BSc Computer Science

Academic Year 2019 - 2020

COM3001: PROFESSIONAL PROJECT REPORT

A DESTINATION RECOMMENDATION SYSTEM

Student: Valentin Foucault

Supervisor: Professor Yaochu Jin

UNIVERSITY OF SURREY

URN 6497215 - vf00070@surrey.ac.uk - valentinfoucault.com

Declaration of Originality

I confirm that the submitted work is my own work and that I have clearly identified and fully acknowledged all material that is entitled to be attributed to others (whether published or unpublished) using the referencing system set out in the programme handbook. I agree that the University may submit my work to means of checking this, such as the plagiarism detection service Turnitin® UK. I confirm that I understand that assessed work that has been shown to have been plagiarised will be penalised.

Valentin Foucault

May 2020

Abstract

With the rapid development of the internet, recommender systems have become more and more important in our daily lives. From recommending physical products on e-commerce websites such as “Amazon”, or recommending specific content to watch on video streaming services such as “Netflix” and “YouTube”, or even to recommending people to add as “friends” on social networking websites such as “Facebook”, a lot of people interact with recommender systems on a daily basis without necessarily knowing it.

As I would like to be working in the travel industry in the future and am fascinated by recommender systems, I wanted to understand how they work and what kind of impact do they have in this industry. So, I developed a Destination Recommendation System (DRS) that takes advantage of the latest research on recommender systems and web development techniques to enhance the customer experience within the travel industry.

This report presents the results of all the research that I have made on the subject of recommender systems and some of the LSEP (Legal, Social, Ethical, and Professional) aspects surrounding them, alongside with a description of how my DRS was designed, implemented, and tested.

Acknowledgements

To start with, I would like to thank all the professors and researchers that I encountered at the University of Surrey throughout my bachelor. This project is a combination of all the knowledge that I acquired in the Computer Science area over the past 3 years.

In particular, I would like to thank my supervisor, Professor Yaochu Jin, for all the guidance that he gave me to build my final year project and write this report.

Finally, I would like to express my gratitude for all the work the has been done by the open source community over the past decades. This project wouldn't have been possible without the millions of lines of code that have been written by its members to develop the tools and libraries that were used to build the implementation of this project.

Table of Contents

Table of Figures

Table of Tables

Acronyms / Abbreviations

AI	Artificial Intelligence
AJAX	Asynchronous JavaScript And XML
AWS	Amazon Web Services
BCS	British Computer Society
BE	Back-End
CLI	Command-Line Interface
CS	Computer Science
DRS	Destination Recommendation System
FE	Front-End
GAE	Google App Engine
GCP	Google Cloud Platform
GUI	Graphical User Interface
IDE	Integrated Development Environment
LSEP	Legal, Social, Ethical, and Professional
MVC	Model–View–Controller
MVP	Minimum Viable Product
OTA	Online Travel Agency
PAAS	Platform As A Service
PCA	Principal Component Analysis
RMSE	Root Mean Square Error
RS	Recommendation System
SAGE	Self-Assessment for Governance and Ethics
SPA	Single-Page Application
SVD	Singular Value Decomposition
UI	User Interface
UUID	Universally Unique IDentifier
VCS	Version Control System

Section 1: Introduction

1.1 Overview of the project topic

The goal of this project was to develop a Destination Recommendation System (DRS) to improve the customer experience within the travel industry.

A Destination Recommendation System (DRS) is a computer system that takes advantage of a Recommendation System (RS) to provide recommendations to tourists for travel destinations based on features acquired on them such as their tastes or their behaviours.

A Recommendation System usually consists of three components:

- **An interface**, with which the user interacts to provide features about him and receive recommendations.
- **A database,** that contains data about both the users and the items to recommend
- **A recommendation engine**, that takes data from the database and processes it to generate recommendations for the user

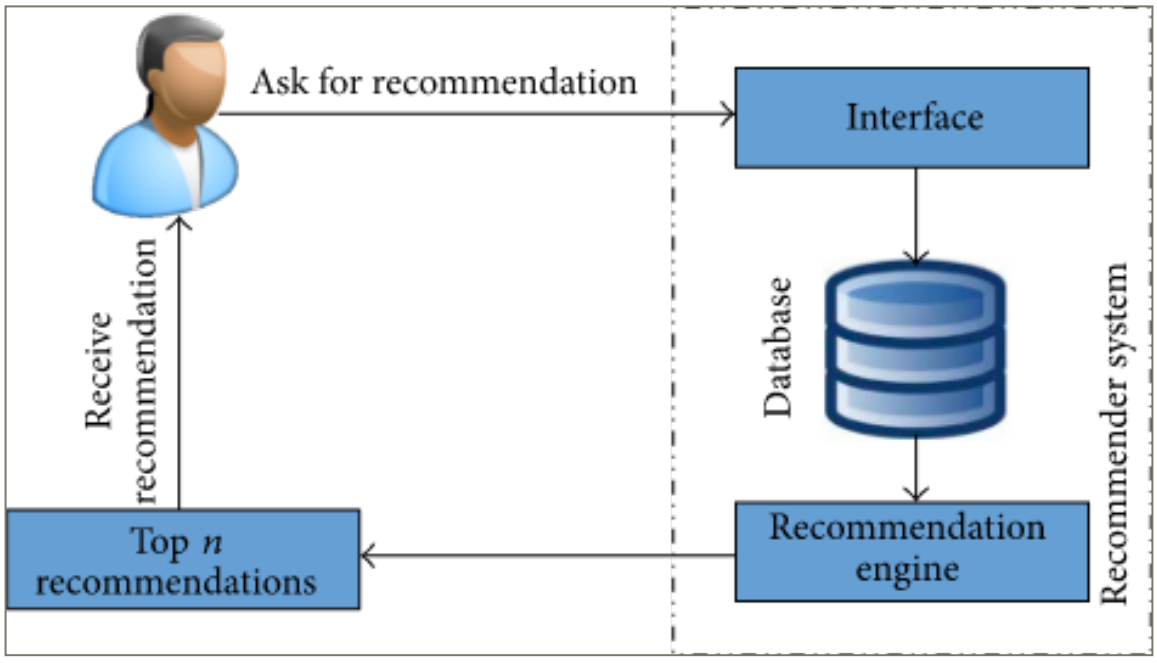

Figure 1: Overview of a Recommendation System (RS) [1]

Throughout the entire duration of the project the components of the RS were reviewed, designed, developed, and tested.

1.2 Background & Motivation

Having spent about three years studying Computer Science (CS) at the University of Surrey and working with a variety of clients as a Freelance Software Developer, I thought that my final year project would be a great opportunity for me to apply all the knowledge that I acquired both academically and professionally over the past years. Thus, I have chosen to work on a problem

which would allow me to combine my Artificial Intelligence (AI) skills with my Front-End (FE) and Back-End (BE) skills.

In addition to that, my belief was that this project could be a great opportunity for me to learn more about certain technologies and development techniques used on a regular basis by IT companies, such as how to deploy micro-services on the Google Cloud Platform (GCP), or how to configure a MongoDB database cluster on MongoDB Atlas.

Finally, as I aim to work in the travel industry in the future, my thinking was that this project could allow me to have an interesting involvement in the industry and open doors to potential collaborations with companies in the sector. I already worked on a travel search engine called SnapAbroad (https://www.snapabroad.com/) in the past, but was unable to bring it to profitability due to certain issues with the business model.

1.3 Project benefits

If the objectives of the project are fulfilled and the system works correctly, it would be able to benefit travellers by enabling them to discover new destinations that they never thought of, and that are in accordance with their personalities and tastes.

On top of that, the fact that travellers would be able to discover new destinations could contribute to the economies of certain destinations such as La Paz (Mexico) or Fez (Morocco) that are less known (yet still attractive) than very popular destinations like Thailand, Bali, or Paris.

Finally, the success of the project could demonstrate how recommender engines can be used to improve the customer experience within an industry and, as I aim to make the project open-source, this could also enable new developers to learn how to build a recommender engine and how to develop a web application with the technologies used in this project.

1.4 Aim and Objectives of the project

Based on the project's topic, an aim and a variety of objectives have been defined.

Aim of the project: To develop a Destination Recommendation System (DRS) that takes advantage of the latest research on recommender systems and web development techniques to enhance the customer experience within the travel industry.

Objectives of the project:

- Review the literature available on recommender systems
- Develop a hybrid recommender engine for the travel industry by using content-based, collaborative-based, and knowledge-based filtering
- Design, implement, and test a user-friendly web application that uses the recommender engine to appropriately display the recommended destinations to the user
- Evaluate the developed system against the requirements of the project
- Review the Legal, Social, Ethical, and Professional aspects of the project

Whether if the aim and objectives have been fulfilled or not in this project will be covered in the conclusion (Section 7) part of this report.

1.5 Success criteria of the project

We will consider that the project is successful based on if it is able to fulfil its aim: "To develop a Destination Recommendation System (DRS) that takes advantage of the latest research on recommender systems and web development techniques to enhance the customer experience within the travel industry." To measure that we will evaluate the system in the section 5 (Testing & Evaluation) against the exact requirements of the project, that are defined in the section 3 (System Requirements and Specification).

