) algorithm, explicitly designed for recommendation systems. MLlib can handle collaborative filtering on massive(possible big data) datasets, making it capable for scenarios where scalability and performance are paramount. However, for smaller datasets or simpler projects, the overhead of setting up and maintaining a Spark environment seems to outweigh the benefits. Given the size of the dataset, half million rows which shrinks down to approximately 65,000 lines(removing bookings with no ratings) this option would be an overkill.

3.5.4. LightFM

LightFM(<https://making.lyst.com/lightfm/docs>) is a Python library dedicated to offering hybrid recommendation algorithms. As a hybrid model, it can leverage both collaborative and content-based filtering, making it adaptable to various recommendation scenarios.

LightFM shines in situations where there's a mix of user-item interactions and item metadata. For a hotel recommendation system, this translates to considering both user reviews/ratings and hotel attributes (like location, amenities, price). LightFM can effectively handle 'cold start' scenarios, where new hotels without any reviews can still be recommended based on their attributes. However, for extremely large datasets, LightFM's scalability will become a limiting factor.

3.5.5. Surprise

Surprise(<https://surpriselib.com>) is a Python library explicitly crafted for building and analyzing recommendation systems. It offers a variety of algorithms, from basic ones to more advanced recommendation strategies, and comes with built-in tools for model evaluation.

Given its dedicated nature, Surprise is aptly suited for building a hotel recommendation system, especially when starting out or dealing with medium-sized datasets. Its tools allow for quick prototyping, model evaluation, and optimization. On the downside, its focus on being a dedicated recommendation system tool means it might lack the broader flexibility and scalability features present in other, more extensive frameworks.

3.5.6. Fastai

Fastai(<https://docs.fast.ai>) is a deep learning library for Python that operates on top of PyTorch. What sets Fastai apart is its focus on providing high-level components that allow users to create complex models with minimal code. It aims to make state-of-the-art deep learning techniques accessible and easy to implement.

Fastai, with its deep learning backbone, can be used to design intricate recommendation systems for hotels, especially when non-trivial patterns in user behavior need capturing. Neural-based recommendation models can be built more effortlessly using Fastai compared to other deep learning frameworks. However, it's noted that due to neural network complexities (and computational demands) of a deep learning approach are resource heavy, especially if the dataset isn't of large size(perhaps into millions of rows of data) or if the recommendation problem isn't overly complex.

In order to ascertain the best modeling toolkit to use I have scored each library between 0-5 giving the highest point for certain criteria. The total score will give good indication as to which system is best fit for this project.

Factor	SK-L	TFlow	MLib	Light FM	Surprise	FastAI
Simple implementation	5	1	1	5	5	5
Scalable	2	5	5	1	1	5
Lightweight	3	2	2	3	3	0
Requires minimal setup	5	3	0	3	4	1
Suited for regression	5	1	5	5	5	2
Built in recommendation model	0	0	5	5	5	2
Available in Python	5	5	5	5	5	5
Total Score	25	17	23	27	28	20

Fig AML. Model building tool evaluation

As per the total scores among the chosen criteria TensorFlow clearly doesn't fit the bill. Surprise, LightFM and Sci-kitLearn have the highest rates. In my past experience I have used Scikit-learn and I was expecting it to come out on the top, so I decided to do some more investigation about the other two tools which I had not heard about before. When comparing between just these three I decided to look for their launch date, industry usage and size of their development team. Here is what I found:

Scikit-learn:

Launched in 2007, Scikit-learn is one of the most popular machine learning libraries in the Python ecosystem. It's open-source and provides simple and efficient tools for data mining and data analysis, accessible to everybody, and reusable in various contexts. Scikit-learn has had contributions from over 2,000 developers, making it one of the largest and most active communities in the machine learning landscape.

Scikit-learn supports a wide range of supervised and unsupervised learning algorithms. Extensive documentation and tutorials are available. Large community means more support, tutorials, and frequent updates. Easily integrates with other popular libraries such as NumPy, pandas, and Matplotlib.

LightFM:

Launched in 2015, LightFM is a Python implementation of a number of popular recommendation algorithms for both implicit and explicit feedback, including efficient implementation of BPR and WARP ranking losses. It's particularly suited for datasets containing both user-item interactions and item/content features. LightFM isn't as large as Scikit-learn in terms of contributors, but it has a dedicated team and a focused scope, primarily maintained by Maciej Kula.

Allows the creation of hybrid collaborative/content-based models without requiring a cold start. Efficiently handles large datasets with millions of items/users. Specifically tailored for recommendations, unlike Scikit-learn which is more general-purpose.

Surprise

Launched in 2015, Surprise is a Python kit for building and analyzing recommender systems. It offers various collaborative filtering algorithms and has tools for evaluation, analysis, and dataset handling. It's a smaller project compared to Scikit-learn but has a niche focus on recommendation systems. Simple interface for training, testing, and evaluating recommendation models. Provides several algorithms out of the box like Singular Value Decomposition (SVD), k-Nearest Neighbors (k-NN), and more. Built-in functions for cross-validation, RMSE, MAE, and other metrics pertinent to recommender systems.

Scikit-learn is by far the most widely used tool among the three given its extensive functionality with a large development base. This library is also the oldest so they have more experience in the field.. Even though both LightFM and Surprise are tailored tools for building recommendation systems I decided to go with Scikit-learn because I already have some experience with it and it is a popular standard industry choice.

3.6. Software development(Backend)

The software development aspect of this research project was needed in realizing the goal of a sophisticated and modern system for Voordeeluitjes.nl. This phase of the Smart Search Results (SSR) system involved researching data-driven model design, REST API development, user interface implementation, and documentation ideas.

The software development process includes the design and implementation of the recommendation model. Applying insights gained from data collection and analysis. Python, with its versatile libraries such as Scikit-Learn and Pandas, played an important role in model development.

The development of the SSR's REST API system. Flask, a lightweight and efficient web framework, was chosen to create the API endpoints for fetching data. The API was designed to be consumed by Voordeeluitjes.nl's existing website to make the process smooth. Access control mechanisms were integrated to safeguard sensitive user data and model endpoints. This phase also involved thorough testing to validate the API's functionality and responsiveness.

There are diverse choices available between programming languages, front-end libraries, connection protocols, and other critical elements that are commonly employed in software projects. This stage aimed to identify and evaluate numerous options for creating a recommendation system that could seamlessly integrate with Voordeeluitjes.nl's website trying to apply best practices.

One of the basic decisions during this research was the selection of programming language for developing the recommendation system. From the programming languages including Python, Java, and JavaScript. Python is renowned for its simplicity and an abundance of machine learning libraries (though not exclusively limited to ML), was considered due to its versatility in web development and efficient data processing. Java, known for its robustness, was also considered for its capability to handle large-scale systems. JavaScript, as the language of the web, was evaluated for its client-side functionalities and ability to provide dynamic user experiences. Javascript is a must for frontend but within a framework to avoid having to build everything from scratch.

The choice of frameworks and libraries played a role in shaping the software development process. Various front-end libraries like React, Angular, and Vue.js were considered for building an interactive and responsive user interface. These libraries enable the creation of dynamic web applications that can seamlessly fetch and display personalized hotel recommendations to users. Back-end frameworks such as Django, Ruby on Rails, and Express.js were explored for their capabilities in handling data requests, managing APIs, and ensuring system security. Open-source libraries like TensorFlow.js and PyTorch.js were investigated to see their feasibility for incorporating machine learning components.

Efficient communication between different system components and external data sources is critical. Protocols such as HTTP/HTTPS, WebSocket, and RESTful APIs were assessed to determine their appropriateness in facilitating data exchange. Integration with third-party services and data providers, such as hotel booking databases and location services, was considered for the recommendation system's functionality.

The research phase also involved an investigation into data storage and database systems. Both relational (e.g., MySQL, PostgreSQL) and NoSQL (e.g., MongoDB, Cassandra) databases were looked into to determine which would best accommodate the system's data requirements. The choice of database system directly impacted data speed, server requirements, scalability, and ease of maintenance.

To protect user data and secure access, various security measures and authentication methods were explored. This included considerations of OAuth 2.0 for user authentication, SSL/TLS for data encryption, and

token-based authentication for API access control. Data anonymization techniques were also examined to align with data protection regulations, such as GDPR.

Creating a visually appealing user interface was a critical aspect of the software development. Prototyping tools like Figma, Draw.io and Adobe XD were considered for designing user interfaces that would enhance the user experience. Usability testing methodologies and A/B testing frameworks were explored to validate the effectiveness of different interface designs and recommendation presentation formats.

In the research phase I explored various development methodologies, including Agile, Scrum, and Waterfall, to determine the most suitable approach for project management. Agile methodologies, known for their adaptability to changing project requirements, were selected for its iterative and collaborative development process.



Fig. Logos for Flask, Node, SpringBoot and Golang

In order to answer the [RQ3](#)(Implementation) I investigated various methods by which I could realize this part of the product requirement. These are discussed as following:

3.6.1. Flask

Flask is a lightweight and micro web framework written in Python. Its minimalist design allows developers to build web applications and APIs rapidly without the overhead of larger frameworks. Flask provides flexibility, allowing you to add only the components required.

For integrating a Python-based ML model, Flask stands out as an excellent choice. Its Pythonic nature ensures seamless integration with most ML frameworks, including those written in Python like TensorFlow, Scikit-learn, and Fastai. Flask is highly modular, so one can easily add extensions as project scales. However, being lightweight, it requires additional configurations or plugins for more complex tasks. In larger production environments, Flask's scalability could be a limitation.

3.6.2. Django

Django is a high-level, open-source web framework written in Python. It follows the "batteries-included" philosophy, providing developers with a comprehensive set of tools and features right out of the box, including an ORM, an admin interface, and more.

Django, with its robust built-in features, can be overkill for a simple API. However, for recommendation system's backend features like user authentication, database interactions, or an admin dashboard, Django is a suitable choice. The Django Rest Framework extension makes it straightforward to expose APIs. However, its heavyweight nature might add unnecessary overhead for a purely API-driven service such as this project begs.

3.6.3. FastAPI

FastAPI is a modern web framework built on top of standard Python type hints. It's designed to create APIs rapidly while ensuring high performance. FastAPI automatically generates interactive API docs, simplifying testing and debugging.

FastAPI is particularly suited for ML-based projects such as this one due to its seamless integration with Python-based ML tools and its asynchronous capabilities, offering fast performance. Speed and performance are delivered when querying the ML model. Its automatic validation, serialization, and documentation features can significantly reduce development time.

3.6.4. Node.js/Express

Node.js is a runtime environment that allows JavaScript to execute on the server side. With its event-driven architecture, Node.js is known for its scalability and performance in handling numerous simultaneous connections.

While Node.js itself is not a framework, frameworks like Express.js facilitate building APIs in Node.js. Knowing JavaScript before is a development advantage. Integrating a Python-based ML model would typically require creating a separate service for the model and communicating via HTTP requests or using a tool like edge.js to bridge Node.js with Python. This added complexity can be a drawback compared to using a Python-based web framework. It is almost certain that the recommended model is Python based so using Node Js as backend would be an unwise choice.

3.6.5. Spring Boot

Spring Boot is an extension of the Spring framework for Java, simplifying the process of building production-ready applications with minimal setup. It offers a wide range of tools for building web applications, RESTful services, and more.

For projects that require enterprise-level robustness, security, and scalability, Spring Boot is a top contender. Its vast ecosystem ensures almost any feature you need is within reach. However, integrating a Python-based ML model with a Java backend requires additional steps, such as using the Java API for Python (JEP) or running the model as a separate microservice. Again this would not be suitable when part of the system is Python based.

3.6.6. Go

Go, often referred to as Golang, is a statically typed, compiled language developed by Google. Known for its simplicity, performance, and strong support for concurrency, Go has been gaining traction for backend development.

Go offers impressive speed and performance benefits, especially for high-concurrency tasks. Its standard library facilitates building web servers and APIs without much external dependency. However, integrating a Python ML model with a Go backend would require running the model as a standalone service or using a tool like gRPC for communication. Go's static typing and different programming paradigm also requires a steep learning curve as opposed to dynamically typed languages like Python. To my understanding Golang would be least attractive for this backend system.

Just as I did in choosing the model building tool, in order to ascertain the best suited backend system to use I have scored each stack between 0-5 giving the highest point for certain criteria. The total score will give good indication as to which system is best fit for this project.

Factor	Flask	Django	FastAPI	NodeJs/ Express	Spring Boot	Go
Easy implementation	5	2	5	5	1	3
Scalable	2	3	2	5	4	3
Lightweight	5	0	5	2	1	3
Requires minimal setup	5	2	5	2	1	3
Python Based	5	5	5	0	0	0
Total Score	22	12	22	14	7	12

Fig AAPI. Backend systems choice evaluation

Here the choice is clearly narrowed down to Flask and FastAPI. FastAPI is just an addition to Python just like Flask. However, having no prior experience working with FastAPI and having used Flask previously I decided to use Flask for this project as well.

3.7. Database

There was no real need for a database system but it is essential to have a database because there is always some information that needs to be served from a database system. In this project the biggest thing that needs a database is to store user accounts information. Even though from beginning Sqlite was best choice I still documented the cons and pros between few three systems as follows:

3.7.1. SQLite

SQLite is a unique serverless database engine. Rather than adhering to the typical client-server model, SQLite processes read and write operations directly to a singular file on a disk. This design makes it the go-to choice for various platforms such as mobile apps, desktop software, and some web applications due to its simplicity and lightweight nature. Its main advantage lies in its self-contained nature, which means it doesn't require an external server or setup to function. Furthermore, the entire database is contained within a single file, making it particularly easy to transport, share, or backup.

For recommendation systems, especially those in the nascent stages or with lighter loads, SQLite can be a practical choice. Its minimalistic setup ensures that developers can integrate it quickly, and the portability aspect can be beneficial for small teams or individual developers. However, its strengths can also be its limitations in larger contexts. SQLite might not be suitable for handling massive concurrent write operations, as it can encounter locking issues. Additionally, the absence of some advanced features, which other relational databases typically provide, might constrain its utility in more intricate systems or as the system scales up.

3.7.2. MySQL

Owned by Oracle Corporation, MySQL is an open-source relational database management system that has etched its reputation on speed and reliability. Widely recognized for powering web applications, MySQL is an integral component of the popular LAMP (Linux, Apache, MySQL, PHP) stack. Its agility in processing read operations makes it a sought-after choice for a myriad of online applications and platforms.

In the context of recommendation systems, MySQL stands out with its capacity to manage extensive datasets, marking its suitability for mid to large-scale applications. The speed of its read operations can significantly enhance the efficiency of recommendation systems, which often rely on rapid data retrieval. The expansive MySQL ecosystem offers a wealth of tools, libraries, and plugins, facilitating various functionalities. However, it does come with its set of challenges. While its open-source nature is commendable, the licensing could pose concerns for certain businesses due to Oracle's proprietorship. Moreover, write operations in MySQL can be a tad slower than its counterparts like PostgreSQL, and setting it up might demand more effort than SQLite.

3.7.3. PostgreSQL

PostgreSQL proudly stands as a sophisticated open-source relational database system. It's celebrated for its versatility, allowing both SQL and JSON-based querying. Moreover, its reputation for extensibility and feature-rich offerings sets it apart from many other database systems. Whether it's table partitioning, point-in-time recovery, or the array of custom data types, PostgreSQL provides advanced solutions for intricate requirements.

In a recommendation system, PostgreSQL is a potent contender, especially when the system demands complex query handling. Such capabilities make it conducive for systems driven by intricate algorithms, like recommendation engines. Its extensibility, marked by a rich set of extensions, adds layers of functionalities that can be tailored for specific needs. Yet, it's essential to note its challenges. Novices might find its comprehensive features leading to a steeper learning curve. In certain scenarios, PostgreSQL might exhibit more overhead than MySQL. Achieving the optimum performance it's renowned for requires meticulous configuration and tuning, making the initial setup more complicated compared to simpler solutions like SQLite.

Factor	SQLite	mySQL	Postgre SQL
Easy implementation	5	3	1
Scalable	3	5	5
Lightweight	5	3	3
Requires minimal setup	5	3	2
Total Score	18	14	11

Fig ADB. Database systems choice evaluation

As per my prediction SQLite is a suitable choice for a project at this stage.

3.8. Software development(Frontend)

For the Admin interface a choice of frontend was needed. I chose three systems for their cons and pros to decide which one should be used for this project.



Fig. Logos for ReactJs, VueJs and AngularJs

Here is an evaluation of three frontend frameworks which are most popular for any website as frontend for any kind:

3.8.1. ReactJs

React is a JavaScript library developed and maintained by Facebook. It specializes in building user interfaces and facilitates the creation of dynamic web applications with data that can change over time, without reloading the page. React operates using a component-based structure, allowing developers to craft reusable UI components and manage states effectively with the Virtual DOM.

React's component-based structure is useful for building modular web interfaces where various components like user metrics, recommendation settings, and feedback panels can be integrated seamlessly. React's vast ecosystem, backed by a strong community, provides a plethora of third-party libraries and tools that can be leveraged to expedite the development of features specific to recommendation systems. Its flexibility ensures that as your recommendation system evolves, the admin interface can adapt without significant overhauls. Voordeeluitjes.nl is also built using ReactJs

so they have developers which will not have hard time continuing with the development in future. This gives ReactJs a huge plus point.

3.8.2. Vue.js

Vue.js is a progressive JavaScript framework renowned for its simplicity and adaptability. Whether it's crafting user interfaces or full-blown single-page applications, Vue offers an intuitive and developer-friendly approach. Vue's reactive data system ensures that the UI remains in sync with the underlying data, providing a streamlined development experience.

Vue's incremental adoption capability makes it a versatile choice for developing web interfaces of varying complexities. For an internal staff-operated recommendation system, Vue can offer a straightforward and clean UI, facilitating easy monitoring and management. Its plug-and-play nature, combined with an active community, means there are ample resources and plugins available to embed analytics, visualization tools, or other utilities tailored for recommendation system management.

3.8.3. Angular

Angular is a robust, TypeScript-based open-source web application framework provided by Google. Distinct from its previous version confusingly enough called AngularJS, Angular offers a set of tools for building web applications. It integrates two-way data binding, dependency injection, and a modular architecture, providing a comprehensive web development toolkit.

Angular's all-encompassing framework can be particularly useful when devising a sophisticated admin interface for recommendation systems. Its two-way data binding ensures that any adjustments made within the admin panel are instantaneously reflected in the live recommendation system. Angular's structured environment might be especially beneficial for teams that prefer a stringent development paradigm. For a website where real-time updates, intricate feature sets, and integration with backend services are most sought after, Angular offers a resilient and scalable solution.

This however seems an overly complicated system for building a few basic pages. Therefore it does not seem like a suitable choice here.

With not a lot of difference between these systems apart from their development team there is not much to compare. The easy choice here is to go with ReactJs as it is used by the client company's main website.

3.9. Recommendations

After detailed research and due diligence I arrived at the following systems chosen for reasons discussed above. In summary they are:

- Scikit-learn for building the recommendation model
- Flask for backend API server
- Sqlite as only user data and metadata is to be saved
- Reactjs for frontend admin website

3.10. Conclusion

The research phase involved deciding between various programming languages, backend and front-end libraries, connection protocols, databases and security measures. The goal was to gain an understanding of the options available to create a recommendation system that could integrate with the client's website.

In conclusion, this early stage of the project has produced a logical set of systems and tools in order to build a reliable and effective SSR system. The software components, including the recommendation model, REST API, and website interface contribute to the system's ability to provide personalized and relevant hotel recommendations for Voordeeluitjes.nl

4. Functional Design

4.1. Introduction

In this document I provide the SSR's (Smart Search Results) functionalities. It lays out intended behavior, features and what users can expect from the system.

4.2. Overview

The objective of the SSR system is to provide personalized recommendations based on the visit history of a particular user. Since the Voordeeluijtes.nl website doesn't have a user login system, the recommendations are based on visited hotel history of currently unlogged users and these recommendations are predicted based on booking history of the hotel in question with a combination of other booked hotels by the same user id. This user id is identified to be the email address of booking.