1.6 Overview of the developed system

The developed system can be accessed at this address: https://com3001.valentinfoucault.com/.

While the technical details of how the system works will be explained in the System Design & Implementation (section 4) part of this report, the end product that the user can access and interact with is presented here.

As shown on Figure 2, when arriving on the home page the user is able to see a selection of 100 destinations. Under each of the destinations, he can select either three options:

- **Want to go,** when the user already has plans to go to this destination
- **Already visited,** when the user has already visited the destination
- **Favourite,** when the user particularly likes a certain destination (this will have a strong impact on the nature of the recommendations)

If the user doesn't rate a destination, we just assume that he's not interested in this particular destination (yet).

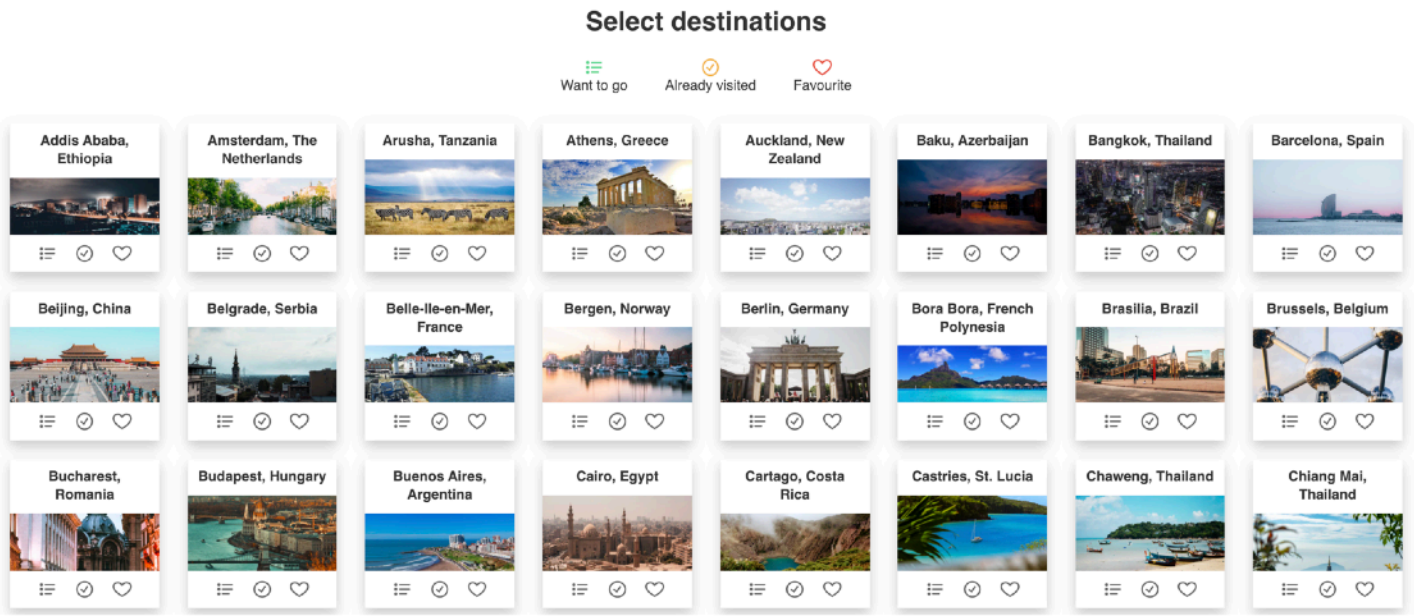

Figure 2: Home page of the application

After rating the destinations that he's already interested in, the user clicks on "Compute recommendations" at the bottom of the page. His ratings are sent to the recommendation engine and, after waiting for a few seconds on a loading screen, the user can then discover 10 destinations that are recommended to him by order of importance, as shown on the figure 3.

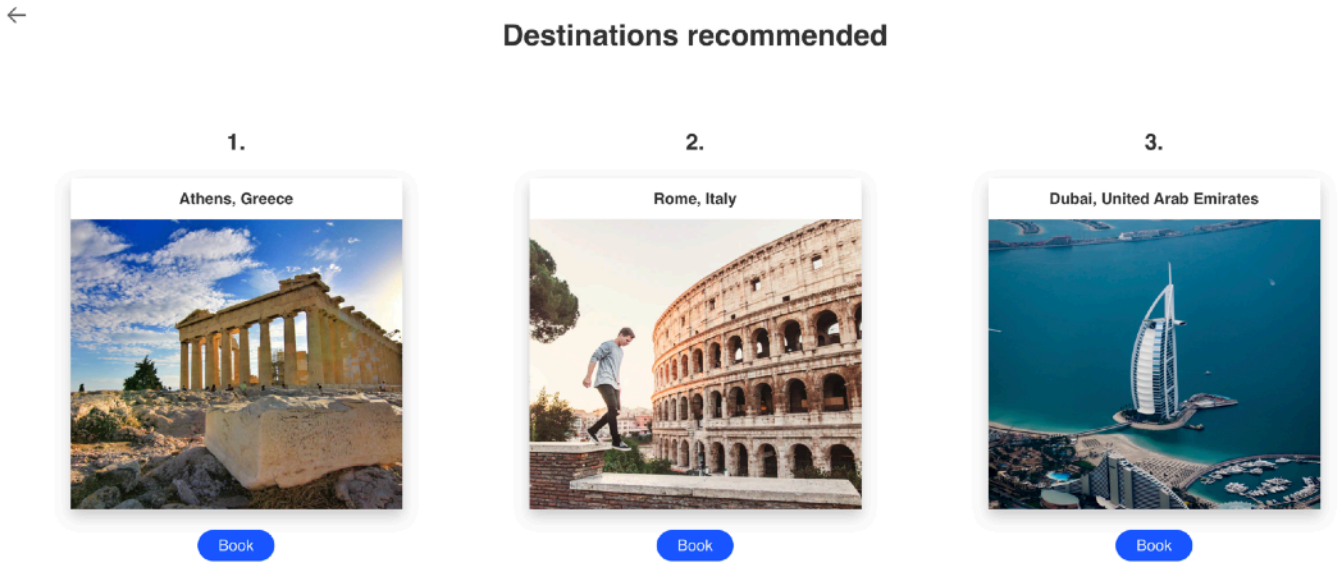

Figure 3: "Recommendations" page of the application

1.7 Structure of the report

This report is structured into 7 different sections with specific purposes, as follows:

Section 1 (Introduction): Introduces the project.

Section 2 (Literature Review): Provides a comprehensive explanation of the topic. Reviews literature (such as research papers) on recommendation systems and key technologies used to build the project.

Section 3 (System Requirements and Specification): Specifications of the project. Requirements gathering and prioritisation. Identification of the key features (and the optional ones) of the project.

Section 4 (System Design & Implementation demonstration): Application of the chosen development methodology to design the system. Justification of the design choices and highlights the challenges encountered throughout the development of the project.

Section 5 (Testing & Validation): Evaluation of the built system against the requirements of the project.

Section 6 (Statement of Ethics): Legal, Ethical, Social, and Professional (LESP) considerations for the project.

Section 7 (Conclusion): Conclusion of the report. Review of the aim and objectives. Review of what went well, what didn't go well, and what I learned throughout the project. Discussion on potential improvements to the project in the future.

Section 2: Literature Review

The goal of this section will be to make a literature review of the subject, by analysing a variety of papers written on the theory behind recommender engines, and by taking a look at some interesting implementations.

2.1 Recommender engine techniques

To start with, we shall mention that when building a recommender engine there are four main approches: the collaborative approach, the content-based approach, the knowledge-based approach, and what we call the “hybrid” approach.

Each of these approaches have their own particularities with their own advantages and disadvantages, which we are going to review.

2.1.1 Collaborative filtering

The collaborative filtering approach is the most commonly used approach when building a recommender engine [2]. In order to determine a recommendation for a specific user of the system, it will use the “ratings” of similar users of the system on specific items to determine recommendations [2]. The more ratings are collected on the users’ behaviour, the more useful and precise the recommender engine will become [3]. This approach can be summarised as “What is popular among similar users”.

One of the main issues with collaborative filtering is what we call the “cold-start problem” [4]. Because this approach needs to have a lot of data on the other users of the system in order to be able to provide recommendations, it may not work correctly in the beginning if the system has a very little amount of data. As content-based filtering doesn’t need to have data on the other users in order to be able to provide recommendations, the cold-start problem could potentially be solved by using a combination of collaborative and content-based filtering techniques [5].

There are two types of collaborative filtering: the memory-based approach, and the model-based approach. The difference between the two is that the memory-based approach will find similar users in the system by using cosine similarity or a weighted average of ratings, while the model-based approach will directly use machine learning in order to find user ratings by using methods such as Principal Component Analysis) (PCA), Singular Value Decomposition (SVD), Neural Nets, and Matrix Factorisation [6].

In contrary to the model-based approach, the main issue with the memory-based approach is that it is not very scalable because the performance reduces the more the data becomes sparse [3].

2.1.2 Content-based filtering

The content-based approach consists of recommending specific items to the user (such as destinations) based on his past behaviour with (for example) the items that he bought in the past, the content that he watched, or his past travel behaviour [7]. This approach can be summarised as “Give me recommendations for similar items”.

In order to create the “recommendation profile” for each of the users, the system would focus on two types of information [8]:

- A model of his preferences
- A history of his interactions with the system

One of the big advantages of using this approach is that, in contrary to collaborative filtering, it doesn’t suffer from the “cold-start” problem [9].

2.1.3 Knowledge-based systems

The knowledge-based approach consists of recommending items by using a “knowledge base” containing each of the items’ characteristics and the users’ needs, instead of just data on the users’ behaviours [10]. This approach can be summarised as “Which items would fit my needs”.

A knowledge-based system is usually made of three components [10]:

- **Interface:** Allows the user to query the system
- **Inference engine:** By interacting with the knowledge base, gleans insights to make and support decisions
- **Knowledge base:** Contains the actual “expert” knowledge, encoded as rules. The solutions to old problems are represented as “cases”.

The knowledge-based approach used to be the main approach used within the travel industry to make recommendations to the user. It has now been replaced by a combination of content-based filtering and collaborative filtering [11].

2.1.4 The hybrid approach

The hybrid approach cannot actually be considered as a proper filtering technique. It consists of mixing multiple filtering techniques (such as the collaborative one the content-based one, and the knowledge-based one) within one single recommender engine to make better recommendations [12].

By combining several filtering techniques, we may be able to balance the disadvantages of certain methodologies with the advantages of others. When it comes to determining the tourists’ preferred destination, a PhD thesis from the Bournemouth University [11] estimated that the most appropriate approach to use would be the hybrid approach, by using content-based and collaborative filtering.

2.2 Existing implementations of recommender engines

Although we’ve analysed the “theoretical’ side of things for recommender engines in the previous section, I believe that it would now be interesting to have a look at the practical side of things.

2.2.1 Usage of recommender engines in companies

Recommender engines are already used in a wide range of industries such as e-commerce, video streaming, and social networking.

The main reason why companies decide to implement recommender engines in their systems is because they push users to buy more products and spend more time on their platform. As such, there is a direct correlation between the performance of a recommender engine and how much profit the company can generate [13].

Great examples of companies using recommender engines are the e-commerce giant Amazon (which uses them to recommend products to its users), the social networking website Facebook (which uses them to recommend “friends” to add), and the video streaming website YouTube (which uses them to recommend content that the user may like).

On average, it has been estimated that the implementation of a recommender engine leads to a 10-25% increase in revenue for the companies [14].

2.2.2 The Netflix prize

Since recommender engines can have a direct impact on the profit generated by a company, an extensive amount of research has been made over the past couple of years to perfect them. A great example of that is when Netflix, a company well known for providing video streaming services, decided to organise a contest called the “Netflix Prize”. Participants were given a training data set that consisted of a list of ratings, users, and movies, and we're asked to develop the best recommender engine possible [15].

The competition, which spanned over three years from 2006 to 2009 and awarded a US$1,000,000 prize to the winning team, enabled Netflix to develop an algorithm that was far more superior to Netflix's own algorithm by improving the RMSE (Root Mean Square Error) score between the predicted ratings for a set of movies and the actual ratings by more than 10% [15].

2.2.3 Recommender engines within the travel industry

While we’ve seen that recommender engines are already used in a wide range of industries, we shall also have a look at its usage within the travel industry.

A variety of papers have already been written on the subject. As such, a research from the University of Klagenfurt on recommendation technologies within the Travel and Tourism industry [16] showed that, among the various recommender techniques that are available, collaborative filtering is the most widespread within the industry. Although content-based filtering is also presented as an interesting methodology for the industry, its weaknesses on product categorisation don’t make it a preferred candidate.

One way to potentially fix this problem is by using a hybrid recommender. By combining collaborative filtering with content-based filtering, we could mutually suppress the disadvantages of the two to improve the quality of the recommendations that are being made [16].

Section 3: System Requirements and Specifications

This section aims to make an analysis of what kind of system should be built and determine its requirements and specifications. It also discusses the software development methodology that should be used to build the system, and how the architecture of the system should look like.

3.1 System analysis

Prior to defining the requirements and specifications of the system, we shall first identify the existing recommendation systems in the industry.

3.1.1 Existing solutions

When it comes to determining destination recommendations for a traveller, the main developments that have already occurred in the industry come from the Online Travel Agencies (OTA). Indeed, the major OTAs such as “Kayak”, “Expedia”, or “Booking.com" are notably known for using recommendation engines to recommend destinations or specific hotels to their customers. Those recommendation engines have a direct economic impact for them, because they result in an increase in bookings, and so an increase in revenue [17].

As an example, flight booking website Kayak displays recommendations for certain destinations directly on its front page to attract potential customers. Those recommendations are automatically generated based on the home airport of the traveller, the current prices for the next two months, and if those prices represent a drop compared to the prices usually charged at this period of the year.

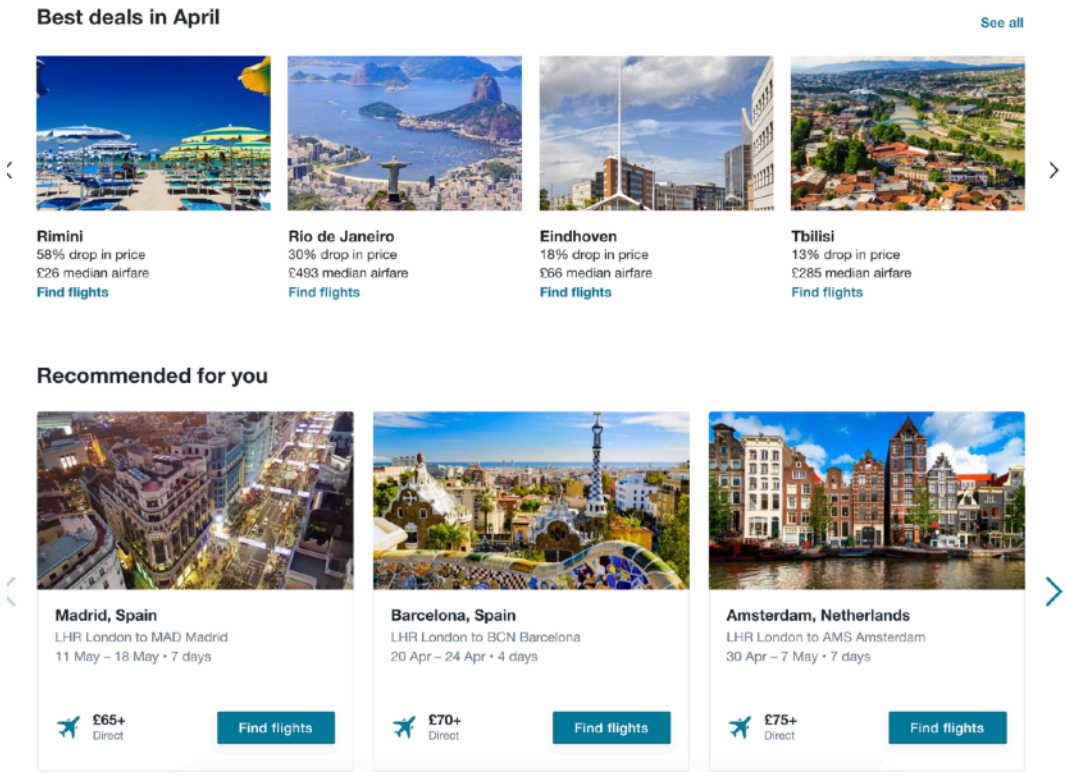

Figure 4: Kayak recommendation system

Recommendation systems within the travel industry are so important that in 2013, Expedia even organised a contest with a total cash prize of $25,000 [18] with the aim to improve its recommendation engine so that it could “learn to rank hotels to maximise purchases”.

3.1.2 Proposed solution

As the aim of the project was to “enhance the customer experience within the travel industry”, the proposed solution was designed to offer something innovative and useful to the travellers, while still following the standards and best practices in place within the industry.

The main characteristics of the proposed solution should be the following:

- **Easy to use:** The whole system must be designed to be used by a wide range of people from a wide range of age groups, so it should be easy to understand and easy to interact with.
- **Innovative:** The system should bring something that truly changes the industry to differentiate itself from the competition and fulfil its aim.
- **Complete:** The system should have enough functionalities and data on the destinations so that it can be considered as “complete”.
- **Reliable:**. Recommendations must be accurate, and the system should be available 24/7 with a downtime as low as possible.
- **Secure:** Due to the sensitivity of the data that will be stored on the users, data privacy and security should be at the core of the development process from the beginning.
- **Fast:** The user interface should work without lag and the recommendations should be generated quickly enough so that the user doesn’t want to leave the website before seeing them.