4.3. Requirements

4.3.1. Functional requirements

As per project specifications following are non-functional requirements that specify what the system should do for the users and the administrators.

Ref	Description
F1	The system should generate personalized recommendations based on the user's interests and preferences, which are derived from their past booking history, current website interaction and other relevant data sources.
F2	The personalized recommendation results should be fetched by VDU's existing booking website.
F3	The system should be able to update its recommendation ML model based on the user's new data.
F4	The system should allow administrators to view various reports on the model's performance.
F5	The system should allow administrators to set custom properties or overrides for specific hotels or keywords, such as boosting or suppressing them in the personalized recommendations. The system should allow administrators to activate or deactivate the model or its components.

4.3.2. Non-Functional Requirement

These are requirements that specify how the system should perform or behave in terms of quality, reliability, security, usability, etc

Ref	Description
NF1	The system should be able to handle a large number of requests without compromising performance or accuracy.
NF2	The system should be secure and protect the data and the model from unauthorized access or manipulation.
NF3	The system should be user-friendly and intuitive for both users and administrators, providing clear instructions, feedback and error messages.
NF4	The system should work with the existing website system.
NF5	The user's data that will be processed by ML should be anonymized by removing any user's names and contact details.

4.3.3. User Stories

As a client, I want to...

Ref	Description
US1	see personalized recommendations for hotels and destinations based on my interests and preferences, so that I can find the best options for my holiday.
US2	..have a new option to sort results by personalized recommendation along with other existing sorting functions (price, rating, popularity), so that I can easily compare and choose the most relevant results for me.
US3	..see images, ratings, prices and other details of the hotels or destinations in the personalized recommendations, so that I can get a clear and appealing overview of the options.
US4	..provide feedback on the personalized recommendations, such as liking, disliking or rating them, so that I can improve the quality and accuracy of the recommendations for me and other users.

As an administrator, I want to...

Ref	Description
US6	..view various reports and insights on the model output, user segments, user profiles and conversion rates, so that I can evaluate the effectiveness and impact of the personalized recommendation system.
US7	..set custom properties or overrides for specific hotels or keywords, such as boosting or suppressing them in the personalized recommendations, so that I can adjust the results according to business needs or preferences.
US8	..activate or deactivate the model or its components, such as using only collaborative filtering or content-based filtering or hybrid methods, so that I can experiment with different approaches and methods for the personalized recommendation system.
US9	..update or retrain the model with new data, so that I can keep the personalized recommendation system up-to-date and relevant.
US10	..monitor and manage the machine learning model that generates the personalized recommendations, so that I can ensure its optimal performance and functionality.

4.3.4. Use Cases

Here are some use cases as per UML(Unified Modeling Language).

Actor	Unregistered user
Description	The user can request to get an access token in order to access the system.

Actor	Registered user
Description	The user can access various APIs but only with valid access token

Actor	Admin user
Description	Enable or disable user accounts by enabling or disabling registered users' access tokens

Actor	Registered user
Description	The user search for hotels for which user wants to see recommendations for

4.3.5. User Interface

The user interface should have white background. Where feasible information should be presented by way of lists. A search bar should be present to filter through the list. Users should be able to easily navigate through the website. The website should have possible success or error messages.

The user interface should feature a user-friendly homepage with a search bar and clear navigation options. When a user does a search, the results page updates with hotel recommendations in a list format, with each recommendation showing hotel names and other relevant details. Showing images for each hotel is nice to have.

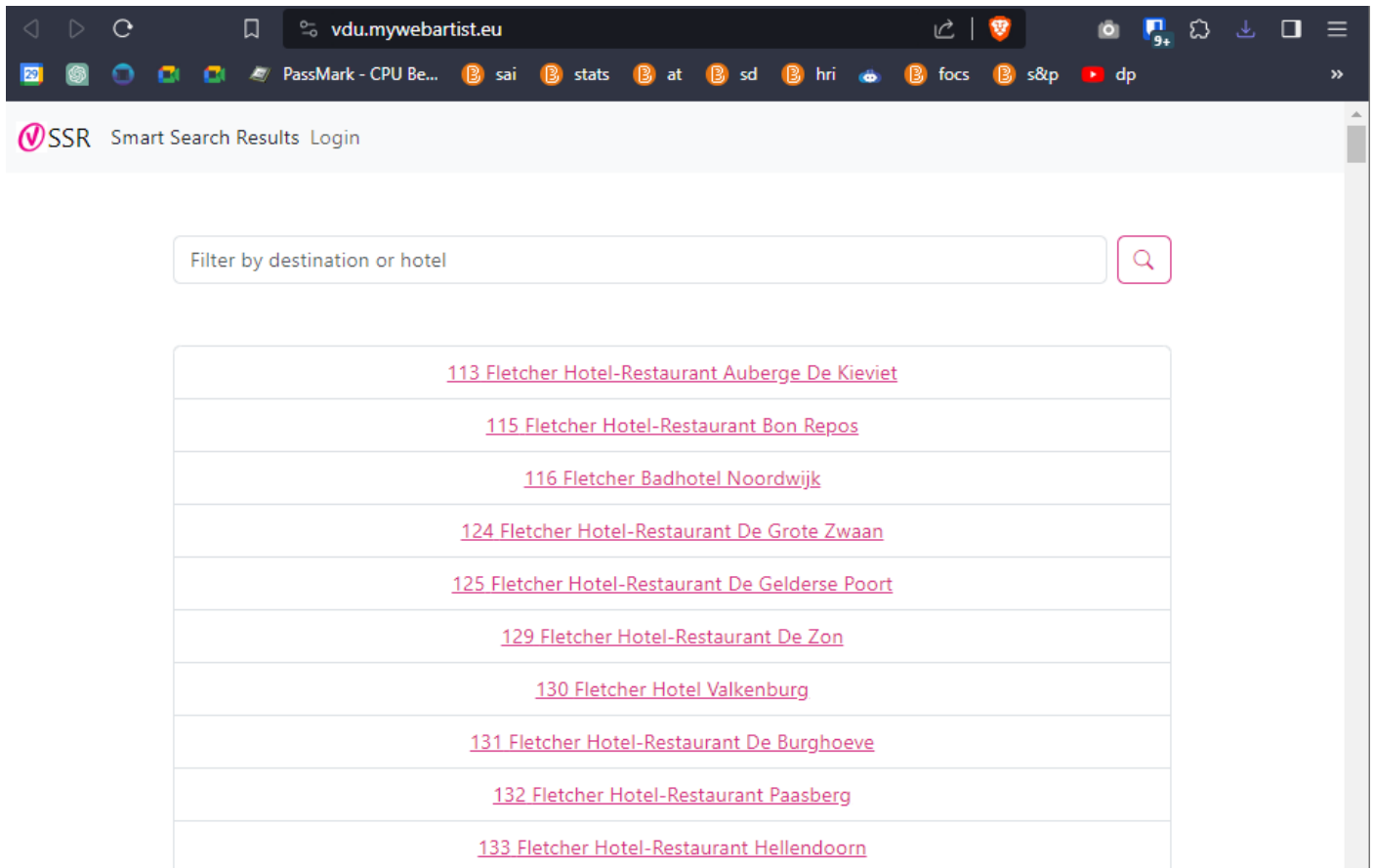


Fig. Website front page

4.4. Data

The SSR application has two types of data. One is user data stores user information. This user information contains login details and identifiers with which the user can be identified. The other source of data is from the Voordeeluitjes.nl website. The data is directly kept uploaded to the server and not added to the code repository. The user data should be anonymized, with names and contact details removed or encrypted in order to protect user privacy.

4.5. Assumptions

It is assumed that the Voordeeluitjes.nl has historic data available with hotel details and also booking records. It is also assumed Voordeeluitjes.nl's staff member is able to consume the SSR's API system to use

it in their existing website. It also assumes that server resources, such as memory and processing power, are adequate to support the system's scalability.

4.6. Acceptance Criteria

The SSR system should minimally have the following features.

- Give recommendations of hotels
- Generate similar hotels based on selected hotel
- Have user registration and login system
- Users should be able to update users' personal details
- Have response time less than 6 seconds on admin website
- Have API response time of less than 1 second
- Data protection that aligns with GDPR requirements

5. Technical Design

5.1. Introduction

This Technical Design document is intended for developers of the system. It provides architectural information and technologies needed to build the SSR(Smart Search Result) system. Once the system is built this document is ideal to refer to when seeking technical details.

5.2. System Architecture

The system is built with a model version of MCV(Model View Controller) pattern design in mind. It is not strictly MVC but follows a variation of it(explained below).. This design choice is well adapted in the industry in order to separate concerns(keep modularity). The system has three main components.

- Backend REST API (Python Flask)
- Front end admin interface (React Js)
- Database (Sqlite)

Model: This part of the system handles data and logic processing. It receives requests via REST API to do CRUD(create, read, update, delete) operations namely GET, POST, PUT, DELETE in HTTP terms. For the hotel recommendations the API takes in hotel id for which it returns hotel ids. The user management is also dealt via API system. There is security in place via a token which needs to be sent as header to be granted access.

View: The view is the admin website built using React. I chose React for its popularity and also for the fact that Voordeeluitjes' website is built using this. The SSR is meant to be used by Voordeeluitjes' website directly from the backend API; however this admin website can be used by their sales team or other developers to see the data in a user-friendly way. It also serves as a proof concept to demo which hotels are being recommended in order to evaluate SSR's capabilities.

Controller: Conventionally, the controller is a middle-man between Model and View, which handles user inputs and sends it for data processing and returns values back to the view. In this design case the controller is distributed in model and view. Since React can handle some of user interaction and data flow the rest is handled by Flask. The controller otherwise is mainly embedded in the Flask if accessed purely via API.

This mixed architecture is sometimes referred to as “backend for frontend” or vice versa and this is a practical industry approach for building modern web applications like this one.

5.3. Database Design

The main data storage requirement posed in this project is to store user's login details. All the data pertaining to the recommendation model are from CSV(comma separated values) files. This data is located in Voordeeluitjes.nl's database service which runs on mySQL. When designing the machine learning model relevant tables and columns were exported into CSV files. From these CSV files the SSR system produces the recommendation model.

Between few options for a database technology from SQLite, MySQL and PostgreSQL etc, Sqlite is chosen for the database as all relevant business logic data comes off of CSVfiles exported from Voordeeluitjes' website. Sqlite has fewer functions but it is still widely used such that Android devices use Sqlite for its reliability. These CSV files are uploaded and stored in a data folder from which the machine learning model loads these files in order to process the machine learning part. Pandas library is used to create a data frame which is similar to a database object.

All data is processed from it in order to come up with machine learning recommendations. Since there isn't any need for a separate database system as lite version of SQL is used to save other information such as users authentication access tokens, their permission level and logging information. The logging information is to keep record of all requests made to the system.

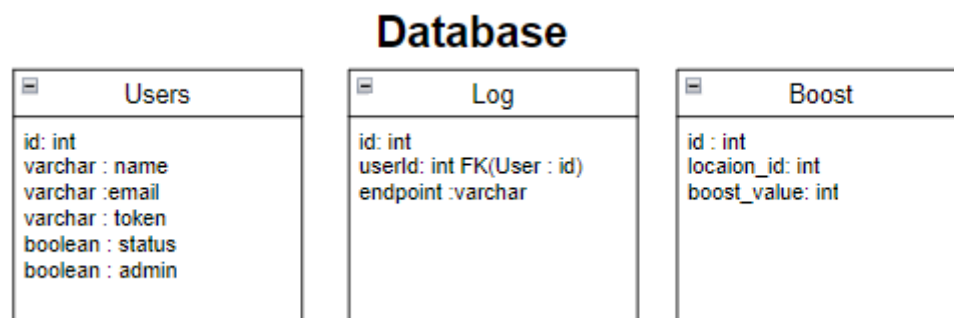


Fig BKDB. Database design at backend

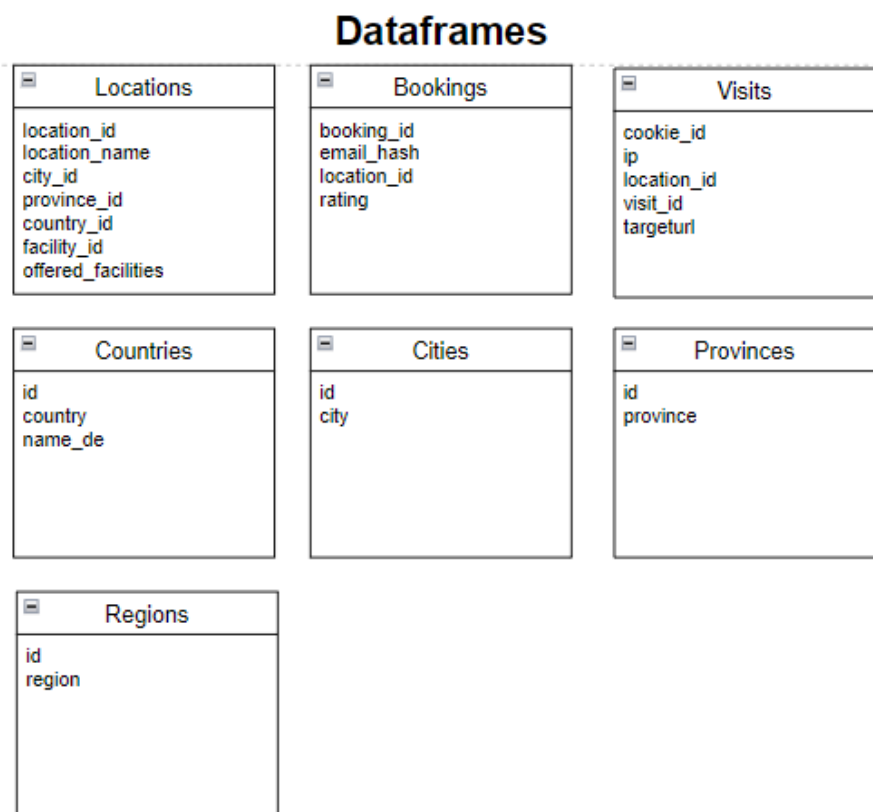


Fig BKDF. Dataframes at the backend

5.4. Backend Design

As already mentioned for backend API Python is used along other libraries listed below. Python is the industry standard for its extensive libraries and making machine learning algorithms. The Flask library is widely used for API development.

From the CSV files in the dat folder the backend will read those files in order to generate a recommendation model. The class diagram below shows the classes implemented for the backend. Interestingly enough the User class required many more functions and skills for user administrations and user authentication.

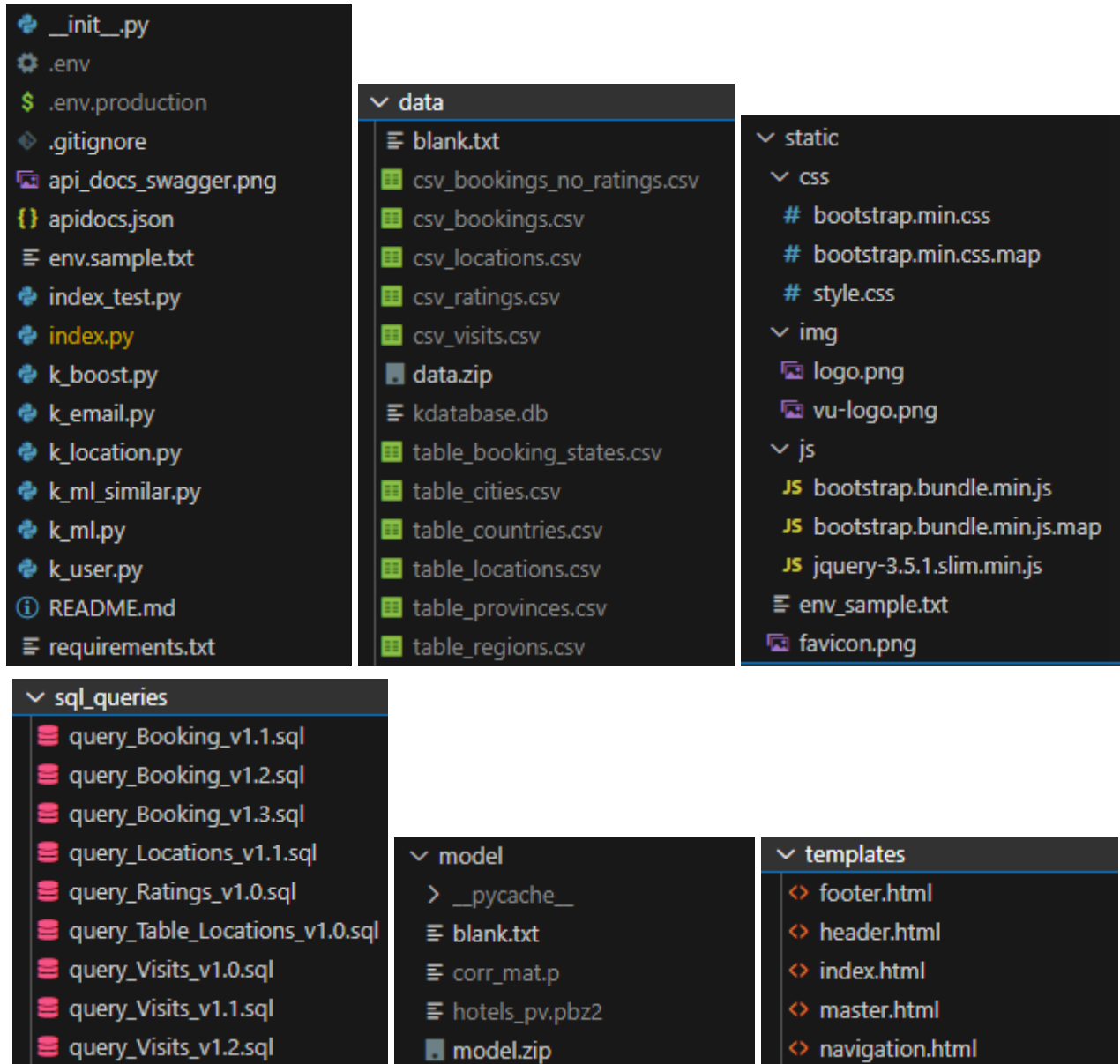


Fig. Backend files

The system uses classes to keep separation of concerns and follows an OOP(Object Oriented Programming) design pattern.

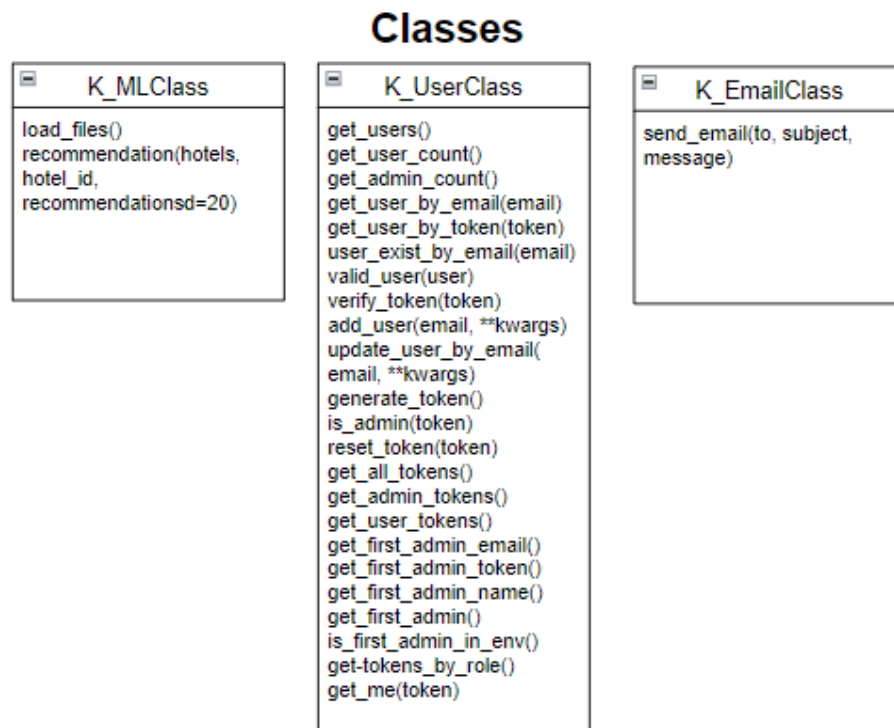


Fig BKCL. Classes at backend

5.4.1. Libraries and systems for backend development.

Email validator 2.0.0.post2

Flask 2.3.2

<https://flask.palletsprojects.com>

Flask-cors 4.0.0

Flask-swagger-ui 4.11.1

Flask-swagger 0.2.14

Flask-httpauth 4.8.0

Marshmallow 3.20.1

<https://marshmallow.readthedocs.io>

Nltk 3.8.1

Pandas 2.0.3

Python 3.11.4

<https://www.python.org>

Python-dotenv 1.0.0

Pytest 7.4.3

Scikit Learn 1.3.0

<https://scikit-learn.org>

SqlAlchemy 2.0.20

<https://www.sqlalchemy.org>

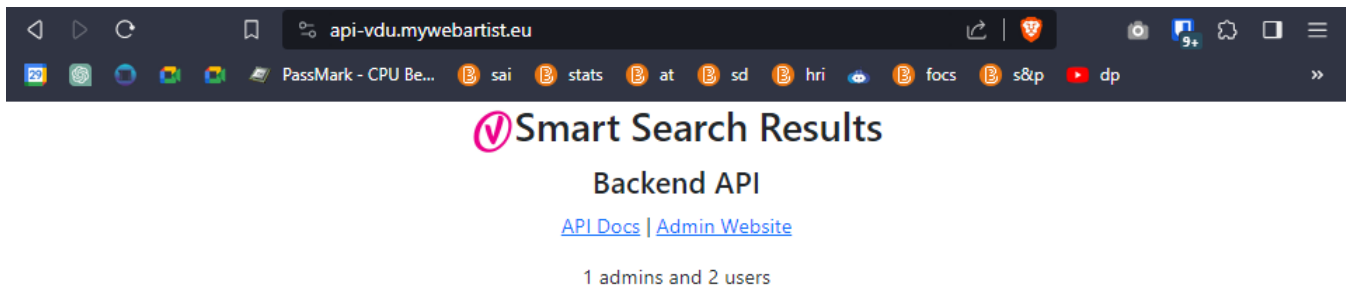


Fig. Backend API on demo server

5.5. Recommendation Model

One issue I faced was that Voordeeluitjes does not currently make use of the user login system. That means implementing personalized recommendations was not straight forward. To get around this issue I still used the standard user-item matrix taking into account ratings then instead of inputting user_id and location_id, the system requires only location_id effectively skipping the need to enter user_id. This approach may not give optimal recommendations however it does pick up hotels which would probably be of interest for someone who seems to be interested in the location_id queried for.

5.5.1. Model Logic

This data is the booking history of users and hotel facilities information. With this information two types of models are to be trained. First model is for prediction likeness of hotels and second is similarity of hotels based on facilities offered.

The first logic behind hotel recommendations is based on booking history where a user has booked more than 1 hotel. In these cases there is some sort of relationship between these hotels and the user. With this information one can assume that if a visitor from the website is checking out a hotel then there is a higher chance that this website visitor may also be interested in booking another hotel.

The second recommendation logic revolves around similarity of hotels. For example if two different hotels have similar facilities such as pool tables, restaurant, swimming pool, bar etc then we could calculate a similarity between these two hotels. This way if a website visitor visits one hotel then we could say that there is a higher probability that this user may also be interested in booking the other hotel. This hotel similarity approach is different to using filter method on website because as soon as user selects for example bar in the filter any hotel without bar will be filtered off the list, this is not the case with this system as hotels search results are calculated depending on overall score of facilities common between all the hotels in the dataset.

5.5.2. Location(Hotel) Recommendation

The steps of creating location recommendations are as follows:

Step 1: Read csv file with email_hash, bookings and ratings.

Step 2: Create user-location matrix(pivot table). For every email_hash there is a location with given rating

Step 3: Create correlation matrix using Cosine similarity. This took 22 seconds.

To find the recommendations for a particular hotel, use the matrix to look up columns for that location and sort the column where highest similarity is on the top.

5.5.3. Model management

The model's administration and customization are important because the model needs to be updated on a timely basis in order to keep recommendations current. This could be managed via the admin interface to rebuild the recommendation model. This gives the company flexibility to make changes to the search results. Administrators can make adjustments based on business needs such as the sales team. Customization allows administrators to set custom properties or overrides for specific hotels on specific national holidays, seasons or hotel availability.

Moreover, administrators had the flexibility to activate or deactivate specific model components, experiment with different recommendation approaches, and keep the system up-to-date by updating or retraining the model with new data. This level of customization meets Voordeeluitjes.nl's goals and objectives, marking a significant contribution in the research project.

5.5.4. Model Testing

In order to test how well the recommendation model performs there are various types of tests. Usually this is measured in terms of accuracy. Some accuracy measures for the regression model are Mean Absolute Error(MAE), Mean Squared Error(MSE) and Root Mean Squared Error(RMSE).

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

Fig. Accuracy formula

In the model developed I have used correlation matrix to generate item-item similarity based on different users anonymously identified by email addresses encrypted by MD5 hashing. This model does not predict ratings, rather find the closest match of hotels based on a similarity matrix created based upon previous bookings and their ratings. So these correlation similarity matrices are actually not predictions but a calculation based on ratings value given by different users, for this reason the appropriate approach to test the accuracy cannot be met by MAE, MSE or RMSE rather something called Hit Rate.

Hit Rate measures the percentage of users for whom at least one relevant item was recommended. It's a simple yet informative metric to evaluate the performance of a recommendation system. For this reason in my model testing I decided to use the Hit Rate metric to evaluate my model. In order to calculate Hit Rate you first have to collect a list of recommended items and ground truth(actual booked places) for each user. Then for each user check if there is at least one overlap between the recommended item and ground truth and if there exists one this is counted as a hit for the user. Divide the number of users with at least one hit by the total number of users and then multiply by 100. This gives a Hit Rate for the model based on the given data.

In order to understand how to achieve a high Hit Rate. I used different sizes of input data to see how the Hit Rate changes. Here is table of my findings:

Hit Rate % = (Users with at least one hit / Total users)*100

Data Lines	Lines with Ratings	Usable Data	Hit Rate
100	0	n/a	0%
1000	91	9.1%	83.52%
10,000	1180	11.8%	83.07%
50,000	6,463	12.9%	82.27%
100,000	13,098	13.0%	81.36%
250,000	33,152	13.2%	82.98%
455,000	65,039	14.2%	82.27

Fig BKHR. Hit Rate for given size of data set

Lines with Ratings	Training Time
0	n/a
100	0.02 second
1,000	0.08 second
2,000	0.19 second
2,500	0.27 second
5,000	0.57 second
10,000	1.41 seconds
20,000	3.59 seconds
65,039	23.41 seconds

Fig. Training time for given size of data set

5.6. Frontend Design

Only in the development environment the frontend requires installation of Node Js. However on build the files are regular HTML, CSS, JS files which need to be placed on to the server.

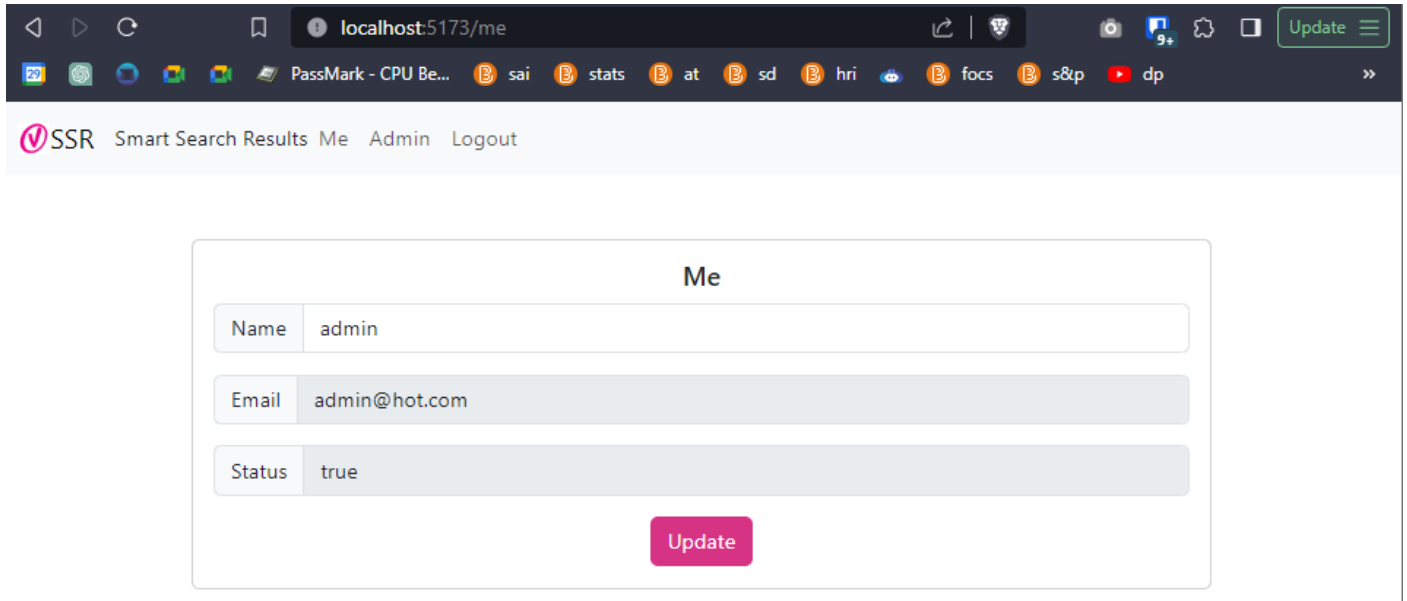


Fig. Details of user with valid access token

The file structure of the React website system is given below. Separate reusable components have been created in order to follow best practices such as DRY(don't repeat yourself). A central storage has been created to centralize the data as the user interacts with the website which is updated to provide a smooth interactive rich user experience.

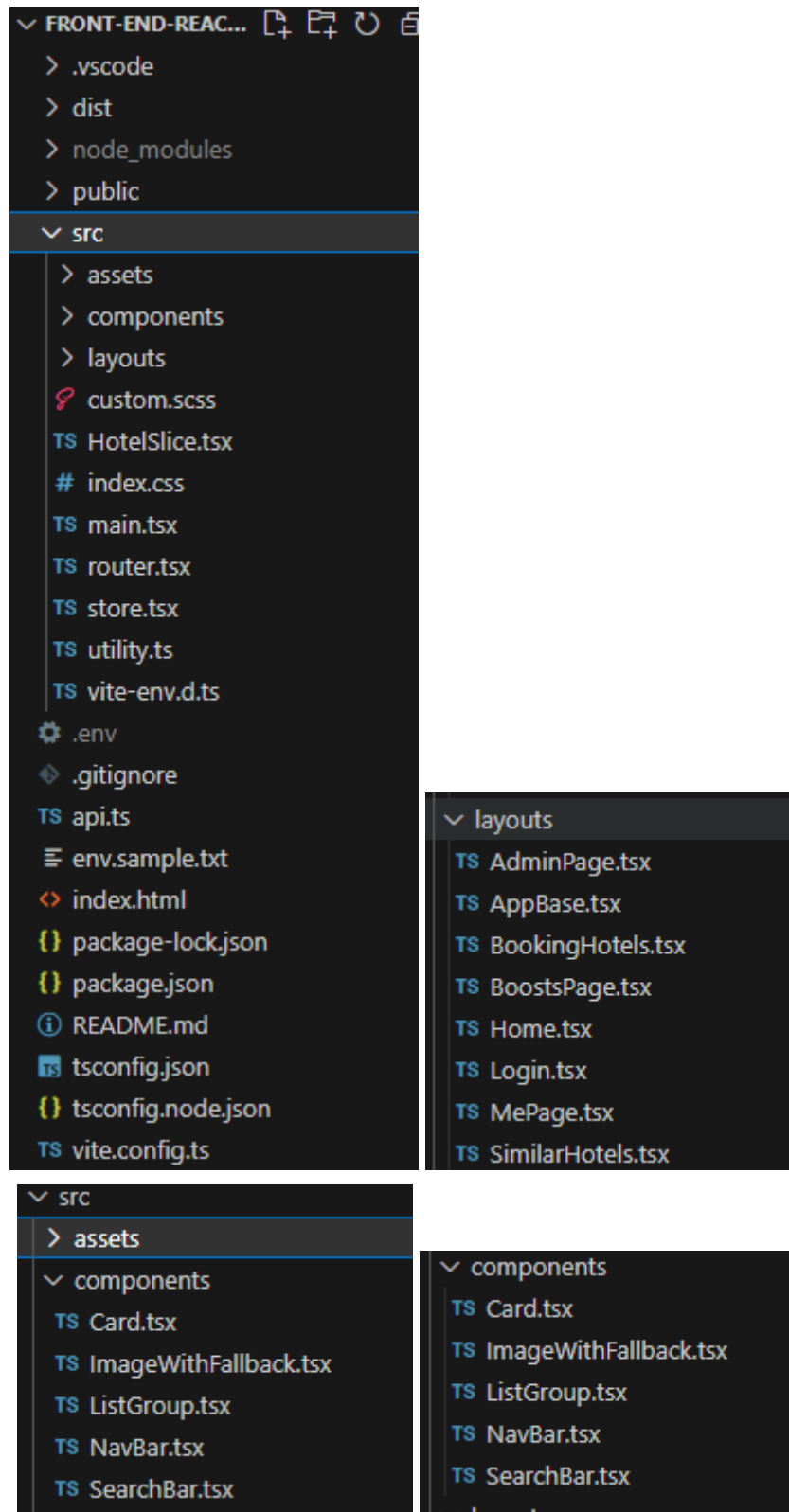


Fig. Frontend development files

5.6.1. Libraries and systems for frontend development.

@wojtekmaj/react-daterange-picker 5.4.3

Bootstrap 5.2.3 <https://getbootstrap.com>

Bootstrap-icons 1.10.5

Dotenv 16.3.1

React 18.2.0 <https://react.dev>

React-dom 18.2.0

React-redux 8.1.2

React-router 6.14.2

React-router-dom 6.14.2

Sass 1.64.2

Node 18.14.2 <https://nodejs.org>

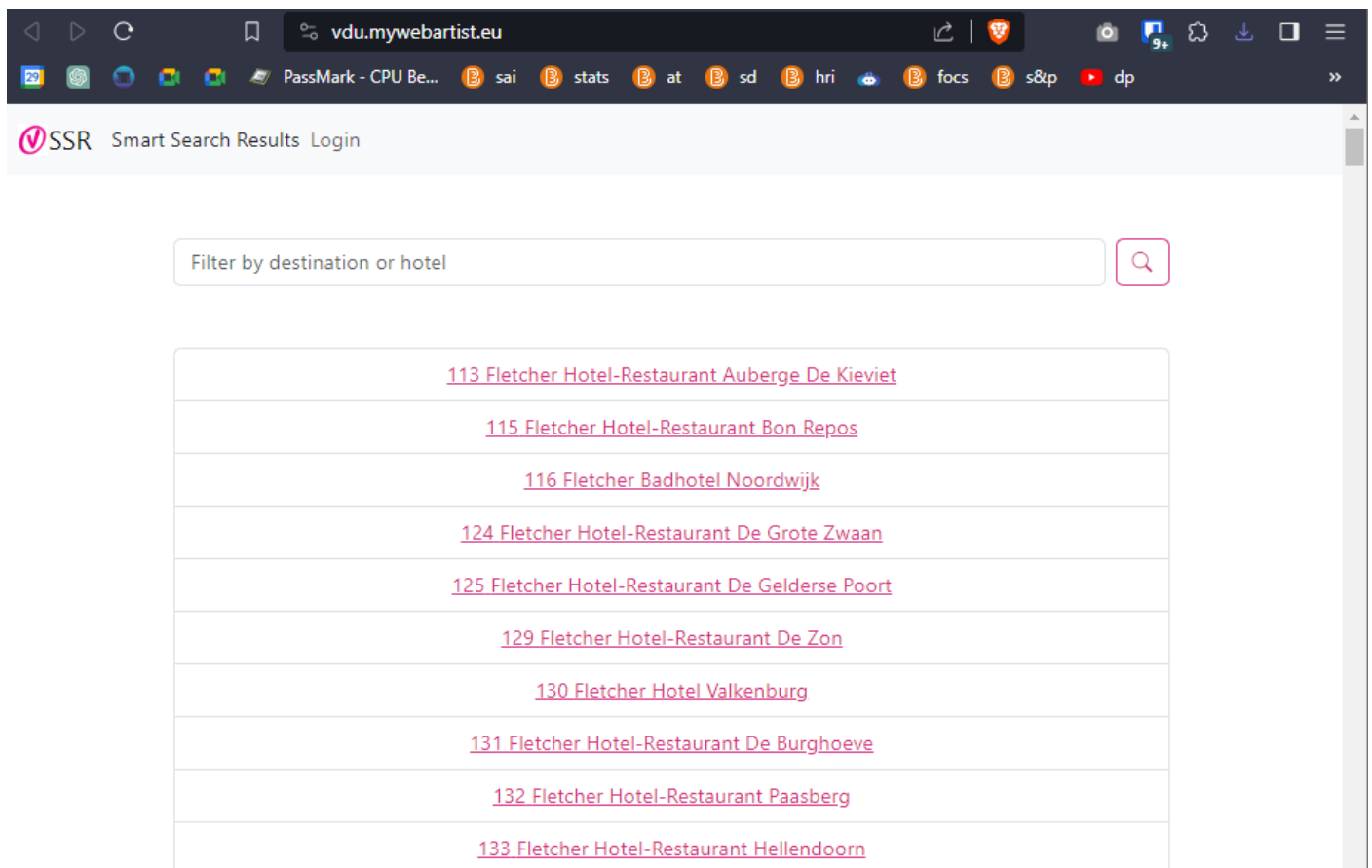
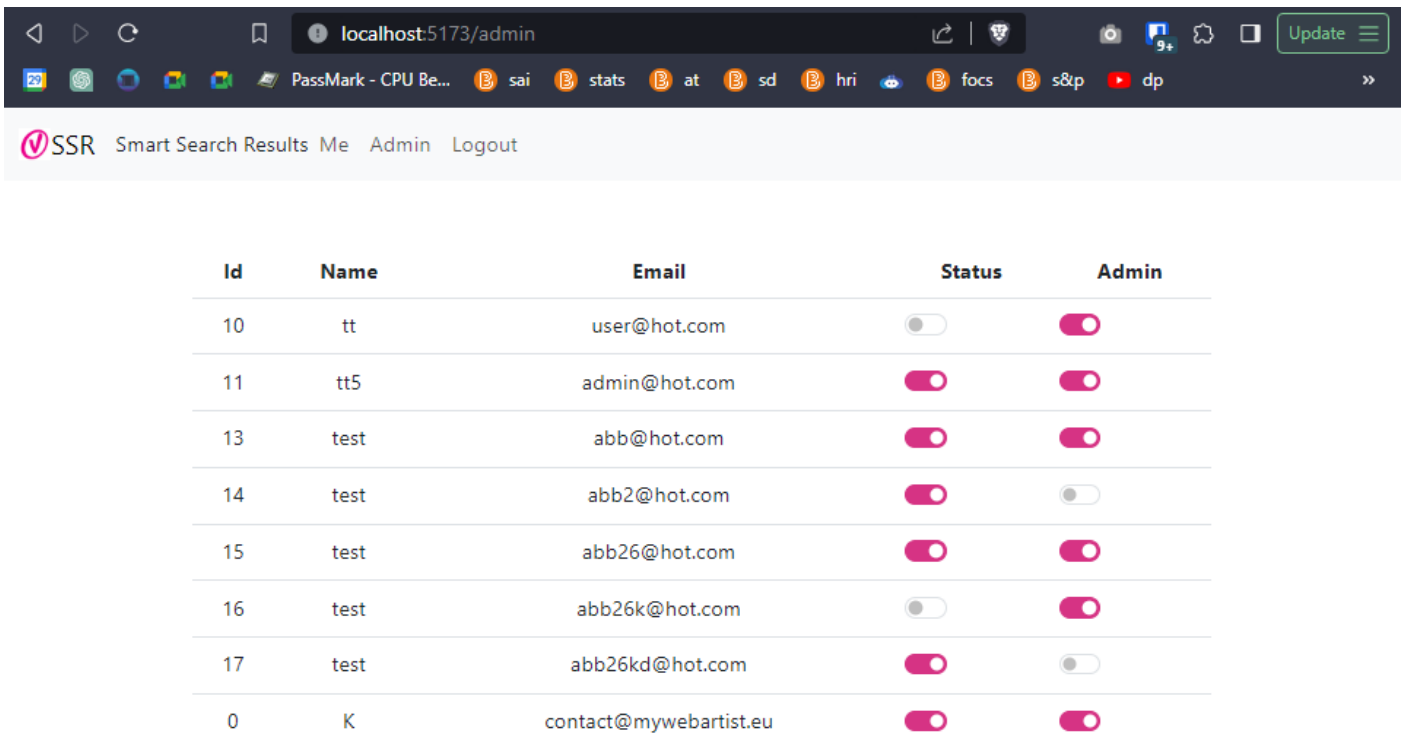


Fig. Frontend admin website

5.7. Security

A secure application is a must in today's data privacy requirement. For example GDPR requires safety of the user's personal data. Utmost care has been taken to make sure that no client's private data is exposed outside of staff who are supposed to have access to such data.