As this project was quite ambitious and I knew that my time wouldn’t be unlimited, it appeared to me that it would be interesting to separate the project into “key” and “optional” features. The “key” features are the features that must absolutely be part of the project in order to meet the aim and the characteristics set out above, while the “optional” features are features that could potentially be added to the system later on, but that are not needed to fulfil the project’s aim.

3.1.2.1 Key features

Our goal with the key features is to be able to have a first version of the system that is sufficient to rate and recommend destinations based on an AI model that uses collaborative filtering.

As such, the key features of the system should be as follows:

- To have a database of 100 destinations spread across the entire globe with their exact name, country, and a “cover picture” taken from the website unsplash.com (a repository of images that are in the public domain).
- To have an interface that allows the user to visualise those 100 destinations and, if he wants, to have the ability to rate them based on three criteria: “Want to go” (when the user would be

interested in going there), “Already visited” (when the user has already visited the specific destination), “Favourite” (when the destination is among the user’s favourites).

- To have the possibility to pass the user’s ratings to a recommender engine which, based on the ratings from other users on the destinations, will be able to determine recommendations for specific destinations that the user should visit.
- To have an interface that is able to display the 10 most recommended destinations to the user, while having a button under each of the recommendations that would redirect to a travel website that would allow the user to plan a trip to the destination.
- To be able to identify the user so that, if he/she wishes to submit new ratings in the future, his new ratings can replace the old ones in the database.
- To have a sufficient level of security within the system to ensure that the users’ data are kept safe at all time.

3.1.2.2 Optional features

The optional features are features that could be added later on to improve the system.

The potential improvements to the system that I identified are as follows:

- Instead of visualising the destinations as a plain “list”, having the possibility to visualise them on an interactive map directly, with additional appreciable functionalities such as being able to filter the destinations, access more information on them, and leave comments.
- Enable the user to register an account on the website so that his ratings and recommended destinations can be saved and retrieved later more easily.
- Instead of forcing the user to enter all of his destinations’ ratings manually each time he wants to get recommendations, enable him to connect to the website through his Facebook account, so that the ratings can be fetched directly from his “pages liked” and “places visited” on Facebook.
- Improve the recommender engine by replacing the collaborative filtering with a “hybrid” filtering that would pass the ratings of the user through several filters (collaborative-based, content-based, knowledge-based…) at once in order to make the most precise recommendations possible.
- Add a proper branding to the website to make it look more “professional”.

3.2 System Requirements

Based on the proposed solution set out above and the key features that must be implemented, we can now determine the functional and non-functional requirements of the system. Functional requirements are defined as requirements that specify what the system should do, whereas non-functional requirements are defined as requirements that specify how the system should work to achieve a specific functionality.

3.2.1 Functional requirements (prioritised)

The functional requirements have an ID (to identify them more easily later in this report), a name, a description, a priority (in the range Low / Medium / High / Essential), and an explanation of how to properly evaluate them.

The list of functional requirements is as follows:

ID	Name	Description	Priority	Evaluation
F1	See 100 Destinations	The system should be able to display 100 destinations from around the world.	Essential	The requirement is fulfilled whenever the user can access 100 destinations from around the world.
F2	Rate destinations	The user should be able to rate each of the destinations with the ratings “Want to go”, “Already visited”, and “Favourite”.	Essential	The requirement is fulfilled as long as the user can rate the destinations, and those ratings can be sent to the Back-End.
F3	Get recommendations	The user should be able to receive 10 recommendations for potential destinations to visit based on his ratings.	Essential	The requirement is fulfilled as long as long as the user can visualise 10 recommendations computed for him after he clicks on “compute recommendations”.
F4	Book destination	The user should be able to go on a travel website to “book” a trip to one of his recommended destination.	Medium	The requirement is fulfilled if, by clicking on a “Book” button below the recommended destination, the user is redirected on the destination’s page on a travel website.
F5	Legal terms	The user should be able to have access to the legal terms of the website.	High	The requirement is fulfilled if the system provides links to a privacy policy and a cookie policy in the footer.

ID	Name	Description	Priority	Evaluation
F6	Block nonessential cookies	In order to comply with the GDPR regulation, the user should have the possibility to refuse nonessential cookies.	High	The requirement is fulfilled if a “GDPR cookie banner” is shown to the user when he first loads the website, enabling him to accept or refuse nonessential cookies.

Table 1: Functional requirements of the project

3.2.2 Non-functional requirements (prioritised)

Same as for the functional requirements, the non-functional requirements have an ID (to identify them more easily later in this report), a name, an exact description, a priority (in the range Low / Medium / High / Essential), and an explanation of how to properly evaluate them.

Our list of non-functional requirements is as follows:

ID	Name	Description	Priority	Evaluation
N1	Identify user	The system should be able to identify the user (via a cookie for example) so that his/her records can be updated in the database whenever new ratings are being submitted.	High	This requirement is fulfilled if a cookie that consists of an automatically generated ID is saved on the user’s device.
N2	Security of the data	The system should ensure that the users’ data are protected at all times.	High	Requirement is fulfilled if the website uses an SSL certificate to encrypt the user’s transmitted data, and access to the database is properly controlled with a password. Regular backups should also be made.

ID	Name	Description	Priority	Evaluation
N3	Limited Server usage	In order to reduce the costs of the system as much as possible, each of the servers of the system (com3001-frontend, com3001-backend, com3001-recommender) should only run when needed, i.e whenever there's a user on the website.	High	This requirement is fulfilled if the three servers are automatically started when a user accesses the website, and are automatically turned off after the user leaves it.
N4	Scalability	The website should be scalable enough so that it could manage a potential surge in traffic.	Medium	We consider this requirement to be fulfilled if new instances of com3001-frontend, com3001-backend, and com3001-recommender can be automatically deployed whenever there's a surge in traffic on the website.
N5	Low loading times	The loading times of the website should be as low as possible so that the user can have a great experience.	Medium	Requirement fulfilled if the website can load in less than 5 seconds, and the recommendations can be generated in less than 15 seconds.
N6	Block nonessential cookies	In order to comply with the GDPR regulation, the system should be able to block nonessential cookies (such as Google Analytics).	High	The requirement is fulfilled if we can observe that nonessential cookies are effectively not saved on the user's device.

ID	Name	Description	Priority	Evaluation
N7	Generate recommendations	By using the ratings of the user and collaborative filtering, the recommendation engine (com3001-recommender) should be able to generate at least 10 recommendations for the user.	Essential	Requirement is fulfilled if the deployed instance of com3001-recommender is able to return 10 recommended destinations based on a list of ratings from a user .
N8	Responsiveness	The web application should be responsive so that it can be used from a mobile.	High	Requirement is fulfilled if all of the key functional requirements are fully met on a mobile device, and the user interface adapts automatically on mobile.

Table 2: Non-functional requirements of the project

3.3 Software specifications

Based on the requirements defined above, we shall also define the exact software specifications of the system. Our goal here will be to ensure that we select the technologies and tools that re the most appropriate for building the system.

Due to its reliability and scalability, I have chosen to host the Front-End, Back-End, and Artificial Intelligence components of the project on the Google Cloud Platform (GCP), and more specifically on the Google App Engine (GAE) module.

3.3.1 Front-End

The technologies that will be used on the front-end are the following:

- Axios - HTTP client used to make API requests
- Create-react-app - React app template
- Git - Version control system for the project
- Grommet - UI library
- Js-cookie - Cookie JavaScript API to handle cookies
- Node.js - JavaScript runtime to develop and build the bundle of the React.js app
- React.js - Web application framework

- React-router - A library to create and navigate between routes on React.js apps
- React-spinners - UI library to display spinners during loading time
- Redux - State container library for keeping a “general state” within the React.js app
- Redux-thunk - Middleware for the Redux library to enable asynchronous tasks (such as API calls) while updating the state
- Styled-components - Component styling module using CSS
- Uuid - Library for the creation of RFC4122 UUIDs
- Visual Studio Code - Integrated Development Environment (IDE)
- Yarn - Dependency manager used to add/install/remove Javascript modules on the project

3.3.2 Back-End

The technologies that will be used on the back-end side are the following:

- Axios - HTTP client used to make API requests
- Cors - Express.js middleware to enable Cross-Origin Resource Sharing (CORS) requests
- Dotenv - Javascript module to load environment variables from .env files
- Express - Web application framework to build a server that can handle API requests
- Git - Version Control System (VCS) for the project
- Mongodb - Node.js driver to interact with MongoDB databases
- Morgan - Express.js middleware used to log HTTP requests in the console
- Node.js - JavaScript runtime to run the server
- Visual Studio Code - Integrated Development Environment (IDE)
- Yarn - Dependency manager used to add/install/remove Javascript modules on the project

3.3.3 Artificial Intelligence

The entire Artificial Intelligence component of the project (with com3001-recommender, com3001-tripadvisor-profile-finder, com3001-tripadvisor-profile-scraper) use the following technologies:

- Dnspython - DNS toolkit for Python to use MongoDB’s “mongodb+srv://" connection URIs
- Falcon - Web API framework to handle the API requests
- Git - Version Control System (VCS)
- Gunicorn - Python WSGI HTTP Server

- Lightfm - Python implementation of a number of popular recommendation algorithms for collaborative filtering
- Microsoft Bing Web Search API - Web search API used to find random public TripAdvisor profiles
- Node.js - JavaScript runtime to run the script that finds random public TripAdvisor profiles using the Microsoft Bing Web Search API
- Pip - Package installer for the Python ecosystem
- Pymongo - Python driver to interact with MongoDB databases
- Python-dotenv - Package that loads environment variables from .env files
- Python3 - The Python programming language and runtime
- Scikit-learn - Machine learning library
- Scipy - Library used for scientific and technical computing
- Scrapy - Python web scrapper tool to scrape the TripAdvisor profiles
- Visual Studio Code - Integrated Development Environment (IDE)
- Yarn - Dependency manager used to add/install/remove Javascript modules on the project

3.3.4 Database

When choosing a database technology, one of the most important things to think about is to determine if we want the database to be of a “Relational” type or of a “NoSQL” type. While a “relational” type is quite suitable for enterprise projects that need a good amount of reliability, in the end I made the choice to go with the “NoSQL” MongoDB database because it is quite flexible and would allow me to scale the project more easily in the future (requirement N4).