The SSR application requires booking history in order to train a machine learning model. The booking history includes private details of customers such as customers name, address, email etc. In order to protect user's private data upon export of the CSV files none of such data is used. However in order to create correlation of bookings by the same user email of booking person is needed. In order to avoid leak of email addresses the email column has been encrypted by using one way MD5 hashing technique. This way each email generates unique encrypted text for every email which confirms the booking details by one user, this process is the basis for creating the booking matrix needed to generate recommendation models



Id	Name	Email	Status	Admin
10	tt	user@hotmail.com	<input type="checkbox"/>	<input checked="" type="checkbox"/>
11	tt5	admin@hotmail.com	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
13	test	abb@hotmail.com	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
14	test	abb2@hotmail.com	<input checked="" type="checkbox"/>	<input type="checkbox"/>
15	test	abb26@hotmail.com	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
16	test	abb26k@hotmail.com	<input type="checkbox"/>	<input checked="" type="checkbox"/>
17	test	abb26kd@hotmail.com	<input checked="" type="checkbox"/>	<input type="checkbox"/>
0	K	contact@mywebartist.eu	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Fig. User administration page by admin

The SSR system has the option to be deployed in a closed server which would be exclusively accessible by the Voordeeluitjes.nl's website. This approach would be secure as no outsiders can make direct requests to the SSR system.

Another approach would be to expose the SSR system to the internet but only those with an access token would be able to use the system. There has been a fully functional access token system created to allow only authorized requests to the system.

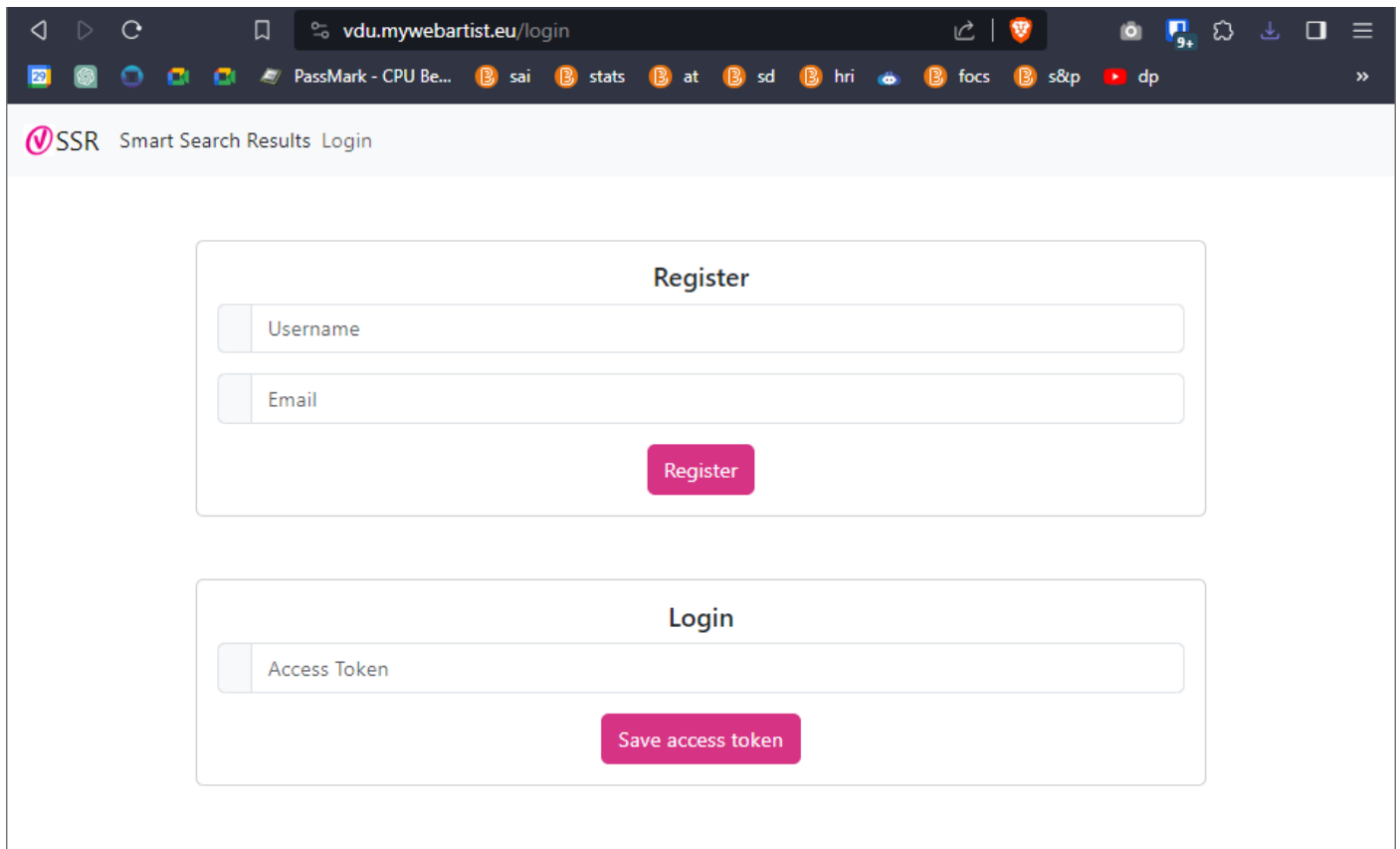
The image is a screenshot of a web browser displaying the login page for 'vdu.mywebartist.eu'. The browser's address bar shows the URL. The page has a dark header with navigation links: 'SSR', 'Smart Search Results', and 'Login'. The main content area contains two forms. The first form, titled 'Register', has input fields for 'Username' and 'Email', and a pink 'Register' button. The second form, titled 'Login', has an input field for 'Access Token' and a pink 'Save access token' button.

Fig. Registration and login at Frontend

There are two access roles, Admin and User. Admin users have elevated permissions which can enable and disable access of other users. A standard User could access the machine learning API endpoints. The benefit of this approach is that Users can access the recommendation data via internet access. This would be useful for the sales team to see what recommendations the SSR system is giving on certain hotels. Another use case could be that Freetime Company B.V contracts a third party vacation booking website to sell the recommendations generated by the SSR's ML model.

The demo server is behind Cloudflare(<https://www.cloudflare.com>) protection which offers various access attacks specially DOS(denial of service) attacks. The server administrator should keep the server well updated with security patches on time to make sure new viruses and malwares are kept at bay.

5.8. Performance

Any system which is slow and crashes has serious questionability of being used. Voordeeluitjes.nl operates in a competitive travel industry and it is crucial that the potential customers do not have to wait long(< 10 secs) for a page to load. Before they decide to include a new system, it must not cause errors or slow down their existing system.

For an SSR system to work optimally there needs to be considerations made for the server specifications taking into account CPU, RAM and disk space. The administrator needs to keep in mind scalability requirements by estimating current and future traffic size so that the system can handle increased request load. At bare minimum a server should be Ubuntu 20.04, Intel pentium, 1GB RAM and 30 GB disk or

higher. In order to scale for performance and handling Vertical scaling should suffice as the system would be able to handle more requests and return responses quicker.

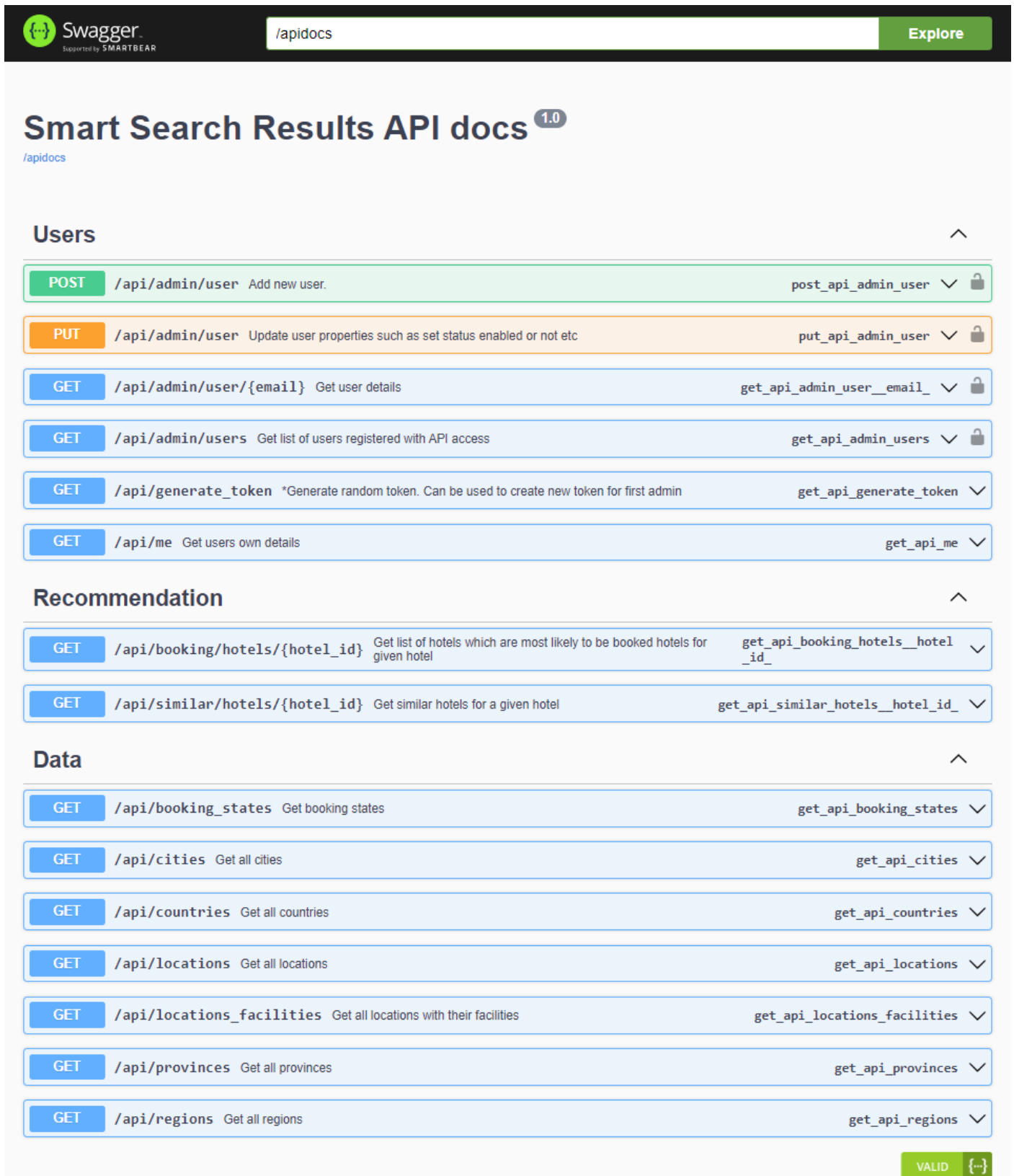
Caching is a good technique to speed up retrieval of data. Since the system uses a data frame all the information is already in memory however there could be further speed increase by using a library called Flask-Caching. This has been implemented to avoid reading from sqlite- or recalculate repeated calls unnecessarily which could burden the system. This cache is updated depending on cache expiration time which could be minutes, hours or until the next update of new incoming data at which a new model is trained to replace the previous one. Currently without using cache at the local development system the API returns response in 16ms, while at the demo server the API returns response in 200ms. This response time could change depending on the server specification where Voordeeluijes.nl deploys the live application.

For troubleshooting purposes, the SSR system has been equipped with a logging trail. At every request the system is going to log in the log table. This can help diagnose system errors and potentially uncover unauthorized access.

5.9. API Documentation

This section describes how the API endpoints have been documented plus how and where these could be accessed from. The documentation is generated by Swagger library which creates a nice interface to layout all endpoints. This could accessed from here <https://api-vdu.mywebartist.eu/swagger>

Some endpoints are open, for example a register requires a secret token as this is open for new registrations. Some endpoints require parameters for example name and email when sending registration requests. These parameters are well defined and clearly stated in the documentation. In case of incorrect request an error message is returned with possible cause of failed request. The API response is always JSON() format. There is always a success or error key available so that the system calling it knows whether the request was successful or not. In case of error, there is an error message sent back. On top of that the response also sends back “status code” for example 400 for bad requests or 404 for resources not found.



The image shows the Swagger UI for the 'Smart Search Results API docs' version 1.0. The interface is dark-themed with a top bar containing the Swagger logo, the API name 'Smart Search Results API docs 1.0', and an 'Explore' button. Below the top bar, the API name is repeated in a large font. The main content area is divided into three sections: 'Users', 'Recommendation', and 'Data'. Each section contains a list of API endpoints with their methods, descriptions, and function names. The 'Users' section has six endpoints, including POST, PUT, and GET methods. The 'Recommendation' section has two GET endpoints. The 'Data' section has six GET endpoints. At the bottom right, there is a 'VALID' button and a JSON schema icon.

Method	Endpoint	Description	Function Name
POST	/api/admin/user	Add new user.	post_api_admin_user
PUT	/api/admin/user	Update user properties such as set status enabled or not etc	put_api_admin_user
GET	/api/admin/user/{email}	Get user details	get_api_admin_user_email
GET	/api/admin/users	Get list of users registered with API access	get_api_admin_users
GET	/api/generate_token	*Generate random token. Can be used to create new token for first admin	get_api_generate_token
GET	/api/me	Get users own details	get_api_me

Method	Endpoint	Description	Function Name
GET	/api/booking/hotels/{hotel_id}	Get list of hotels which are most likely to be booked hotels for given hotel	get_api_booking_hotels_hotel_id
GET	/api/similar/hotels/{hotel_id}	Get similar hotels for a given hotel	get_api_similar_hotels_hotel_id

Method	Endpoint	Description	Function Name
GET	/api/booking_states	Get booking states	get_api_booking_states
GET	/api/cities	Get all cities	get_api_cities
GET	/api/countries	Get all countries	get_api_countries
GET	/api/locations	Get all locations	get_api_locations
GET	/api/locations_facilities	Get all locations with their facilities	get_api_locations_facilities
GET	/api/provinces	Get all provinces	get_api_provinces
GET	/api/regions	Get all regions	get_api_regions

Fig. API Documentation via Swagger

The documentation contains description, address, method, permission level and example response for each endpoint. Given a valid access token one could run the commands directly from this Swagger documentation

page to see what input and outputs are given for the recommendation system. During the development, a famous system called Postman was used to test the endpoints. Postman also has a feature to publish the endpoints as a Collection. These are available from this link <https://documenter.getpostman.com/view/11320596/2s9YJeygrm>

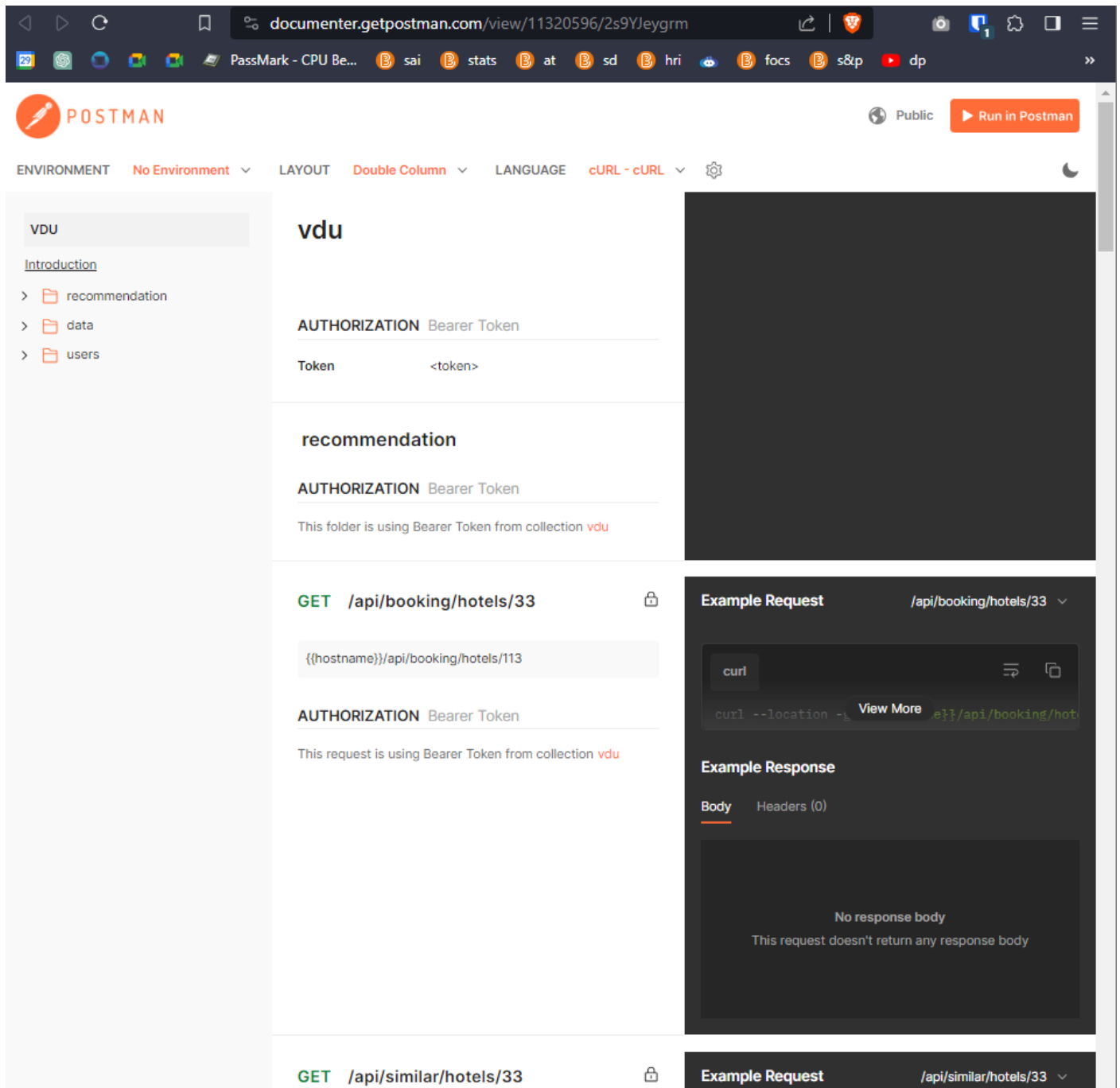


Fig. API documentation via Postman app

It is to be noted that not all endpoints are open and require a valid access token. To retrieve this token one needs to send the register at /api/user using POST command and then get the token in their email. An admin must first enable the account before the token is accepted.

5.10. Testing

Testing is another enormously important area of software development. I applied various testing techniques to create a reliable product. Different testing covers different testing aspects of the system, these are unit testing, integration testing, user acceptance testing(UAT). During development, testing is merely a matter of trial and error in order to get the program to work as expected. Later on automated testing is used as it is not practical for one person to check the whole system after every modification.

Apart from manual testing to see if the program works as expected, I implemented unit testing using the Pytest framework for the backend Python code for three functions. For example, the recommendation generation algorithm was extensively tested to verify that it produces accurate recommendations based on input data as well as the user class and the email class. For user acceptance testing I asked the team at Voordeeluitjes.nl's to try out the system at the demo server. For integration testing I ensured that the backend Flask API and the React frontend integrated seamlessly without any errors on the website as well as in the console log. This involved testing API endpoints to confirm that data is correctly transferred between the frontend and backend. The company staff used the admin interface and provided feedback on the usability and functionality of the SSR system. The SSR system consistently achieved response times of less than 1 second which is good speed.

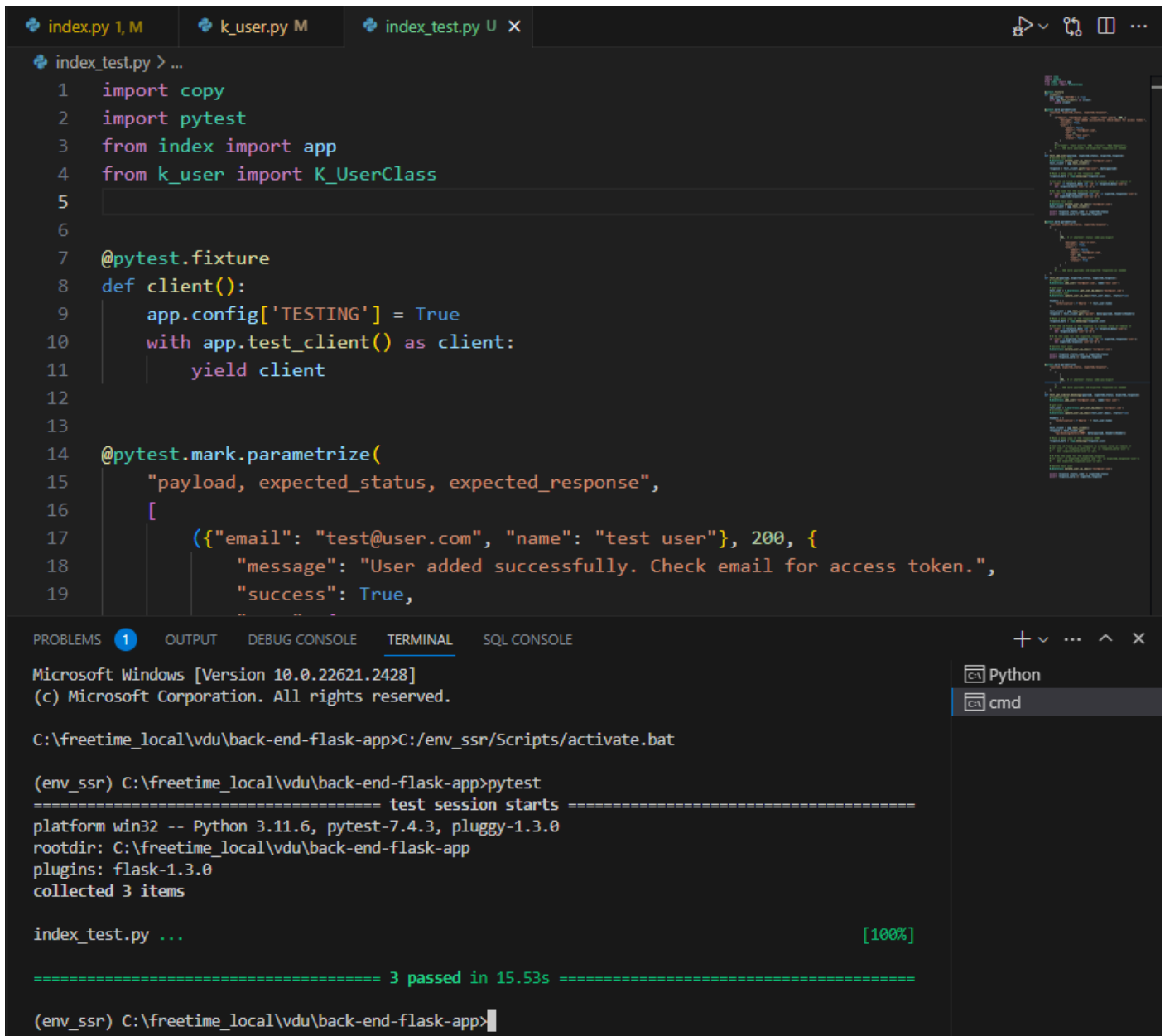
The three functions actually test more functions as some functions are dependent on other functions. The three functions are as follows:

- `test_add_user()`
 - As it appears this will add a new user to see if the user is registered successfully.
- `test_me()`
 - This is the user's profile page. This tests whether user is able to successfully get access to the system and get their details
- `test_get_similar_bookings()`
 - This test compares the output of recommendations based on what data was used to train the model. The test here is to see whether the function returns data as expected. This test is not intended to test the accuracy rate for booking recommendations. For that see Hit Rate in [Model Testing](#) section for this.

Security testing included scanning open ports to allow for the only port on which the backend is running which is port 3006. Additionally, in case real email gets exported in the CSV files the system would flag it so that only hashed emails would be processed. Testing covered scenarios where invalid requests were made to API endpoints. The system consistently provided informative error messages, this helps users understand the cause of the issue. The Swagger-generated API documentation and Postman collection were individually checked to ensure that they accurately returned data as required results.

Detailed testing played a vital role in verifying the reliability and functionality of the SSR system. It allowed me to identify and address issues in the development process. With help of testing I was able to create a robust and dependable end product.

Here are test result report showing passed tests:



```
index.py 1, M | k_user.py M | index_test.py U x
index_test.py > ...
1  import copy
2  import pytest
3  from index import app
4  from k_user import K_UserClass
5
6
7  @pytest.fixture
8  def client():
9      app.config['TESTING'] = True
10     with app.test_client() as client:
11         yield client
12
13
14  @pytest.mark.parametrize(
15     "payload, expected_status, expected_response",
16     [
17         ({'email': 'test@user.com', 'name': 'test user'}, 200, {
18             "message": "User added successfully. Check email for access token.",
19             "success": True,
```

PROBLEMS 1 | OUTPUT | DEBUG CONSOLE | **TERMINAL** | SQL CONSOLE

Microsoft Windows [Version 10.0.22621.2428]
(c) Microsoft Corporation. All rights reserved.

C:\freetime_local\vdu\back-end-flask-app>C:/env_ssr/Scripts/activate.bat

(env_ssr) C:\freetime_local\vdu\back-end-flask-app>pytest
===== test session starts =====
platform win32 -- Python 3.11.6, pytest-7.4.3, pluggy-1.3.0
rootdir: C:\freetime_local\vdu\back-end-flask-app
plugins: flask-1.3.0
collected 3 items

index_test.py ... [100%]

===== 3 passed in 15.53s =====

(env_ssr) C:\freetime_local\vdu\back-end-flask-app>

Fig. Passed test results using Pytest library

5.11. Deployment

During development all files were on the company's laptop where I used files at the local server. These files were committed to GitLab at this repository <https://gitlab.com/mywebartist/vdu>. There were mainly two branches: dev and main. At the localhost development all code was pushed to the dev branch and at the demo server dev branch was merged into the main branch from where services were started.

Among few cloud server choices such as AWS(Amazon web services), Heroku, Vercel, Google Cloud etc there were plenty of options where to deploy the SSR system for the demo. In the end I chose the demo to be deployed on a droplet at DigitalOcean <https://www.digitalocean.com> running Ubuntu 22.04(Linux variant) and Nginx server with all the necessary resources to host SSR applications. Digital Ocean is a similar cloud service provider similar to AWS(Amazon web services) and Google Cloud. The Nginx server uses a reverse

proxy to handle incoming HTTP requests along with a SSL certificate to securely transmit end to end communication.

Typically a database connection is also required, however due to design choice SQLite was used and this requires no additional setup as the database is simply a file. The Python library SQLAlchemy administers the connection between the application and database engine.

In order for the backend SSR system to run for the first time. A Python command needs to run in order to install all dependent libraries. This command is simply “pip install -r requirements.txt”. The requirements file has the correct library names along with compatible versions.

In order for the frontend SSR system to be updated this node command, ”npm run build”, needs to be run and the files generated in the “/dist” folder needs to be copied into the public folder of the host.

On updating new code from dev to main a gitlab pipeline could be added so that the new changes are automatically reflected on to the API and the frontend system. It seemed unnecessary to implement this feature specially because currently the SSR system is on a demo server. Should Voordeeluitjes.nl decide to implement the system then this Gitlab CI/CD(Continuous Integration/Continuous Deployment)pipeline would come in handy.

Sometimes the cloud services such as SSR can go down for various reasons such as system overload, system restarts or unanticipated errors. In such circumstances it is vital to get notification that the service is down. An email notification could be implemented should the service go down. It's possible to run the SSR service simultaneously on a backup server, in case disruption in the main server node the Voordeeluitjes.nl website can switch to the backup service.

5.12. Logging

It's good practice to keep a log of as many things as possible. These logs come in handy when diagnosing causes of system failure or investigating security-related events. Thankfully the SSR system does not store any client data. The SSR system logs all incoming requests along with the user id, IP address of requester and timestamp. For a system of this size and operation this level of logging seems sufficient.

Log files become very large overtime, for this reason log files should be exported and kept in a safe archive and log table cleared off to free up server disk space. A suitable retention policy like keeping previous 365 days of log files should be decided. This part of the process is to be done by an SSR administrator.

5.13. Scalability

As part of robust software development scalability should not be overlooked. For this project scalability is particularly important as Voordeeluitjes.nl website receives hundreds of thousands of visitors yearly. The search result recommendations could get just as many requests. This number is going to grow over time and so the SSR system needs to be on a server which can increase memory and CPU power overtime. This is something the company needs to determine. Server's control panel shows load levels which is a good indicator of how well the system is able to cope with the load. Load balancing via multiple servers is also a good technique. The Nginx could be set up in a way for a load balancing feature which distributes incoming HTTP requests among multiple SSR server instances. This will make sure of efficient resource utilization and keep response time as low as possible.

Conventionally database scaling is a huge undertaking, however in case of SSR the recommendation data is mostly kept in memory by way of a dataframe object. This size could grow however RAM of a couple of GB is more than sufficient to cater for this. The database in the SQLites is very small and will not grow large. This avoids scalability issues from the database point of view.

5.14. Recovery

All and any systems are prone to all kinds of failures. This leads to data loss and downtime therefore a recovery process needs to be in place. The SSR system operates upon CSV exports from the Voordeeluijes.nl website so in case of disk corruption these files could be easily replaced. The only “new” data the SSR system needs to protect is the kdatabase.db file in the /data folder of the backend system. This is because this database keeps users and logs tables which would be painstakingly difficult to redo and impossible for log data to recover. In order to backup these files a periodic cron job could be set up to copy this file on a separate server. This should mitigate the data loss.

5.15. Support

A proper support channel should be available in case the company requires assistance in ensuring smooth operation of the SSR system. Since I was the only person involved in building the system, I paid particular attention in documenting everything as much as possible. All functions come with comments so that the system could be debugged and extended in the future. Should the company require assistance they may reach me at my support email at contact@mywebartist.eu. In order to understand usability, user friendliness, and the satisfaction of the SSR system I have created a survey form in which I can get feedback. I would take this feedback in development of my future products. Survey link <https://forms.gle/aeHgizWKGWQoAzmW6>.

5.16. Conclusion

This document is a blueprint of the SSR system in which I laid out detailed architectural intricacies, technological choices, and design considerations that shaped the SSR system. This document is an emblem of my commitment to deliver a robust software solution which houses the cutting edge tool of the currently exploding field of A.I. In today's fast moving world a company requires not only a software system but a smart software which helps them stay ahead of their competitors and this is exactly what is aimed at with this project.

The system's architecture is not limited by the conventional strict MVC pattern because it is customized to fit the needs of the project. Python, Flask, React Js, and SQLite are used to create a dynamic back-end logic, front-end usability, along with data delivery.

Security concerns have been paramount and the SSR system uses access tokens to secure the application. API documentation with detailed description provides all information to get the system up and running.

6. Systems Manual

The SSR system is a complete system ready for deployment. However the person made in charge needs to get to know the system with its aspects of setup, installation and any troubleshooting before the system can run as intended while avoiding obvious pitfalls which can lead to loss of time not knowing what the issue is. This Systems Manual should mitigate such scenarios as it gives step by step instructions on how to correctly setup the system. Some basic technical know-how is expected such as working with servers with some basic programming knowledge.

6.1. System setup and Installation

The project setup is described in the 3 README.md files using markdown format. Here is a structure showing where these files are located.

vdu	→ Back-end-flask-app	→ README.md
	→ front-end-react-app	→ README.md
	→ README.md	

The content of the README.md files are provided in [Readme.md](#).

6.2. API Documentation

A complete swagger documentation is provided which can be accessed from directly from backend <https://api-vdu.mywebartist.eu/swagger> and in JSON format from <https://api-vdu.mywebartist.eu/apidocs> See [API documentation preview](#).

6.3. SQL Queries

Their system uses a database for which SQLite is used. There is a .db file which holds all the database data and there are SQL queries written in order to export CSV files from which the recommendation model is generated. The database files are located below.

vdu	→ Back-end-flask-app	→ data	→ csv_bookings.csv
			→ csv_
		→ sql_queries	→ query_booking.sql
			→ query_locations.sql
			→ query_ratings.sql
			→ query_table_locations.sql
			→ query_visits.sql

The content of some complicated SQL queries are provided in [SQL queries](#). The type of data required in csv files are shown in [CSV file format](#).

6.4. Known Issues

During development following issues were discovered. These were unresolved by the end of the project. These are documented here along with possible solutions.

- A. The menu button on the admin website does not work on mobile devices.
Possible solution: Add back missing navigation menu code from Bootstrap 5 website as something was deleted which is needed for menu burger button to work on mobile device
- B. The recommendations page on [http://localhost:5173/booking/hotels/\[location_d\]](http://localhost:5173/booking/hotels/[location_d])
- C. At odd times the frontend is presented with this error. It is obvious that this is a frontend issue. A simple refresh page resolves this but of course that is not the permanent fix here.

Unexpected Application Error!

Unexpected end of JSON input

```
SyntaxError: Unexpected end of JSON input
    at JSON.parse (<anonymous>)
    at https://vdu.mywebartist.eu/assets/index-cb720da1.js:99:130119
    at ru (https://vdu.mywebartist.eu/assets/index-cb720da1.js:40:24263)
    at Ia (https://vdu.mywebartist.eu/assets/index-cb720da1.js:40:42315)
    at Hw (https://vdu.mywebartist.eu/assets/index-cb720da1.js:40:41163)
    at Ur (https://vdu.mywebartist.eu/assets/index-cb720da1.js:40:40212)
    at rm (https://vdu.mywebartist.eu/assets/index-cb720da1.js:40:36824)
    at Ir (https://vdu.mywebartist.eu/assets/index-cb720da1.js:38:3274)
    at https://vdu.mywebartist.eu/assets/index-cb720da1.js:40:34207
```

- ~~D. The demo server stops responding if the bookings page is opened too quickly (say refresh within 10 seconds). This issue is not present on the local server. The obvious thing that comes to mind is that the demo server is only 1GB of ram which may cause it to get overloaded. The other factor is that the model needs to be modified in a way as to either store recommendations in SQLite or change the code in a way so that recommendations do not need to be calculated from the model. This fix does not require too much time to implement but it is definitely technically challenging.~~
- E. The training of the recommendation model does not work on the server. This could be because of memory limitations on the server. This function works fine locally, so if facing issue then train the model locally then upload to server in ./model directory
- F. Sometimes the menu buttons on frontend go off. Refreshing page fixes this issue
- G. Scrollbar on the right on the main page is not showing. It was there in the start but it disappeared. Initial guess is that due to the dynamic nature of list loading the browser does not know the list is long. Use the mouse to scroll down.

6.5. Unfinished parts

Due to time constraints some of the systems were unfinished. Below is a list of items which were being worked upon and were left unfinished.

- Logging table where each request and system change is recorded. What goes with that is an API endpoint which would fetch this data from this table. This then could be presented on the Admin website. This would be a nice addition in terms of keeping a good audit trail of changes and usage.
- A second model was developed in which the idea was to cluster hotels with similar facilities. This in the end was left out as it required more work. The relevant files for work done to develop this model are as follows:
 - `k_ml_similar.py`
 - `/data/csv_locations.csv`
- Boost date range widget on <https://vdu.mywebartist.eu/booking/hotels/144> is not implemented

30 Oct ▼ 2023 11:07 – 06 Nov ▼ 2023 11:07 ✕ □

6.6. Troubleshooting

This section helps to troubleshoot usual issues which can cause the API backend or Admin website to malfunction and how to go about resolving those issues.

Issue	Possible Solution
API backend system is showing bad gateway	<p>Check if you are on the correct host and port. See .env file for these system variables</p> <p>HOST=0.0.0.0 PORT=3006</p> <p>Here the 0.0.0.0 refers to localhost or 127.0.0.1 Check to see if the chosen port is open on the server. On local system this port should be open by default</p>
Access token is not sent over email	<p>Check to make sure that email SMTP settings are correct in .env file</p> <p>MAIL_FROM_EMAIL=email@email.com MAIL_SMTP_SERVER=smtp.server.com MAIL_SMTP_PORT=587 MAIL_SMTP_USER_EMAIL=email@email.com MAIL_SMTP_SENDER_NAME=email@email.com MAIL_SMTP_PASSWORD=password MAIL_TURN_OFF_EMAILS=False</p>
Unable to access upon first installation	<p>Check to see that first admin details are entered in .env file</p> <p>FIRST_ADMIN_TOKEN=haha</p>

	FIRST_ADMIN_EMAIL=email@email.com FIRST_ADMIN_NAME=K
User is not able to access the API	<p>Check to see if the access token is valid and activated. Any user with admin access can check this from front end https://vdu.mywebartist.eu/admin or by sending API GET request at https://api-vdu.mywebartist.eu/api/admin/users</p> <p>You need to send in admin access token in your request otherwise you will get unauthorized Access message such as below</p> <pre>{ "error": "Unauthorized Access - Token not found" }</pre> <p>You may enter this in the Console window of your browser in the Developer tools section. Replace the address and access token.</p> <pre>fetch(['https://api-vdu.mywebartist.eu/api/admin/users'], { method: 'GET', headers: { 'Authorization': 'Bearer [YOUR TOKEN]' } }) .then(response => response.json()) .then(data => console.log(data)) .catch(error => console.error('Error:', error));</pre>

7. Graduation Report

7.1. Introduction

This project involves software development and machine learning skills. Overtime I have been learning both of these art forms. This is my last opportunity to demonstrate that I have these skills to build a complex system which could be used by a company. Freetime Company B.V has contracted with me so that I can build a solution for them.

7.2. Problem definition

The heart of the problem is the use of conventional keyword based search results and recommendations based on traditional methods such as hotels with most bookings or most availability etc. These methods do not provide users with personalized hotel recommendations. In today's incredibly fast evolving world of A.I a recommendation system is needed which can provide the best close match to what visitors are looking for. This system needs to be encapsulated into a software system which can be securely accessed by their website outlet Voordeeluitjes.nl.

7.3. Assignment

The problem definition was to take data stored in a database in order to build a system that could be integrated into Voordeeluitjes.nl website. The system should take hotel ids as input and return multiple hotel ids which hold high probability of getting booked by visitors. This system should operate on HTTP protocol. See Technical Design report for complete details of requirements.

7.4. Design decisions

The company gave me complete freedom to come up with my solution. I worked side to side with highly experienced programmers. I was able to discuss my plans to get their feedback on their ideas. I conducted research on many possibilities to understand what might be the best way to develop this system. The process involved using my experience working with different systems and investigating what other possible systems could be used to speed up the development process.

7.5. Development environment

The development environment consisted of MS Visual Code which is an IDE(integrated development environment). The local development system also had Python 3 and Node 18 installed. I used Gitlab to store files in a repository. DB Browser for SQLite to query data. Jupyter notebook as VS Code extension was used during modeling of the recommendation algorithm. SSH to log on to Ubuntu server to transfer files for the demo server. I used Postman to try out the API endpoints to see if they give results as expected.