Due to its reliability and scalability, I have also chosen to host it on the Google Cloud Platform, and to administrate it by using the MongoDB Atlas service.

3.4 Web architecture

As the system has specific requirements in terms maintenance and scalability, I had to spend some time finding the most appropriate system architecture for the web application.

3.4.1 Existing architectures

I identified 4 “major” architectures, which I analysed in detail in order to determine which one would be the most appropriate for the project.

3.4.1.1 Single-Page Application

The main point of an SPA (Single-Page Application) is that the user interface consists of a JavaScript application that takes care of updating the web page dynamically during the entire usage of the application by the user, without having to completely reload the page whenever you go to a

different route. In order to update the page, it usually uses Asynchronous JavaScript And XML (AJAX) requests [19].

The main advantage of this architecture is that, after the main JavaScript bundle is loaded, the user can navigate on the website pretty easily and with limited interruptions (outside of waiting for the AJAX requests). It feels way more “natural” to the user.

However, the disadvantages of using this architecture are that you need to load an often big JavaScript bundle during the initial load of the page (which may take a long time and could push users to leave the website before the bundle has finished loading). Also, the fact that you sometimes need to make API requests in order to update the page may expose security vulnerabilities, as the route, body, and header of each of the requests will need to be public.

3.4.1.2 Server-Side Architecture

This architecture is the most commonly used architecture on the internet. It works by using Server Side Rendering (SSR) to generate HTML pages that correspond to the user’s requests. The server usually communicates with a database that is hidden from the client [20].

Here the advantages are that, as this architecture has already been used for a long time, most of the existing software development technologies can be used and it may probably be easier to find the correct technology and an extensive amount of documentation. Also, the initial load of the page will seem quicker (as less data needs to be loaded compared to an SPA app), and the app will be more secure as most of the API calls will happen on the server side, not on the client side.

But the disadvantages are that, as a certain amount of data needs to be exchanged between the client and the server each time you load a page, navigating on the website may seem a bit lengthy and less “natural” compared to a Single Page Architecture. Also, as the web development communities tend to prefer SPAs at the moment, most of the recent web development technologies (such as React and Angular) are mainly meant to be used to develop SPAs rather than Server-Side Applications. Thus, using this architecture would prevent us from using the latest technologies.

3.4.1.3 Monolithic Architecture

An application which uses the monolithic architecture has all of its components (UI, database access, business logic..) bundled into a single deployable software artefact [21].

The main advantage of such an architecture is its simplicity: there’s not need to deploy an instance for each of the components, you only have to deploy a single software artefact.

However there are strong disadvantages with this architecture. As the components are not separated and are bundled into a single software artefact, making updates to one of them will require the entire application to be redeployed and, if one of the components crashes, this may potentially bring the entire application down. The code is also more complex, and there may be performance issues if one of the components slows down the entire system.

3.4.1.4 Microservices Architecture

The microservices architecture is a distributed architecture that consists of multiple components which are deployed separately. Each of the components is responsible for a specific area of the app (such as the User Interface, the business logic, the database management...) [22].

There are several advantages to using such an architecture. Indeed, the fact that the app is separated into multiple components makes that it is way easier to maintain it over time, as you only need to update a single component instead of the entire app whenever you want to make a change. This also allows the entire system to be more robust, as if there's an issue with one of the components this won't prevent the rest of the system from working. Also, it may be more easy and cheaper to scale or downscale the app based on the demand, as you would only have to scale a specific component instead of the entire application.

On the other side, the main disadvantage that comes with using this architecture is that it may take longer to deploy it as you have to put several components into production, and that the system may be a bit more complex to understand.

3.4.2 Chosen architecture

For the architecture of this project, I have decided to come up with a hybrid solution by using both a Single Page Application with a Micro-Services architecture. To achieve this aim, on the microservices side I will develop and deploy several independent components that each will be responsible for a specific area of the app (UI, database, business logic..). But the particularity is that The UI component will be an SPA that interacts with all of the other components through API calls.

I made the choice to use a SPA because, as the front-end of the application shouldn't be very large, the inconveniences of a SPA will be outweighed by the advantage of being able to provide a more "natural" experience to the user.

Also, as SPAs and microservices architectures are more and more used, choosing such a hybrid architecture for my application will enable me to learn more about the latest software development technologies and tools that are being used in the industry.

Finally, as I will need to make several changes to the app after I deploy it (if I identify potential bugs or a lack of important functionalities), I would be able to make changes to the app more easily by having to update only the concerned component (thanks to the microservices architecture), instead of the entire app.

3.5 System Development

Before starting to actually build the project, it appeared to me that it would be important to choose the most adapted Software Development Life Cycle Methodology for the project so that I could be as efficient as possible, while ensuring that the project is built in a reliable way.

Among the many different methodologies available, I decided to take the time to analyse in more detail three that looked more promising to me based on their popularity, the requirements of the project, and the fact that I will be the sole developer working on it.

3.5.1 The waterfall methodology

The waterfall methodology is the traditional method that has been used to develop software for many years. It consists of developing in a sequence of "stages", with the particularity that you cannot move to the next stage until the current stage is complete. The five stages are usually as follows [23]:

- **Requirements:** Gathering of the system's requirements
- **Design:** Design of the system's architecture in preparation for the implementation phase
- **Implementation:** Implementation of the system
- **Verification:** Testing phase to make sure that the developed system does what it's supposed to do and is reliable.
- **Maintenance:** Following the deployment of the system, it's a "support" phase to make the necessary updates and ensure that the system will continue to work on the long-term.

The main advantage of this methodology is that it is easy to follow and very structured, thus guaranteeing the reliability of the system on the long term.

But the main disadvantage is that, because the requirements and design of the system must be set out in advance prior to moving to the implementation phase, it may be difficult to add or modify the requirements later on if you discover that an error has been made during the design phase, or that a missing or incorrect requirement has been produced. So it's not very flexible.

In the end I made the choice to not use this methodology because, as the success of the project wasn't guaranteed and I needed to have some flexibility in order to be able to test things easily, I arrived at the conclusion that this method would be too constraining for me and not really adapted to the nature of the project.

3.5.2 The agile methodology

First introduced about 20 years ago with the famous "Agile Manifesto" [24], this is an incremental methodology that enforces four concepts:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan.

It works in what we call "sprints", which are short periods of time with the aim of focusing on a specific feature only. This methodology ensures that the development team focuses on one feature at the time only.

Here the main advantage is that you can split the development of software into multiple iterations, with each iteration delivering a new feature that is fully tested and ready to be deployed in production. Thus, even if you don't have time to develop all the features of the project, you will be able to release the software in advance and improve it over time easily.

The main disadvantage of this methodology is that it may not be very scalable and that it can be hard to estimate the time that it will take to develop a specific feature, thus it often happens that the feature you're focusing on isn't actually ready to be deployed in production by the end of the sprint. This is an issue that I personally encountered several times, especially while I was building a mobile

app with a few other people last year within Surrey University's "COM2027: Software Engineering Project" course.

This methodology could potentially provide me with the flexibility and speed that I need to build this project. However I made the choice to not use it because I found that it would be still a bit too constraining for me and, as I will be the sole person working on the project and won't have to collaborate with other developers, this would be too constraining for me to have to follow a strict pattern of sprints and to have to write documentation that no one (other than the examinators of the project) will have to read.

3.5.3 The prototyping methodology

This methodology consists of developing prototypes of the project and make adjustments very rapidly without a lot of constraints project-wise until the system reaches a version sufficiently good enough to be deployed to the users [25].

There are mainly two variants of this methodology:

- **Throwaway prototyping:** Where all the prototypes that are being created will be discarded later on until a successful one is found.
- **Evolutionary prototyping**: Where a single, robust prototype is created and constantly improved until it is good enough to be considered "final".