7.6. Results

Below are the results from the project work which lists all the function and non-functional requirements as well as user stories.

Ref	Status	Ref	Status
F1	Achieved	US1	Achieved
F2	Achieved	US2	Possible
F3	Achieved	US3	Possible
F4	Partly achieved	US4	Possible
NF1	Achieved	US6	Out of Scope
NF2	Achieved	US7	Partly Achieved
NF3	Achieved	US8	Partly Achieved
NF4	Possible	US9	Achieved
NF5	Achieved	US10	Not Achieved

Fig. Outcome from work on the project

The SSR system came together bit by bit over the course of about 6 months. The API system is available with full documentation and link to frontend at this link <https://api-vdu.mywebartist.eu>. The company appreciates my efforts in this attempt to game up against their rival booking sites. The system does what it should as outlined in functional requirements. My design decisions were on par with how modern web applications are built using separate backend and interactive frontend. The application is very secure against malicious attacks (SQLi and XSS for example) and data breaches against user private information protected by using access tokens. The application could be set behind the company VPN so that unauthorized IP addresses cannot reach the service. Lastly, there are no user identifiable details in the CSV files so in case of data break no private information is compromised. The system is complete with backend, frontend, detailed documentation and a systems manual.

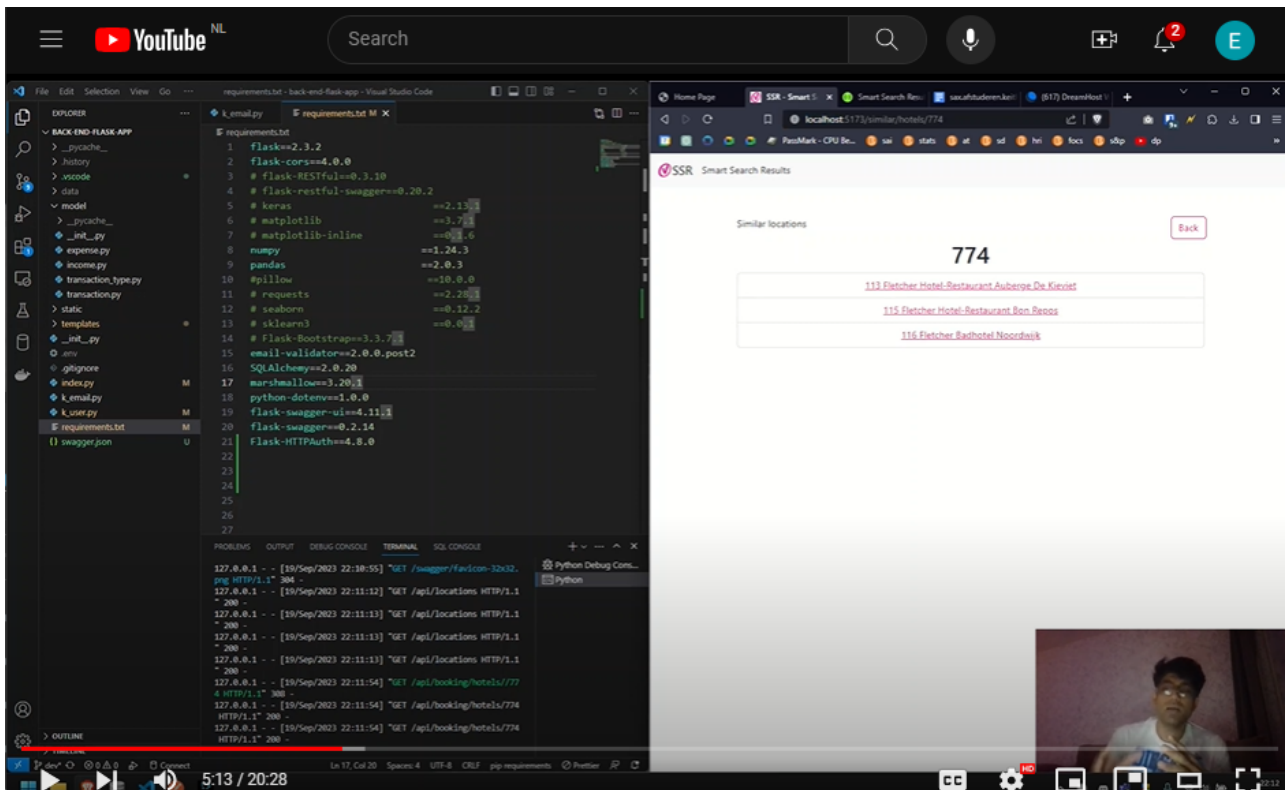


Fig. Intermediary video demo presentation [watch here](#)

Below are screenshots from the frontend admin website where I showcase how the system in the end looks like. For live demo simply visit <https://api-vdu.mywebartist.eu>

7.7. Recommendation

My recommendation to the company would be as following:

- To implement a user login system. This way there is complete browsing history available to the company about a particular user no matter from where or which device they log in to look for their next holiday destination.
- Their database is huge and each table has millions of records. It takes very long to export data, so they need to find a way to export this data faster or without overloading the database. There are techniques such as batch processing or running queries using low priority so that a large dataset could be exported without burdening the live database. In order to overcome this I was able to have two different csv files exported then merge them together before the model could be developed. Obviously this middle step could be removed if the csv files were exported directly in the format required, reducing errors and extra work.
- Try out multiple different models by training various models based on different ideas. Via the Admin website the staff could then see which models are working and try to understand why and repeat the process to optimize search results even further. Administrators will have the flexibility to activate or deactivate specific models, experiment with different recommendation approaches, and keep the system up-to-date by updating or retraining the model with new data.

- Use the booking page as a hint of having great interest in liking the location(hotel), this could be used as one of the attributes taken into account when creating the recommendation model.
- Lastly, to use my project as soon as possible and build upon it. There is an A.I race in every area and the ones who start first will stay ahead.

7.8. Conclusion

In bringing the SSR system to a working stage was a long winding journey which has finally come to an end. This graduation project is an ambitious one for any student. Countless hours of dedication and hard work was needed to get everything to come together.

This journey of the last four years was nothing short of a hopeless experience. Working late nights and starting early. I played the role of a student, a developer, a problem-solver while trying to live and survive.

Many times I wanted to quit the program for different reasons but the people around me wanted me to keep going. Thankfully this is the end so that I can start anew on my next journey. As a student I was self responsible to solve all problems. With this experience I learned that learning comes from determination and a commitment to continue on which is what I am built for.

Ability to conclude this program with this project was an achievement. A milestone of more harder things to come my way. It's a reminder for me that one step at a time will take me to my destination which I sought out for. It's about the passion to pursue knowledge, the courage to take risks, and the resilience to overcome hurdles. As I close this chapter, I carry with me not just a completed project but a newfound confidence and a deeper appreciation for the world of software development.

7.9. Reflection

The final graduation project is nothing short of a long painstaking process. Looking back at it from the start I'm glad to have found Voordeeluitjes.nl to work with. Their team is a pack of herd which moves together as one. Their warm welcome everytime pushed me to get the best of me and I thoroughly enjoyed working next to them. Despite having done numerous software development projects, this one was unique and I practiced my existing skills and learned new things. More projects means more experience and with each new project I become faster and more technically savvy. With this project I also fulfilled my long awaited desire of working with Reactjs.

Time management is a big one because it's not like one cannot achieve anything like this. What matters is how long one takes to get it done and there is only a limited amount of time. On the flip side software development is something very time consuming so the two are always against each other. For someone keen as myself I'm interested to learn how something works or doesn't and when digging into reasons to find out it takes even longer. I would have continued working with this project even longer however there are set dates which I must adhere to and development of features can never end. That in a shell a black of a hole software development is.

Finally, I would like to thank the team(see Personal Reflection section 8.1) at Voordeeluitjes.nl for letting me help them move one step closer to bringing new features to their already amazing system.

8. Video Demo Presentation

8.1. Draft

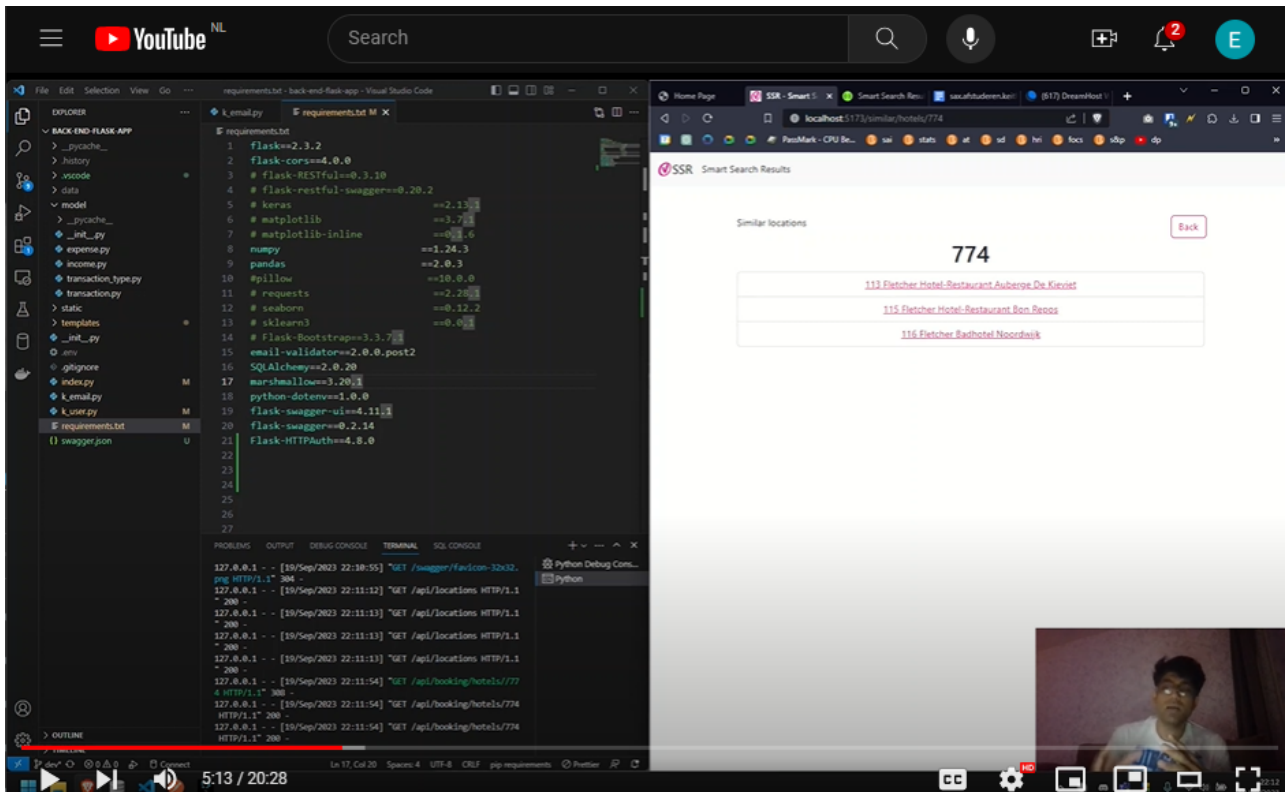


Fig. Intermediary video demo presentation [watch here](#)

9. Appendix

9.1. Graduation proposal HBO-ICT

The format for the graduation proposal to be submitted by you is described below.

Fill in all requested data and describe the proposal in your own words and as clearly as possible. You can then submit the graduation proposal via Blackboard.

Graduation proposal of Keith I. at Voordeeluitjes.nl

Student	Keith I. , 487130, I don't have phone
Degree program	HBO-ICT, profile <SE> (strike through or delete what does not apply)
Versie	1.0
Date	28 Mar 2023
Intended starting moment	<input type="checkbox"/> Week 1 of quarter 1 <input type="checkbox"/> Week 1 of quarter 2 <input type="checkbox"/> Week 1 of quarter 3 <input checked="" type="checkbox"/> Week 1 of quarter 4
Admission requirements	Indicate your situation at the <u>date of submission</u> , and clearly indicate when you intend to complete the modules that have not yet been completed: <input checked="" type="checkbox"/> I meet the transfer requirements. I have completed my propaedeutic phase + at least 142 EC in the main phase. <input type="checkbox"/> I still do not meet the transfer requirements, the following modules have yet to be completed:
Statement	By submitting this graduation proposal, I declare that I: <ul style="list-style-type: none">• am thoroughly familiar with the graduation manual, which can be found on Blackboard• have handed the graduation manual to the prospective company supervisor and have pointed out to him / her the requirements for the company and company supervisor as stated in the graduation manual• this graduation proposal was discussed with the company supervisor by date: 28 Mar 2023, 3 Apr 2023, 14 Apr 2023, 25 Apr 2023

The graduation proposal contains at least the following information:

1. The company

a) Freetime Company B.V, Voordeeluitjes.nl

Munsterstraat 20, 8418 EV Deventer

+31(0) 88 130 4420

They are a holiday website which offers booking for holidays in the Netherlands and Germany.

110 people from which 40 work in the IT department.

b) Development supervisor:

Contact person 1:

Arno Schlepers, IT Manager

Bachelor of Communication, IDM(Informatiedienstverlening en management) at Saxion in 2008

20+ years of working experience in IT

a.schlepers@freetimecompany.nl

Munsterstraat 20, 8418 EV Deventer

LinkedIn <https://www.linkedin.com/in/arnoschlepers/>

Contact person 2:

Rasha Medhat

Senior software engineer

Bachelor of Engineering at Cairo University in 2007

15+ years of working experience in Computer Programming

r.medhat@freetimecompany.nl

Bouwerij 4 h, 1185 XX Amstelveen

LinkedIn <https://www.linkedin.com/in/rasha-medhat-6748238b/>

c) The student will be provided with:

Company systems

Access website system

Access to company locations either in Amstelveen or Deventer

Company data which is based on its existing customers. The dataset includes

User details(user id, address, etc)

User's purchase history(places booked, booking dates etc)

User's rating on visited destinations

User's browsing interaction with website(places searched,

visited hotel pages, pages clicked, favorites destination/hotels)

Hotel details(name, location, features)

Hotel packages/deals/price information

Other data as necessary

The company mentor has informed me that they have necessary data for ML

2. The graduation project

Starting situation: They want to improve their search results for their service to show better search results when users are searching for holiday destinations. Currently their system makes recommendations of search results based on deterministic queries on SQL. These search results offer matching results based on keywords. The company wants to improve their search results by showing holiday destinations/hotels based on users interaction history to provide more relevant holiday destination results to make their offers more appealing.

a) Problem analysis: The search results for particular hotels are arranged in order based on a few criteria such as price, popularity, ratings. Upon users choice these results can be arranged accordingly. However, when sorted upon popularity, the results would simply show hotels first which were either most clicked or

most booked and similarly if sorted using rating the results are sorted based rating high to low or vice versa. These options may end up showing places or hotels which may be of no interest to users as they are not based on the user's own likings.

It's obvious that users are more likely to book holidays at destinations they are most interested in going to. For this reason the company wants to show results which appeal to the users most. In order to create more relevant hotel search results and destination recommendations to users, one approach to solving this problem is machine learning. With ML it may be possible to discover the relationship between a user's booking habits by seeing their past booking history, current website interaction and then try finding the user's personal interests which caused them to book particular hotels or destinations. There may be other users who may have the same interests which could also be appealing to other users. These users would be most likely to book similar hotels again which matches users personal interests and with machine learning relationships in the given dataset can be identified.

b) Problem definition: Search results shown to users of hotels are too generic. Existing sorting and hotel recommendation system shows results based only on keyword/location matching and it misses out on similar places which might be appealing to the users based on hotel/location similarities.

Goal: The users will have a new option to sort results by "personalized recommendation" along with other existing sorting functions(price, rating, popularity). This option will show a list of hotels/destinations which will be generated by the algorithm of machine learning based on the user's interests and personal preferences. The company collects data of the users and their interacting history.

c) Main research question: What ways hotel search results could be made more relevant to users.

d) Additional research questions:

I.Question: What data must be collect data from visitors of website, which could be used to provide personalized recommendations based on user's personal preference

Explanation: Methods of data collection, storage and processing. The basis of machine learning is relevant data. Need to find out what data is already collected and what other data would be useful in building a better recommendation algorithm

II.Question: How can this recommendation system be implemented into an existing website

Explanation: The recommendation system is a machine learning model created upon data. For this model to be used it needs to be implemented inside the company's website system.

III.Question: How can administrators see progress reports and do customization to the ML model.

Explanation: The website system administrators want to be able to see how the ML model is performing. They also want to be able to override the ML model by setting custom settings which would modify the search results giving a combination of recommendations plus the overrides. There should be some kind of secure web interface where these reports can be seen and settings modified.

e) Preconditions: The data provided by the company is relevant and appropriate for applying machine learning. Give enough details and access to the system to build the addon system.

3. Graduation worthiness and HBO-i domain description

When your graduation proposal is assessed, it is important that the graduation proposal is graduation worthy. After all, we do not want you to start with a graduation project that you cannot graduate with.

Explanation:

Our current graduate profiles of HBO-ICT have defined the intended final level with the help of the so-called HBO-i 25-squares model. This model has three dimensions: activities (what does an IT worker do?), architectural layers (in what context?) and mastery levels (how complex?).

According to the graduation requirements, you must demonstrate at least three activities at mastery level 3 during graduation, of which at least two are activities within the architecture layer that matches your graduation profile (architectural layer 'Organizational processes' at Business & IT, architectural layer 'Infrastructure' at Infrastructure and architectural layer 'Software' at graduation profile Software Engineering). It goes without saying that all three activities at level 3 may also be in the architectural layer that matches your graduate profile.

You will find extensive information about this domain description on the HBO-i website (see <https://www.hbo-i.nl/publicaties-domeininformatie/>). In particular, we refer you to this domain description for **exemplary tasks**. In addition, you will also find **videos** here that can provide more insight into the activities (<https://www.hbo-i.nl/domeininformatie-videos/>). For the time being, these videos have only been made for the implementation of the organizational processes architectural layer and only in Dutch.

To gain insight into the graduation worthiness of your graduation project, we ask you to describe the following:

- a) Which of the activities at mastery level 3 associated with your graduation profile do you think you can demonstrate? Indicate this with an "x" in the diagram of the HBO-i 25-squares model.

		activities				
		manage/control	analyze	advise	design	realize
Ar chi tec tur al lay ers	user interaction					
	organizational processes					
	Infrastructure					
	software		X		X	X
	hardware interfacing					

Figure 1: HBO-i 25-squares model 2018

- b) Show in the overview table below whether you can demonstrate the graduation level and how you will do this. You do this by establishing a relationship between the activity (of which you have entered at least 3 in figure 1), one or more research questions and the resulting products. You should place the products in the context of your graduation project as much as possible (see the description at 2d in this graduation proposal).

Problem statement:		
f) Search results shown to user of hotels are too generic		
g) Main research question: What ways hotel search results could be made more relevant to users.		
Objective: Below is a list of products to be delivered. Some will have more weight than others. The main emphasis is on the main research question but there are also other parts which would complete the feature the company is looking to build.		
activity at level 3:	Research questions:	product(s) + explanation:
Architectural layer: software activity: analyze	What data must be collected from visitors of website, which could be used to provide personalized recommendations based on user's personal preference	Product: Research report Methods of data collection, storage and processing. The basis of machine learning is relevant data. Need to find out what data is already collected and what other data would be useful in building a better recommendation algorithm. The company already has a lot of user interaction data such as user clicks, purchase history, liked destinations, liked hotels, favorited pages etc. The mentor has informed me that they should have enough user insight data in order to build a ML model out of, however, since this is the first time they will be creating an ML model there must be some other data points which could have been collected or stored in a different way in order to facilitate ML model production process in future
Architectural layer: software activity: design	h) What ways hotel search results could be made more relevant to users	Product: Machine learning model Design a machine learning model which takes in information from users and tries to match with destinations. The hotels which best fit users personal preferences, giving recommendations as to which places the user would enjoy the most.
Architectural layer: software activity: realize	How can this recommendation system be implemented into an existing website	Product: Model implemented in the website The recommendation system is a machine learning model created upon data. For this model to be used

		it needs to be implemented inside the companies website system
Architectural layer: software activity: realize	How administrators can see reports and customize the ML model	<p>Product: Recommender admin interface</p> <p>The admin interface could be in the form of a website. The website must be secure and give access to only authorized users. The purpose of the website is to let admins manage the Machine Learning model and see various reports. One of the settings should be to be able to set/modify custom properties(such as boost) for a particular hotel to show up upon certain searched keywords. This and other ML model overrides should be possible which can be activated/deactivated for a given timeframe.</p> <p>The website should provide real time insights about</p> <ul style="list-style-type: none"> - Model output - User segments - User profiles - Conversion

4. Motivation

The company operates in the tourism sector. Their customers are booking holidays in the Netherlands and Germany. They have a network of hotels for which they advertise for making a commission on the bookings they bring for the hotels.

The company tries to make their bookings attractive by showing details about the destinations and finding lowest prices to book. The company can significantly increase their bookings by showing more relevant search results for their users.

This can potentially lead to growth of the company as they would have a competitive advantage because users can easily find destinations and hotels according to their interests and they would want to book again.

The company's idea about building this recommendation system is at very good timing as machine learning is applicable to almost any field today. I would be very interested to work on this assignment as I will get to think about the problem and try to find ways to come up with such a system and get to implement it.

The project size is of the right size and it's something I want to build more experience in. My mentor has a lot of experience at the company and I'm looking forward to adding this feature to the website.

9.2. Research Questions

There are 4 research questions in my assignment. They are as follows:

9.2.1. RQ1: (Recommendations)

What ways hotel search results could be made more relevant to users.

9.2.2. RQ2: (Data)

What data must be collect data from visitors of website, which could be used to provide personalized recommendations based on user's personal preference

Explanation: Methods of data collection, storage and processing. The basis of machine learning is relevant data. Need to find out what data is already collected and what other data would be useful in building a better recommendation algorithm

9.2.3. RQ3: (Implementation)

How can this recommendation system be implemented into an existing website

Explanation: The recommendation system is a machine learning model created upon data. For this model to be used it needs to be implemented inside the company's website system.

9.2.4. RQ4: (Interface)

How can administrators see progress reports and do customization to the ML model.

Explanation: The website system administrators want to be able to see how the ML model is performing. They also want to be able to override the ML model by setting custom settings which would modify the search results giving a combination of recommendations plus the overrides. There should be some kind of secure web interface where these reports can be seen and settings modified.

9.3. Contacts

Details of persons involved in the project.

Role	Name	Email
Student	Keith I.	487130@student.saxion.nl
IT manager	Ano Schlepers	a.schlepers@freetimecompany.nl
Senior software engineer	Rasha Medhat	r.medhat@freetimecompany.nl
Team Lead Software Development	Mustapha Idrissi	m.idrissi@freetimecompany.nl
Data Analyst	Shilpa Sasidharan Nair	s.sasidharannair@freetimecompany.nl
College contact	Michael De Louwere	m.h.e.delouwere@saxion.nl

9.4. Glossary

Some useful terminologies.

ML Machine learning	A branch of artificial intelligence that enables systems to learn from data and improve their performance without explicit programming
Machine learning model	A mathematical representation of a real-world process that is learned from data and can make predictions or decisions based on new data.

Collaborative filtering	A technique for recommendation systems that uses the ratings or feedback of other users who have similar preferences to generate recommendations for a given user.
Content-based filtering	A technique for recommendation systems that uses the features or attributes of the products or services to generate recommendations for a given user based on their profile or preferences.
VDU	Voordeeluitjes.nl website
SSR system	Smart Search Result system
RQ	Research Question

9.5. Readme.md

9.5.1. Main

This Readme.md file is in the root folder of the VDU project.

```
# Smart Search Results(SSR)
by Keith I.
```

```
## Description
Hotel recommendation system.
```

There are two folders in this project.

- back-end-flask-app
- front-end-react-app

As the folder names suggest the back end is built using flask which runs on Python and React which runs on Node.

These two systems are independent so they could be replaced without affecting the other.

Installation

To download projects either on server or locally run this command. This repo is set as private on Gitlab. VDU has been given a copy to use. If you are from VDU you may request access for <https://gitlab.com/mywebartist/vdu> by Keith. See contact details below.

```
~~~
git clone https://gitlab.com/mywebartist/vdu.git
~~~
```

Support

Keith at contact@mywebartist.eu
Data or programming team at VDU.

Roadmap

This is the first attempt for a hotel recommendation system for VDU. They may choose to build upon this and implement it on their website.

Authors and acknowledgment

Lead programmer: Keith I. (487130)

License

To be used by voordeeluitjes.nl

Project status

Completed

9.5.2. Backend setup

This Readme.md file is in the backend folder of the VDU project.

Smart Search Results(SSR) - Backend

by Keith I.

Description

Hotel recommendation system.

This is the main API system is the backend system. The endpoints produced from there could be accessed by anywhere with authorized access.

Setup

Requirement

- Python 3.x

Data files

In order for the recommendation model to work. You first need to enter the data files and generate the model. Without this step there will not be any recommendations or information returned by the server as the whole data would be missing. The data folder is in /data. you may upload csv files. Here is list of files which you need:

- csv_bookings.csv (generated)
- csv_bookings_no_ratings.csv *(*important*)*

```
- csv_locations.csv *(important*)
- csv_ratings.csv
- table_cities.csv
- table_countries.csv
- table_locations.csv *(important*)
- table_provinces.csv
- table_regions.csv
```

**(You don't need all the CSV files to get started as the most important csvs in this are table_locations.csv, bookings_no_ratings.csv, and csv_bookings_no_ratings.csv*)*

Setup environment file

Copy and rename sample_env as .env file. This environment file contains system settings which differ depending on whether you want to run the system on a local development or production server. You need to have another .env file with settings for the production server if you are going to set up the production server.

Installation

Before index.py can run you need to have a Python environment setup. This is not required but highly recommended as the installation can break should you run other python programs in your default environment. You only need to set this up once. Here is an example for setting up a Python environment.

```
```
```

```
python -m venv env_ssr
```

```
```
```

You will see a bunch of files being created in the env_ssr folder. Inside env_ssr/Scripts/python.exe executable. You will need to use this to run index.py but before that you need to install a list of dependent library files. These libraries are listed in requirements.txt. To install, run this command.

```
```
```

```
env_ssr/Scripts/pip install -r requirements.txt
```

```
```
```

Start Flask server using this command. This should launch Flask server at its default address <http://localhost:3006>

```
```
```

```
env_ssr/Scripts/python index.py
```

```
```
```

Setup Users

In the environment file you will see these keys to set up the first admin. This is very important to note as this will be the first admin which cannot be deleted or admin privileges revoked. Consider this as the master admin user. The only way to modify this user is to edit the .env file. See env.sample.txt but here you can do what you need to enter.

- FIRST_ADMIN_TOKEN=strong_token
- FIRST_ADMIN_EMAIL=email@email.com
- FIRST_ADMIN_NAME=name

You can generate strong token by sending API request at http://localhost:3006/api/generate_token

Generate model

Having the csv files entered in the data folder. It's time to generate a model. You need to send these two commands in this sequence.

1 <http://localhost:3006/api/model/compile-bookings-csv>

2 <http://localhost:3006/api/model/generate-booking-model>

Do note that the API call will instantly return a success message if nothing has gone wrong. This does not mean that the files are done generating as this is a background task. The speed of completion depends on the speed of the system and size of the data(csv files).

API Documentation

Every endpoint is well documented using the Swagger system. You can see the link at the homepage of API server or directly visit <http://localhost:3006/swagger> A json version of API documentation is published here <http://localhost:3006/apidocs>

Support

Keith at contact@mywebartist.eu

Data or programming team at VDU.

License

To be used by voordeeluitjes.nl

9.5.3. Frontend setup

This Readme.md file is in the frontend folder of the VDU project.

```
# Smart Search Results(SSR) - Frontend  
by Keith I.
```

Description

Hotel recommendation system.

This is a frontend for the API system. The purpose of this website is to have methods for users such as staff working in the sales team as a way to see how the model is performing for particular hotels(locations). They make adjustments to the recommendation model by overriding match rates. To use this you must have an access token.

Setup

Requirement

- Node >= 20.x.x

Setup environment file

Copy and rename sample_env as .env file. This environment file contains system settings which differ depending on whether you want to run the system on a local development or production server. You need to have another .env file with settings for the production server if you are going to set up the production server. You can have two env files .env for local development and .env.production which will be used automatically on 'npm run build' command see below.

Installation

From terminal/command prompt run this command

```
^^^
```

```
npm install
```

```
^^^
```

Build for local development

This should launch Node server at its default address <http://localhost:5176>

```
^^^
```

```
npm run dev
```

```
^^^
```

Build for server


```
This will output dist files in
'/home/user_k/web/vdu.mywebartist.eu/public_html' which is the demo server.
You may edit this output directory in vite.config.ts file as per your
production server directory hierarchy. This will use the .env.production
file.
```

```
```
npm run build
```
```

```
In case you want to build using a .env file for development then run this.
```
```

```
npm run build:dev
```
```

Support

Keith at contact@mywebartist.eu
Data or programming team at VDU.

License

To be used by voordeeluitjes.nl

9.6. SQL Queries

Bookings.sql v1.1

```
SELECT b.id as booking_id,
       v.cookie_id,
       v.remote_addr_number as ip,
       MD5(email) AS email_hash,
       l.id as location_id,
       b.create_date as booking_date,
       b.state as booking_state,
       b.action_id as action_id,
       p.id as package_id,
       p.amount_days as days,
       r.score as rating
FROM bookings b
     INNER JOIN guests g on g.id = b.guest_id
     INNER JOIN packages p ON p.id = b.package_id
     INNER JOIN locations l ON b.location_id = l.id
     inner join visits_bookings vb on b.id = vb.booking_id
     inner join visits v on vb.visit_id = v.id
     LEFT JOIN www_reviews.review r ON b.id = r.BookingId
```

```
where b.state in(1, 2, 7, 8, 10)
      and create_date >= DATE(NOW()) - INTERVAL 7 DAY)
group by b.id
```

Bookings.sql v1.2

```
SELECT b.id as booking_id,
       g.MD5(email) AS email_hash,
       l.id as location_id,
       l.name as location_name,
       b.create_date as booking_date,
       r.score as rating
FROM bookings b
      left JOIN guests g on g.id = b.guest_id
      left JOIN locations l ON b.location_id = l.id
      LEFT JOIN www_reviews.review r ON b.id = r.BookingId
where b.state in(1, 2, 7, 8, 10)
      and create_date >= DATE(NOW()) - INTERVAL 999 DAY)
group by b.id
```

Bookings.sql v1.3

```
SELECT b.id as booking_id,
       g.MD5(email) AS email_hash,
       l.id as location_id,
       l.name as location_name,
       b.create_date as booking_date,
FROM bookings b
      left JOIN guests g on g.id = b.guest_id
      left JOIN locations l ON b.location_id = l.id
where b.state in(1, 2, 7, 8, 10)
      and create_date >= DATE(NOW()) - INTERVAL 999 DAY)
group by b.id
```

Locations.sql v1.1

```
SELECT l.id as location_id,
       l.name as location_name,
       c.id as city_id,
       pr.id as province_id,
       cn.id as country_id,
```

```
l.review_average as average_rating,  
group_concat(distinct usp_facilities.id) as facility_ids,  
group_concat(distinct usp_facilities.name) as facility_names  
FROM www_ftc.locations l  
    inner join locations_g7_facilities loc_fac on l.id = loc_fac.location_id  
    inner join locations_usp_facilities on loc_fac.location_id =  
locations_usp_facilities.location_id  
    inner join usp_facilities on locations_usp_facilities.usp_facility_id =  
usp_facilities.id  
    inner join cities c ON l.city_id = c.id  
    inner join provinces pr ON c.province_id = pr.id  
    inner join countries cn ON pr.country_id = cn.id  
group by l.id
```

Ratings.csv v1.0

```
SELECT b.id as booking_id,  
    b.create_date as booking_date,  
    r.score as rating  
FROM bookings b  
    LEFT JOIN www_reviews.review r ON b.id = r.BookingId  
where b.state in(1, 2, 7, 8, 10)  
    and create_date >= DATE(NOW()) - INTERVAL 999 DAY)
```

Table_locations.sql v1.0

```
SELECT l.id as location_id,  
    l.name as location_name,  
    l.country_id,  
    l.city_id,  
    Concat(p.id, '_', p.name) as picture_url  
FROM www_ftc.locations l  
    left join locations_photos lp on lp.location_id = l.id  
    left join photos p on p.id = lp.photo_id  
group by l.id
```

Visits.csv v1.0

```
SELECT v.time as Date,  
    v.id as Visit_id,  
    v.cookie_id,
```

```
v.remote_addr_number,  
v.targeturl_id,  
t.location_id  
FROM www_ftc.visits v  
    inner join targeturls t on v.targeturl_id = t.id  
    and t.location_id IS NOT NULL  
where date(v.time) >= DATE(NOW()) - INTERVAL 7 DAY)
```

Visits.csv v1.1

```
SELECT v.id as visit_id,  
    v.cookie_id,  
    v.remote_addr_number as ip,  
    t.location_id,  
    v.time as visit_date,  
    v.targeturl_id,  
    t.action_id  
FROM www_ftc.visits v  
    inner join targeturls t on v.targeturl_id = t.id  
    and t.location_id IS NOT NULL  
where date(v.time) >= DATE(NOW()) - INTERVAL 7 DAY)
```

Visits.sql v1.2

```
SELECT v.id as visit_id,  
    v.cookie_id,  
    v.remote_addr_number as ip,  
    t.location_id,  
    v.time as visit_date,  
    v.targeturl_id,  
    t.url,  
    t.action_id  
FROM www_ftc.visits v  
    inner join targeturls t on v.targeturl_id = t.id  
    and t.location_id IS NOT NULL  
where date(v.time) >= DATE(NOW()) - INTERVAL 7 DAY)
```

Bookings.sql v1.3

```
SELECT b.id as booking_id,  
    g.MD5(email) AS email_hash,
```

```
l.id as location_id,  
l.name as location_name,  
b.create_date as booking_date,  
FROM bookings b  
left JOIN guests g on g.id = b.guest_id  
left JOIN locations l ON b.location_id = l.id  
where b.state in(1, 2, 7, 8, 10)  
and create_date >= DATE(NOW() - INTERVAL 999 DAY)  
group by b.id
```

9.7. CSV files format

This section gives details about each CSV file format and data contained therein.

csv_bookings.csv

booking_id	email_hash	location_id	location_name	booking_date	rating
1461055	70344c27cd6f2170f33cfba3572494c5	4750	NH Atlantic Den Haag	13/01/2021 13:57	5.8
1461078	bd1c5948e9d4f65f1281f75f7372acdd	809	De Wapser Herberg	13/01/2021 17:34	9
1461090	54c06873ae62cd2d52a739a5039be644	1008	Hotel Restaurant Vijlerhof	13/01/2021 18:55	9.6
1461154	3a4e03d78657703b631a094cd641e026	2461	Golden Tulip Hotel Central	14/01/2021 14:34	9.5
1461165	701951265531b8bcc72f533a9d2a6416	1008	Hotel Restaurant Vijlerhof	13/01/2021 13:57	9.5

csv_bookings_no_ratings.csv

booking_id	email_hash	location_id	location_name
1460733	7965cd0899b29a517f7eb56fe7b63a09	809	De Wapser Herberg

1460736	3a31ce039d23cd1c1573671c62325a9c	4749	NH Conference Centre Leeuwenhorst
1460737	ddd695c45bba049df1364fe1fb8d5539	1524	Hotel De Hoeve van Nunspeet
1460741	0e6ecaf902d7034ab8cd871d2e345f1c	1154	Bilderberg Europa Hotel Scheveningen
1460742	7470ee5655f8703bcbe1135bad27ba3f	147	Boetiek Hotel BonAparte

csv_locations.csv

location_id	location_name	city_id	province_id	country_id	average_rating	facility_id	offered_facilities
113	Fletcher Hotel-Restaurant Auberge De Kieviet	2	12	1	6.6	17,18,48,49,245,312,429,433,434,437,469,612,1182,3989	Bar,Barbecue,Fietsenstalling,Fietsverhuur,Huisdieren toegestaan (op aanvraag, tegen betaling â,¬ 15 p.h.p.n.),Lift,Lounge,Oplaadmogelijkheid elektrische fiets,Restaurant,Rolstoelverriendelijk,Rookvrij hotel,Roomservice,Terras ,Wasserij/stomerij service
115	Fletcher Hotel-Restaurant Bon Repos	4	6	1	5	17,18,29,30,31,49,50,433,605,612,1123,1614,1726,3512,3988	29 Hotelkamers,Biljart,Draadloos internet (gratis),Elektrische fietshuur,Fietsenstalling, Huisdieren toegestaan (op aanvraag, tegen betaling â,¬ 10 p.h.p.n.),Lift,Mindervalidenkamers niet

							aanwezig, Restaurant, Rols toeliefdevriendelijk, Rookvrije kamers, Tafeltennistafel, T erras, Tuin, Zonbank
116	Fletcher Badhotel Noordwijk	5	12	1	6.2	18,22,3 7,49,42 8,431,4 34,469, 1123,1 182,11 88,161 4,2915, 3988	Draadloos internet (gratis), Fietsenstalling (overdekt), Fietsverhuur, H uisdieren toegestaan (op aanvraag, tegen betaling â, ¬ 10 p.h.p.n.), Internet, Kamers begane grond, Lounge, Mindervali denkamers niet aanwezig, Oplaadmogelij kheid elektrische fiets, Parkeergelegenheid (tegen betaling van â, ¬10 per nacht), Restaurant, Sauna met dompelbad, Solarium, Terr as
124	Fletcher Hotel-Restauran t De Grote Zwaan	9	9	1	5.9	33,48,4 9,428,4 29,433, 434,43 7,469,1 182,11 95,131 2,1397, 3512,3 988	36 Hotelkamers, Bar, Barbecu e, Elektrische fietshuur, Fietsenstalling (overdekt/afgesloten), Fiet sverhuur, Geschikt voor groepen, Gratis parkeergelegenheid, Huis dieren toegestaan (op aanvraag, tegen betaling â, ¬ 10 p.h.p.n.), Kamers begane grond, Lounge, Oplaadmo gelijkheid elektrische fiets, Rolstoelvriendelijk, Rookvrij hotel, Terras

125	Fletcher Hotel-Restaurant De Gelderse Poort	10	4	1	6.6	17,18,4 9,167,4 28,429, 433,43 4,469,6 12,118 2,3988	Bar,Dakterras,Fietsenstalling,Fietsverhuur,Huisdieren toegestaan (op aanvraag, tegen betaling â, 10 p.h.p.n.),Kamers begane grond,Lift,Lounge,Oplademogelijkheid elektrische fiets,Restaurant,Rolstoelverriendelijk,Terras
-----	---	----	---	---	-----	--	--

csv_ratings.csv

booking_id	booking_date	rating
1461055	13/01/2021 13:57	5.8
1461078	13/01/2021 17:34	9
1461090	13/01/2021 18:55	9.6
1461154	14/01/2021 14:34	9.5
1461165	14/01/2021 15:47	9.5

table_locations.csv

location_id	location_name	country_id	city_id	picture_url
113	Fletcher Hotel-Restaurant Auberge De Kieviet	1	2	565_15215462ffa27c2c96c36aa6c5124a48.jpg
114	NH Hoteles Kijkduin	1	34	569_2de99dd37727071802443a2652e0173a.jpg

115	Fletcher Hotel-Restaurant Bon Repos	1	4	3698_de92f2ac9e16633fc8b11c2e0367f553.jpg
116	Fletcher Badhotel Noordwijk	1	5	425_0f2dfe1d91cbf323f3215ee2fdac601d.jpg
124	Fletcher Hotel-Restaurant De Grote Zwaan	1	9	456_65c4390d420696a9b2e0ff2ac0a36de3.jpg

9.8. API documentation preview

Smart Search Results API docs ^{1.0}

/apidocs

Users



PUT

/api/admin/user Update user properties such as set status enabled or not etc

put_api_admin_user



GET

/api/admin/user/{email} Get user details

get_api_admin_user_email_



GET

/api/admin/users Get list of users registered with API access

get_api_admin_users



GET

/api/generate_token *Generate random token. Can be used to create new token for first admin

get_api_generate_token



GET

/api/me Get users own details

get_api_me



PUT

/api/me Update user properties such as set status enabled or not etc

put_api_me



POST

/api/user Add new user.

post_api_user



Recommendation				^
GET	/api/booking/hotels/{hotel_id}	Get list of hotels which are most likely to be booked hotels for given hotel	get_api_booking_hotels__hotel_id_	▼
POST	/api/boost	boost particular location	post_api_boost	▼
GET	/api/boosts	get all boosts	get_api_boosts	▼
GET	/api/similar/hotels/{hotel_id}	Get similar hotels for a given hotel @todo	get_api_similar_hotels__hotel_id_	▼
Data				^
GET	/api/booking_states	Get booking states	get_api_booking_states	▼
GET	/api/cities	Get all cities	get_api_cities	▼
GET	/api/countries	Get all countries	get_api_countries	▼
GET	/api/locations	Get all locations	get_api_locations	▼
GET	/api/locations_facilities	Get all locations with their facilities	get_api_locations_facilities	▼
GET	/api/provinces	Get all provinces	get_api_provinces	▼
GET	/api/regions	Get all regions	get_api_regions	▼
Model				^
GET	/api/model/compile-bookings-csv	this endpoint will combine csv_bookings_no_ratings.csv and csv_ratings.csv and output csv_bookings.csv which is used by /booking/hotels/<int:hotel_id> endpoint. this was necessary because exporting the csv_bookings with rating was taking too long.	get_api_model_compile_bookings_csv	▼
GET	/api/model/generate-booking-model	this endpoint will use the csv_bookings.csv to generate model files and save them in the model folder. this must be run anytime the csv files are updated to update the model. the model building takes few minutes.	get_api_model_generate_booking_model	▼
GET	/api/model/hit-rate	this endpoint will show you the Hit Rate. this is metric used to evaluation the recommendation to measure how often user has shown interest in booking.	get_api_model_hit_rate	▼
				VALID {...}

9.9. Concept design

The company provided me with their existing plans of building this system. Their idea was to generate the recommendations depending on different metrics and not machine learning. Their metrics included things such as most visited hotels, highest booked hotels etc.