The advantage of this methodology is that it allows users to see a version of the software that works already even if the project isn't complete yet. This methodology is quicker than most of the other methodologies because it doesn't require a lot of documentation and can identify the risks and threats surrounding the project early on.

The main disadvantage of this methodology is that it is not very adapted to the projects that need a lot of team collaboration. Also, the fact that doesn't follow a strict process (like with the waterfall methodology) may cause the software produced to not be very reliable.

At the end of my research, I made the choice to use this methodology because it seemed to be the most appropriate based on the facts that it will allow me to have a good amount of flexibility throughout the entire project, will allow me to test new things and iterate rapidly on the AI side until I find a model that works, and will enable me to not have to write an extensive amount of documentation that may not be really useful as I will be the sole person working on the project.

3.6 Feasibility analysis

In order to prevent a potential waste of time and money, it is important to conduct a feasibility analysis to determine if the project is able to meet the requirements listed above on a technical, legal, operational, economical, and scheduling basis.

3.6.1 Technical feasibility

On the technical side of things, we must ensure that the technologies that we are going to use will be appropriate to build the project.

For the software aspect, we should ensure that the technologies that we are going to use are reliable and well supported. Using the JavasScript and the Python ecosystems seems appropriate for such a task as these technologies are already used a lot within the software development communities, so I should be able to fix most of the issues that I will encounter thanks to the big amount of documentation already available online.

For the hardware aspect, it also seems that the scope of the system won't cause any issues. Running a recommender engine doesn't require a very specific hardware, and the flexibility provided by the Google Cloud Platform will enable us to develop and deploy the system very easily. Although we cannot have full control of this aspect because we don't run our own servers, the fact that the Google Cloud Platform is backed by Google and is already used on a wide range of production applications is a good indicator that the application will be reliable and should have a very high availability rate.

3.6.2 Legal feasibility

On the legal side of things, it appeared to me that it would be important to have a look at the regulations in place surrounding the handling of user data.

Although my entire analysis will be explained in more detail in the section 6 of this document, in the end I arrived at the conclusion that, as long as the website complies with the regulations in place in the UK and the European Union (such as the GDPR), there shouldn't be any issues on the legal side of things.

3.6.3 Operational feasibility

The goal of analysing the operational feasibility is to determine if the project can actually be done with the operational resources already available.

As this project is only a software project and doesn't require such an extensive amount of time and expertise in order to develop it, I have determined that I will be capable of developing it entirely by myself.

So, we can consider the project to be feasible from an operational standpoint.

3.6.4 Economic feasibility

When trying to analysis the economic feasibility of this project, we basically try to understand if the project can generate sufficient revenue in order to compensate the costs of building it.

The costs of the project are as follows:

Entity	Cost
Google App Engine server hosting	28 hours of F1 instances free every day, then 0.05 USD/hour
Domain name com3001.valentinfoucault.com	11.60 GBP/year

Entity	Cost
GitHub code hosting	Free
Software Development Technologies and tools (Visual Studio Code, React, Python etc...)	Free (open source)
Iubenda (Tool that handles the legal aspects of the website automatically)	9 USD/month (for 5 websites)
Microsoft Bing Web Search API	Free for 1,000 searches (each up to 500 results) per month
MongoDB Atlas	Free for a basic cluster with Shared RAM & 512 MB of storage

Table 3: Costs associated with the project

Because at this stage we'll stay under 28 hours of F1 instances every day, plus the fact that I had already subscribed to the Iubenda tool and purchased the domain name valentinfoucault.com for other personal projects, I consider the total cost of the project to be equal to zero at the moment.

I also don't plan to generate any kind of revenue with the project for now.

So, although it is true that in theory if we scale the project we won't be able to generate sufficient revenue to compensate to costs of running it, I arrived at the conclusion that this doesn't really matter because the costs are basically at zero at the moment, and scaling the project wouldn't require huge sums of money anyway.

So the project is feasible economically.

3.6.5 Scheduling feasibility

Analysing the scheduling feasibility of the project corresponds to determining if we'll have enough time available to develop and deploy the project.

Although this will be explained later in the document, I decided to separate the project into multiple "stages" in order to ensure that I will have enough time to develop the project, test it, and deploy it before the deadline at the end of May.

If I only focus on the key features of the project that are mentioned in the part 3.1 (which correspond to Stage 1), I arrived at the conclusion that the project should be done around early March, which would give me enough time to write the report and potentially work on some of the "optional features" (which are part of the stages 2 & 3).

So, I found this project to be feasible from a scheduling standpoint.

Section 4: System Design & Implementation

The purpose of this section is to explain the process that I went through to design the system and implement it. We are going to have a look at the project stages, the architecture of the system, its four main components (Front-end, Back-end, Artificial Intelligence, Database), and how it was deployed.

4.1 Project stages

As this project was quite ambitious and I knew that my time wouldn't be unlimited, it appeared to me that it would be interesting to separate the project into multiple stages so that, even if I wouldn't have enough time to complete everything, I could have the most important features of the project ready before the deadline.

The three stages of the project were as follows:

- **Stage 1**: Develop a basic implementation with only the key features of the project. The goal is to develop a basic interface that would allow the user to rate a set of 100 destinations with features such as if he has already been to that destination, if it's on his "bucket list", and if it's a destination that is in his favourites. Then based on those ratings, we would be able to generate recommendations for new destinations that he should visit by using a recommender engine with collaborative-based filtering only.

- **Stage 2**: Improve the system developed in stage 1 by working on the user experience with some of the "optional" features such as a more interactive map to select the destinations and a "Connect via Facebook" button that would enable the user to get his ratings of the destinations directly from his Facebook account, instead of having to manually enter them. At this stage we would also improve the Artificial Intelligence side of the system by converting the engine into a hybrid engine with two filtering mechanisms: collaborative-based filtering and content-based filtering.

- **Stage 3**: Improve the system furthermore by increasing the accuracy of the recommender engine to the maximum and making the front-end look more professional with a proper branding. At this stage the website would be converted into a proper "business" and we would think about ways to possibly monetise it.

By the end of the project, Stage 1 was fully completed and the basis for implementing the hybrid recommender engine (stage 2) was already in place.

4.2 High-level architecture of the system (design)

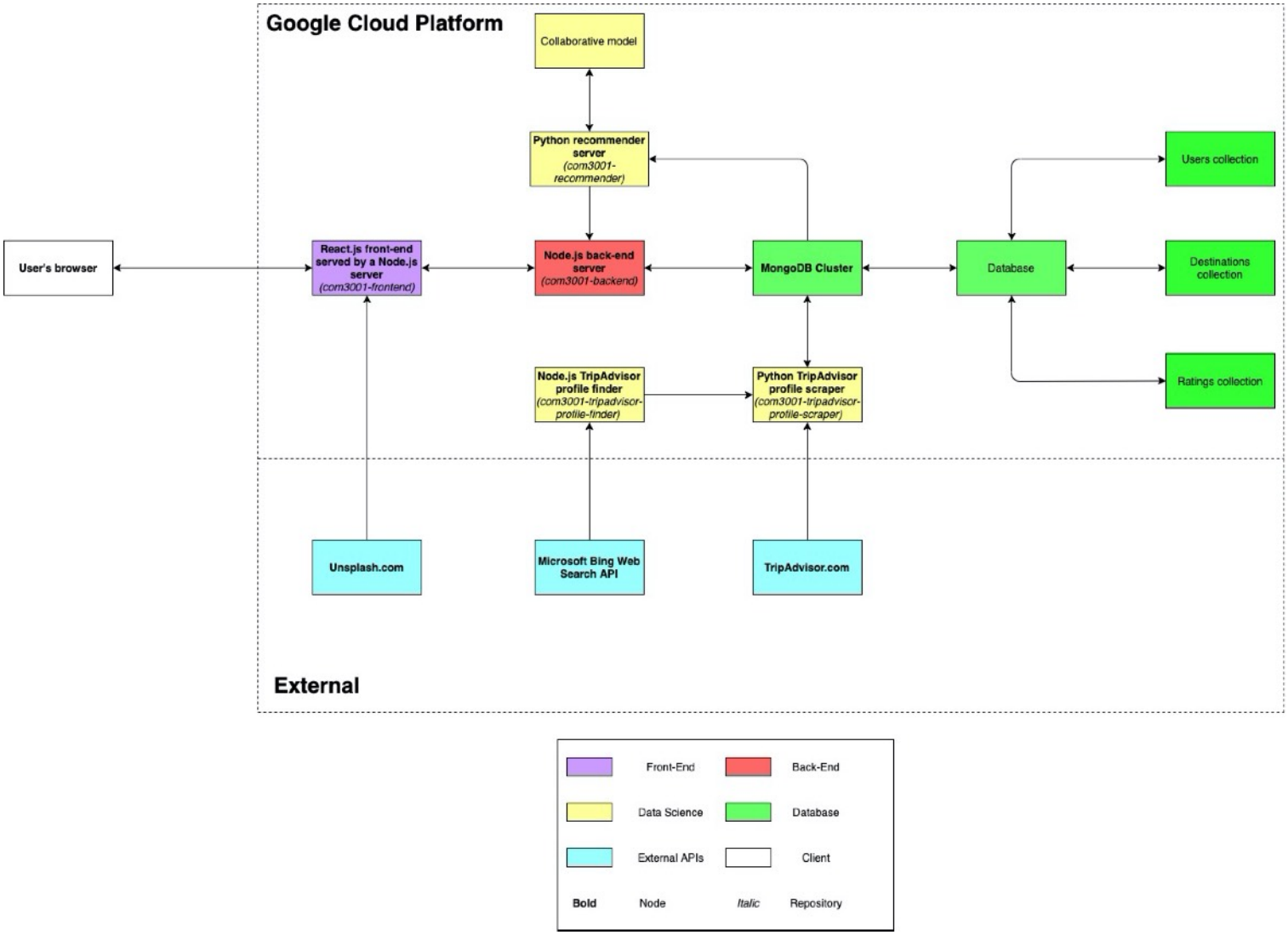

Figure 5: High-level architecture of the system

As you can see on the graph, the system has four major components hosted on the Google Cloud Platform: the front-end component (in purple), the back-end component (in red), the AI / Data Science component (in yellow), and the database component (in green). Represented in blue are all of the external services that were used to build the system as well (Unsplash.com to get images of the destinations, Microsoft Bing Web Search API to find public TripAdvisor profiles, and TripAdvisor.com to get public ratings for the recommender engine).

We are going to analyse each of the four components of the system in the following subsections.

4.3 Front-End component

In the context of a web application, the Front-End corresponds to the presentation layer. It displays the user interface of the system and allows the user to interact with it. For this system, it corresponds to the “com3001-frontend” project.

4.3.1 Design

In order to meet the requirements needed to complete the stage 1 of the project (with the key features only), I made the choice to design the website in a minimalistic way with only five pages:

- **Home page (route /):** Displays the 100 destinations available to the user (with their respective cover pictures), and allows him to rate each of them with either “want to go”, “already visited”, and “favourite”. The user then submits his ratings to the recommender engine by clicking on “compute recommendations” at the end of the home page.
- **Loading page (route /loading):** A loading page that will appear after the user clicks on the “compute recommendations” button. A spinner is displayed, and it will stay there until the recommendations the recommender engine sends back its recommendations.
- **Recommendations page (route /recommender):** This page displays the top 10 recommendations computed by the recommender engine. At the bottom of each of the recommendations a button “Book” is present, that redirects to an external travel website (AirBnB) to allow the user to book an accommodation or a specific experience at the recommended destination.
- **Privacy Policy page (hosted externally):** As required by the EU GDPR regulation whenever you’re handling user’s data, the website needs to have a proper privacy policy in place. As I didn’t have the skills required to properly write one, I decided to use the subscription that I already had on Iubenda (a “legal requirements” tool) in order to generate it.
- **Cookie Policy page (hosted externally):** This page goes in hand with the Privacy Policy page. I also generated and hosted this page thanks to the Iubenda service.

4.3.2 Development

The entire front-end was developed using React.js within a GitHub repository named “com3001-frontend"**.**

In order to go as fast as possible while developing the app, I made the choice to start from a template named “create-react-app” initially. This allowed me to save plenty of time by configuring the project to handle all the packages required to develop with React in a few minutes.

The first thing that I did was to implement the routing between each of the pages, and the GUI by developing a variety of React’s “components”. This enabled me to have a good base before starting to implement the actual business logic.

I then added the business logic part by connecting to the Back-end that I had developed, and by using the state management library Redux (which allowed me that keep a general state throughout all of the components of the app).

4.3.3 Challenges

Although the process of designing and developing the front-end was quite straightforward for me, I struggled a bit when working on the “responsiveness” of the website. Fortunately I managed to fix this problem by using the UI library Grommet, which was designed for making responsive websites.

Another challenge that I encountered was when I had to implement the state management library Redux. Indeed, this library required a good understanding of it before being able to implement it. I solved this issue by searching and reading a certain number of resources on the subject online.

4.4 Back-End component

In the context of a web application, the back-end corresponds to the data access layer. Called by the front-end via Asynchronous JavaScript And XML (AJAX) requests, it enables the front-end to be updated dynamically. In this context, the back-end it is notably used to fetch the 100 destinations available within the project and act as a gateway to the recommender engine and the database in order to compute the recommendations for the user. For this system, the back-end corresponds to the "com3001-backend" project.

4.4.1 Design

As the function of the back-end is mainly to act as a "gateway" to the recommender engine and the database of the project I initially determined that, for the stage 1 at least, it wouldn't require very complex functionalities.

The back-end has two API routes:

- **GET /destinations -** Used by the front-end to retrieve all the destinations available. This route doesn't take any arguments.
- **POST /destinations/recommend -** Used by the front-end to retrieve recommendations based on the ratings of the user. This route also takes care of adding/updating the ratings of the user in the database. The route takes three body parameters: the username of the user, an array containing his ratings, and the number of recommendations to generate.

If we want to implement some of the optional features in the back-end later on (such as the "login via Facebook" one), the current design of the back-end (with API routes, controllers, models, and services) allows us to do this easily.

4.4.2 Development

Again, the development phase of the back-end was quite straightforward.

I first started by using a classic architecture to build Express.js apps, with the folders /api (where the API routes will go), /config (where the config variables of the server will go), /controllers (which handles the requests received by the API routes), /models (which should contain the database models to interact with the database, and /services (which stores the services that are used by the controllers, such as the MongoDB service).

I then implemented the business logic to fetch the destinations from the database and return them to the user, plus the business logic to pass the ratings of the user to the recommender engine and receive the actual destinations' recommendations in return.

In order to transmit data between the front-end and the back-end easily, I have decided to use the JSON format.

4.4.3 Challenges

As I previously had a good amount of experience in building back-end servers using the Express.js framework, I have been fortunate enough to not encounter a lot of challenges while building this component of the system.

Nevertheless, I still struggled a bit while connecting the recommender engine (which runs on Python) with the back-end server (which runs on Node.js). The problem was that I couldn't properly let the two components communicate as the format of the data wasn't always right. I resolved this issue by making sure that the data would always be transmitted using the JSON (JavaScript Object Notation) format, which is a format that is well handled by Node.js and Python.

4.5 Artificial Intelligence component

Separated from the back-end and the front-end, the Artificial Intelligence component of the project consists of multiple services that interact between one another in order to generate recommendations to the user for potential destinations to visit.

4.5.1 Design

When designing the recommender engine, I purposely made the choice to separate it into three different services, as follows:

- **com3001-tripadvisor-profile-finder**: A script written in Javascript that will use the Microsoft Bing Web Search API to find a variety of public TripAdvisor profiles from around the globe.
- **com3001-tripadvisor-profile-scraper**: A script written in Python that scrapes a TripAdvisor public profile found by "com3001-tripadvisor-profile-finder" and returns all the destinations' ratings that the user added to his profile.
- **com3001-recommender**: A Python server that interacts with the users' ratings saved in the database to generate recommendations to a specific user based on a model that uses collaborative filtering.

I made the choice to do this separation because this would make the maintenance of the recommender engine easier in the future, and could allow me to reduce the costs by only running the "TripAdvisor" scripts manually whenever needed, instead of running them permanently as a background service.

4.5.2 Development

One of the most important things to do when developing a recommender engine using collaborative filtering is to make sure that we have access to relevant existing ratings.

So the first thing that I did when I started working on the recommender engine was to find out where I could potentially find those ratings as there was no "official" database of ratings available. After some research, I decided to use some of the ratings that are already available publicly on the travel website TripAdvisor.com.

Initially I had to find a couple of TripAdvisor profiles on which I would be able to scrape the ratings. For that purpose, I decided to write a quick script in JavaScript (com3001-tripadvisor-profile-finder) that would use the Microsoft Bing Web Search API in order to find some. I managed to find a certain number of profiles from 5 different TripAdvisor websites around the globe:

- **Australia (tripadvisor.com.au)**: 988 Profiles
- **Canada (tripadvisor.ca)**: 999 Profiles
- **South Africa (tripadvisor.co.za)**: 986 Profiles
- **United Kingdom (tripadvisor.co.uk)**: 991 Profiles
- **United States (tripadvisor.com)**: 987 Profiles

After removing the duplicates, I ended up with a total of 4888 profiles, which seems sufficient to avoid having the “cold-start” problem.

Following that, I wrote a Python script (com3001-tripadvisor-profile-scraper) that would scrape the TripAdvisor profiles that I found in order to extract all the public ratings of the user directly from his profile, and save those to be used by the recommender engine (com3001-recommender) later on to generate recommendations using collaborative filtering.

In order to develop the recommender engine (com3001-recommender), I used Python to create a model that would use collaborative filtering to generate recommendations by taking the users’ ratings stored in the database. The model was generated using model-based matrix factorisation with the AI library LightFM. Using the Falcon framework, I then developed a quick API server that would take the ratings of the user as input, give them to the model, and then return the recommended destinations as output.