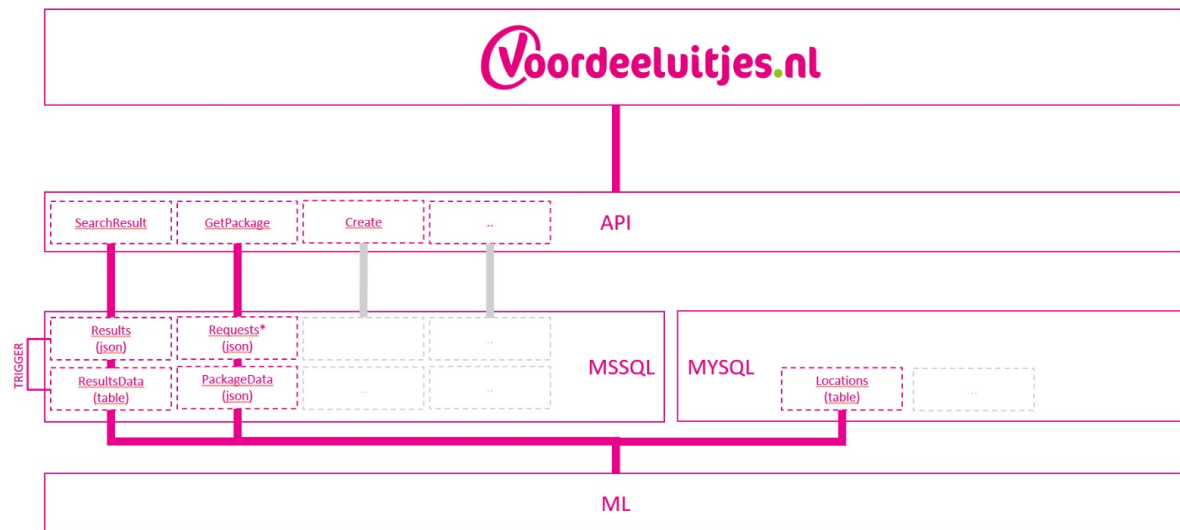


Fig. Possible system design

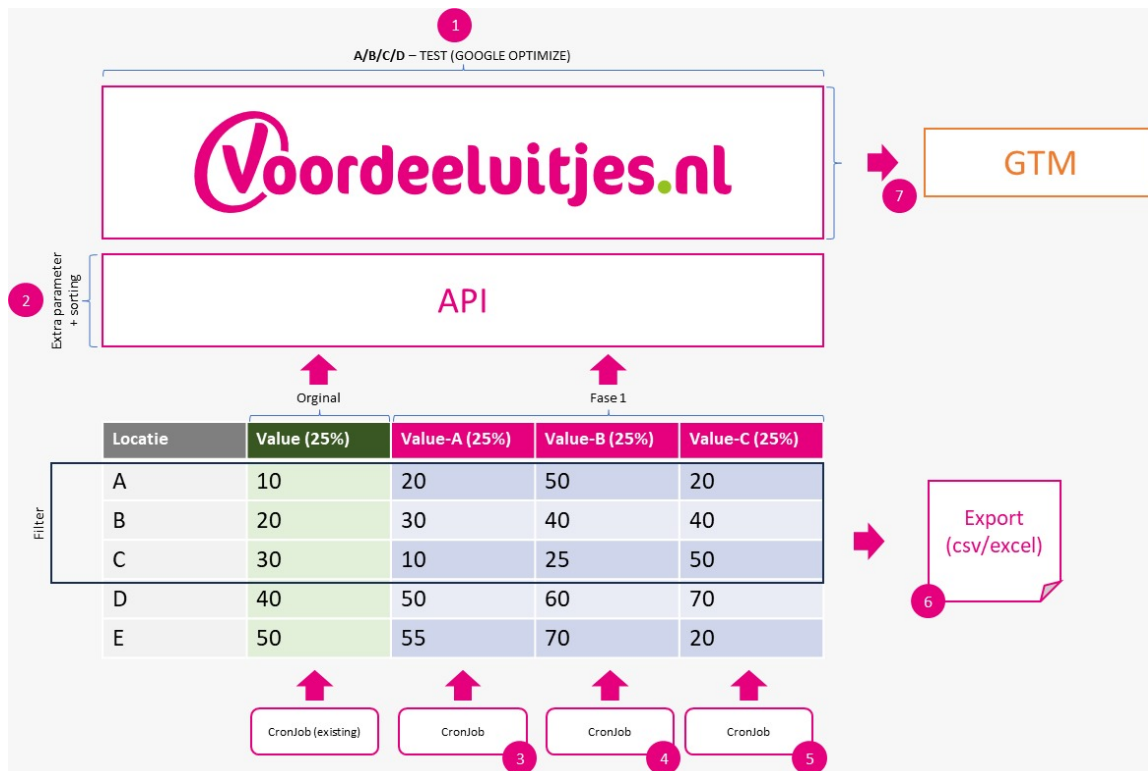


Fig. A/B/C/D conversion testing

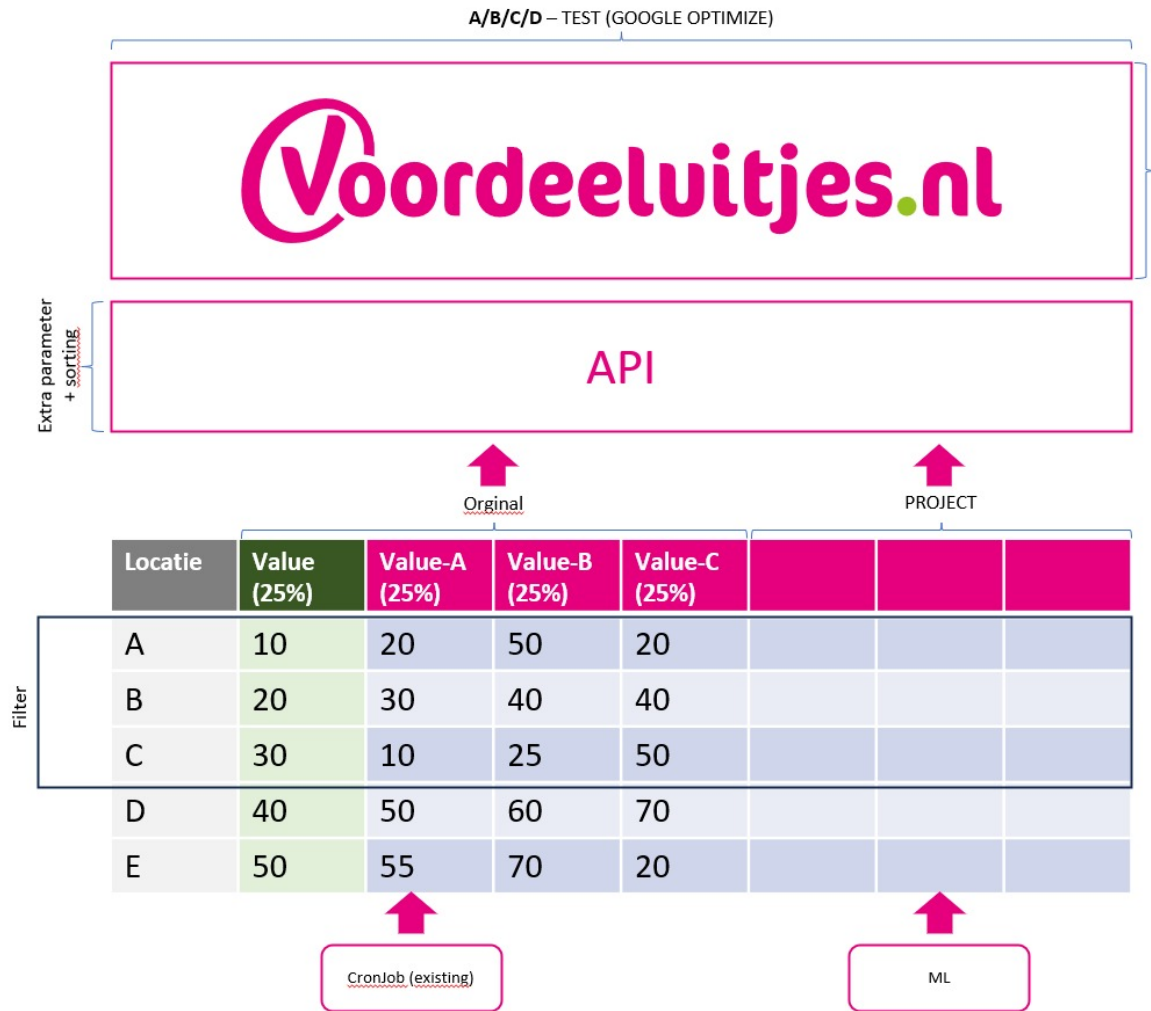


Fig. A/B/C/D conversion testing

9.10. Gitlab commit log

Gitlog contains 187 commits <https://drive.google.com/file/d/18srsVezDvVhkbdY143b8wY1If166XAYC>

9.11. Graphs & Diagrams

Graphs and diagrams were made using draw.io This file is stored in the Gitlab repo as drawings.drawio <https://app.diagrams.net/#Amywebartist%2Fvdu%2Fmain%2Fdiagrams.drawio>

9.12. Screenshots

Smart Search Results

Backend API

[API Docs](#) | [Admin Website](#)

2 admins and 1 users

Fig. Backend API homepage <https://api-vdu.mywebartist.eu>

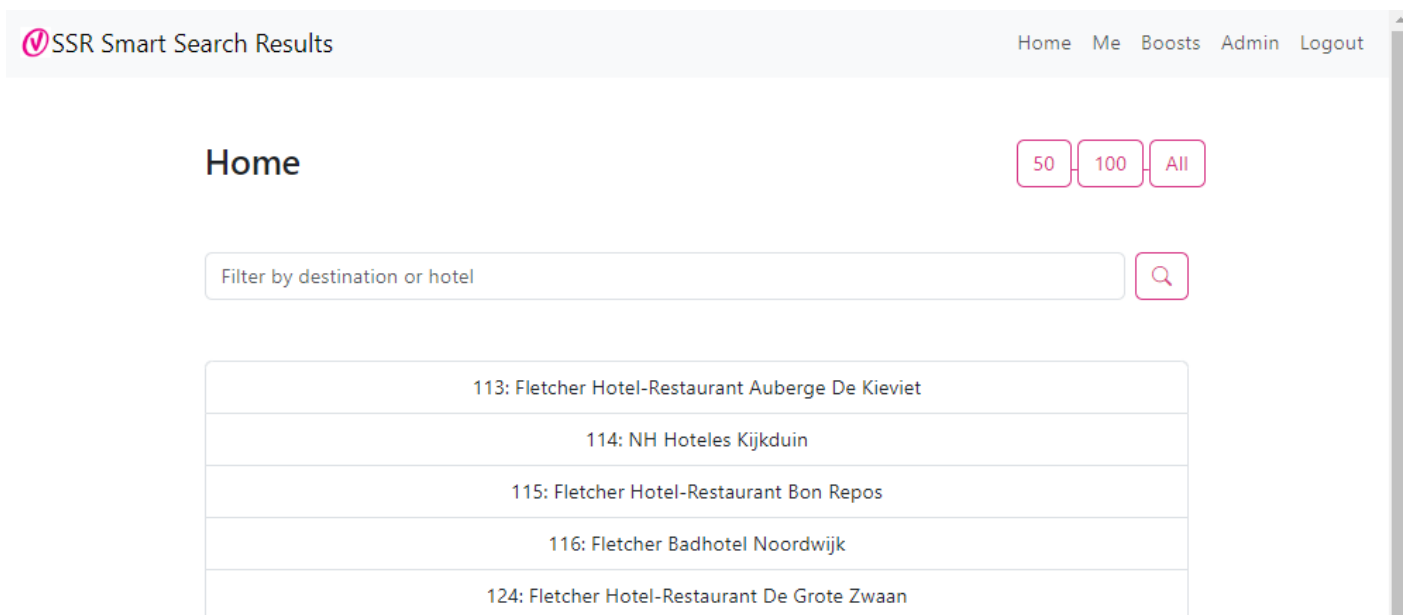


Fig. Frontend homepage <https://vdu.mywebartist.eu>

Booking Recommendations

Back

Hotel Wapen van Delden

Id:144



(100.0% model) | (112.0% boosted)

144: Hotel Wapen van Delden

29 Oct ▼ 2023 04:19 – 05 Nov ▼ 2023 04:19 ✕ 📅

Boost 12%



(82.1% model) | (82.1% boosted)

3225: Havezathe Carpe Diem

29 Oct ▼ 2023 04:19 – 05 Nov ▼ 2023 04:19 ✕ 📅

Fig. Frontend recommendations page <https://vdu.mywebartist.eu/booking/hotels/144>

Me

[Back](#)

Name	<input type="text" value="tt5fg"/>
Email	<input type="text" value="admin@hotmail.com"/>
Status	<input type="text" value="true"/>

[Update](#)

Fig. Backend API homepage <https://vdu.mywebartist.eu/me>


✓ SSR Smart Search Results

Home Me Boosts Admin Logout


Boosts

Back

5900:
4% boost



4496: Mercure Antwerp City Centre
16% boost



795: De Oringer Marke & Stee Hotels by Flow
20% boost

Fig. Frontend boosted locations page <https://vdu.mywebartist.eu/boosts>

✓ SSR Smart Search Results

Home Me Boosts Admin Logout


Admin

Back

1 Compile CSVs 2 Update Booking Model 3 Hit Rate

Id	Name	Email	Status	Admin
10	tt	user@hotmail.com	<input checked="" type="checkbox"/>	<input type="checkbox"/>
11	tt5fg	admin@hotmail.com	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
0	K	contact@mywebartist.eu	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Fig. Frontend admin page <https://vdu.mywebartist.eu/admin>

 SSR Smart Search Results Home Login

Register

Name

Email

Register

Login

Token

Save access token

Fig. Frontend register/login page <https://vdu.mywebartist.eu/login>

9.13. References

These reading materials have been identified to be useful resources when carrying out the assignment.

Cover page picture. AI and Computers, 28 May 2023,
www.tharawat-magazine.com/wp-content/uploads/2019/03/shutterstock_1082585840-Converted-1-1068x742.png

Evans, Eric. Domain-Driven Design : Tackling Complexity in the Heart of Software. Boston, Mass. ; Munich, Addison-Wesley, 2014.

Freeman, Eric, and Elisabeth Robson. Head First Design Patterns : A Brain-Friendly Guide. Sebastopol, Ca, O'reilly, Edition: 10Th Anniversary Ed, 2014.

Gamma, Erich, et al. Design Patterns : Elements of Reusable Object-Oriented Software. Boston, Addison-Wesley, 1994.

Geron, Aurelien. Hands-on Machine Learning with Scikit-Learn, Keras, and Tensorflow 3E. S.L., O'reilly Uk Limited, 2022.

Grinberg, Miguel. Flask Web Development. "O'Reilly Media, Inc.," 5 Mar. 2018.

Hunt, Andrew, and David Thomas. The Pragmatic Programmer. Addison-Wesley Professional, 20 Oct. 1999.

Martin, Robert C. Clean Code a Handbook of Agile Software Craftmanship. Upper Saddle River [Etc.] Prentice Hall, 2010.

McConnell, Steve. Code Complete. Pearson Education, 9 June 2004.

“Recommendation System - Machine Learning - Javatpoint.” Wwww.javatpoint.com, www.javatpoint.com/recommendation-system-machine-learning. Accessed 28 May 2023.

Zhang, Qian, et al. “Artificial Intelligence in Recommender Systems.” Complex & Intelligent Systems, 28 May 2023, <https://doi.org/10.1007/s40747-020-00212-w>.

9.14. Substantive Reflection

Doing this project was an honor because the people I worked with were extremely supportive and helpful in this journey. They knew that my success is their success and this is what team work should be about. We offered each other a helping hand. They gave me the opportunity to sharpen my skills in programming and showcase my skills. Closing off this program with this project means a lot to me. The project requires a lot of reflection over what expectations the client has and how best I can meet them. During this whole process I faced many hurdles but I kept my confidence up in order to keep going. In this project I worked alone but the team members I worked with were part of the software development team so they offered me their expert help whenever I approached them.

After having gone through the whole project I can confirm that I got to learn a whole lot. Even though at this stage we are expected to know how to do all these things but not everything is known so the element of learning continues. This is probably the biggest project we have done and each time the bar raises which is great. I'm not afraid to put in hard work because this is what I wanted to be able to do, build applications for all sorts of use cases to help change the world for the better and give me some skill which I can use to earn myself a living.

Doing documentation is certainly a monumental task and I had spent 90% of my time working towards building the SSR system however the document also needs a lot of work and perhaps counts for 50% of the work so I feel there is an imbalance as to what we are expected of.

Once again the client for which I built the system is hugely appreciative of my efforts and they were flexible for me to work from anywhere just like the rest of their staff members as getting to the office took a whole lot of time from my residence. In all fairness I'm somewhat relieved that this long journey is soon to end to start new things to come. A more detailed separate self reflection is here https://docs.google.com/document/d/1Us3ZDsR1mWdSHGxDyWeXQLCf_F10dbfDEWWyntzcyB0/edit?usp=sharing

9.15. Acknowledgements

These are probably the last words I need to include in my graduation documentation. I would like to mention a most special thanks for my friend Nico M. who connected me with Arno S. who got me working on this assignment. I thoroughly enjoyed being part of their team in Deventer and Amstelveen. Even if I didn't get to work directly with them I still owe them for making me part of the team.

In no particular order a big thank you to..

Shilpa, Amir, Ilse, Arno, Rasha, Wouter, Yaser, Jorge, Coen, Mojo, Michelle, Dena, Mustapha, Nico, Jonathan, Jordy, Joost and the list goes on..