4.5.3 Challenges

I encountered several challenges while developing this component of the system.

First of all, finding and scraping TripAdvisor profiles wasn’t an easy task. To find the profiles, I had to spend a good amount of time finding a proper API that could do this task, as the search giant Google didn’t provide one. After some research, I ended up choosing the Microsoft Bing Web Search API one because it seemed to be the most appropriate one for what I wanted to do.

For the scraping part, the real challenge was that the users’ ratings were not actually shown clearly on the public TripAdvisor profiles. To fix this problem I had to spend a good amount of time understanding how the Scrapy framework works so that I could find the right command that would allow me to fetch the ratings directly from inside the script.

The last challenge that I encountered was when I was building the recommender engine using collaborative filtering. As collaborative filtering for recommender engines wasn’t something that was implemented natively with Python, I had to spend time figuring out how I could potentially implement it. I ended up using the library LightFM, which was quite suitable for the job.

4.6 Database

Within the project, the database is a central component to build a proper recommender engine because it holds all of the ratings of the users, which are then used to compute the recommendations using collaborative filtering.

4.6.1 Design

Inside the database “db”, I decided to have 4 collections:

- **destinations:** Contains a list of the 100 destinations used by the system.
- **ratings:** Contains all the ratings used by the system to determine the recommendations using collaborative filtering (public TripAdvisor ratings and the ratings from the users of the website).
- **tripadvisor_profile_links:** Contains all the TripAdvisor profile links that were fetched using the “com3001-tripadvisor-profile-finder” service.
- **users:** Contains a list of all the users who contributed data to the “ratings” collection.

4.6.2 Development

The development of the database was pretty straightforward.

In order to fill the 4 collections of the database, I initially used the “mongoimport” tool of the MongoDB Command Line Interface (CLI). After filling a CSV file with data on destinations all around the world and getting the TripAdvisor links, ratings, and users from the com3001-tripadvisor-profile-finder / com3001-tripadvisor-profile-scraper services, I manually imported those in the database.

After that, all the data in the database has been updated automatically by the com3001-recommender service whenever it received new ratings for a specific user.

4.6.3 Challenges

The main challenge that I encountered while developing the database was when I was trying to import the data that was generated by the “com3001-tripadvisor-profile-finder” and “com3001-tripadvisor-profile-scraper” services. The problem was that I was trying to import the data into the database from a comma-separated values (CSV) file, and I realised later on that the data would be misinterpreted by the importer because of the way the commas were set inside the file. I fixed this issue by manually editing the CSV file to fix the format and checking that the command used to import the data with mongoimport was correct.

4.7 Deployment of the system (Infrastructure)

Following the development of the three components of the system (Front-End, Back-End, Artificial Intelligence), I then had to have a look at how the system could be deployed.

My idea was to deploy the system on a Platform As A Service (PAAS) because this would allow me to focus more on the development side of things rather than on the deployment. Furthermore, using an infrastructure would allow me scale the system more easily and have access to servers with a high level of reliability.

The three major actors in this space are Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure.

After some research on the advantages and disadvantages of each, I ended up choosing the Google Cloud Platform, and more specifically its Google App Engine (GAE) module.

The main advantage of using the Google App Engine is that it allowed me to deploy the system and scale it very rapidly. With only one command for each of the servers (com3001-frontend, com3001-backend, com3001-recommender), I have been able to deploy the system very easily and, thanks to the scaling capabilities of GAE, new instances of the servers could potentially be deployed automatically based on the demand to scale the system.

For the database part I decided to use MongoDB Atlas, which is a service that enables me to administrate the MongoDB database very easily while hosting it on the Google Cloud Platform.

Section 5: Testing & Validation

Following the design and the implementation of the project, the goal of this section will be to determine if the developed system meets the requirements of the project.

We are going to have a look at each of the functional and non-functional requirements to determine if they are fulfilled.

5.1 Functional requirements testing

5.1.1 F1 Requirement test (See 100 destinations)

Expected outcome: “The requirement is fulfilled whenever the user can access 100 destinations from around the world.”

Test: The system was accessed by loading the website com3001.valentinfoucault.com. A list of 100 destinations appeared on the front page:

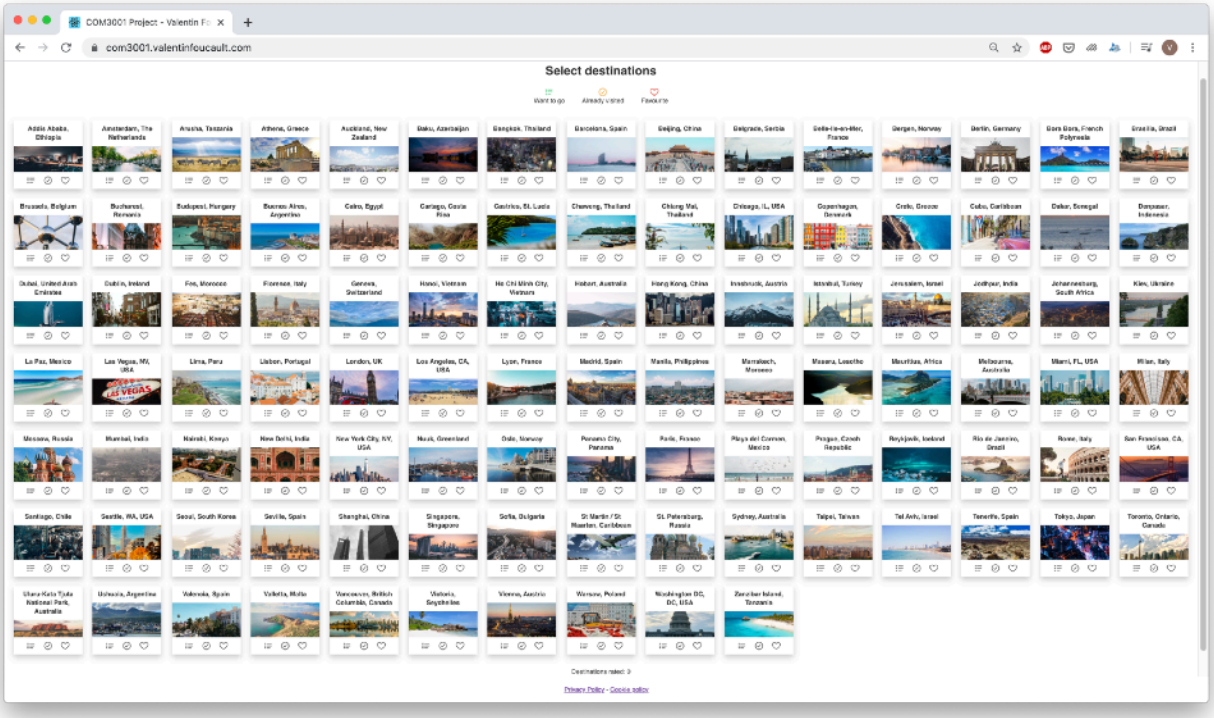

Figure 6: Website’s home page with the 100 destinations

Result: The requirement is fulfilled.

5.1.2 F2 Requirement test (Rate destinations)

Expected outcome: “The requirement is fulfilled as long as the user can rate the destinations, and those ratings can be sent to the Back-End.”

Test: I was able to rate all of the 100 destinations. As an example for Paris, France:

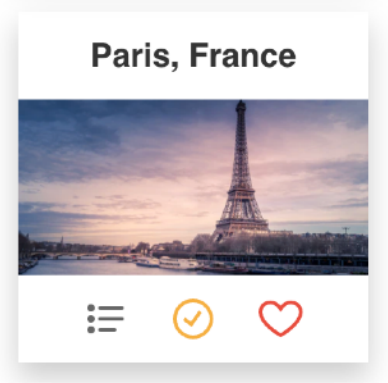

Figure 7: Rating the Paris destination

After clicking on the “Recommend destinations” button at the bottom of the page, the following body was sent to the back-end (“id69” is the id of the “Paris, France” destination):

▼ Request Payload view source
▼ {user: {user_id: "e5161b6c-db99-4a8d-b915-2517d4f298d8",…},…}
num_recommendations: 10
▼ ratings: [{user_id: "e5161b6c-db99-4a8d-b915-2517d4f298d8", destination_id: "id69", rating_value: 12,…}]
▼ 0: {user_id: "e5161b6c-db99-4a8d-b915-2517d4f298d8", destination_id: "id69", rating_value: 12,…}
destination_id: "id69"
rating_origin: "web_app"
rating_value: 12
user_id: "e5161b6c-db99-4a8d-b915-2517d4f298d8"
▶ user: {user_id: "e5161b6c-db99-4a8d-b915-2517d4f298d8",…}

Figure 8: Request payload sent to the recommender engine

Result: The requirement is fulfilled.

5.1.3 F3 Requirement test (Get recommendations)

Expected outcome: “The requirement is fulfilled as long as the user can visualise 10 recommendations computed for him after he clicks on the “compute recommendations” button.”

Test: After rating certain destinations and clicking on the “compute recommendations” button, I was able to see 10 recommendations for potential destinations to visit:

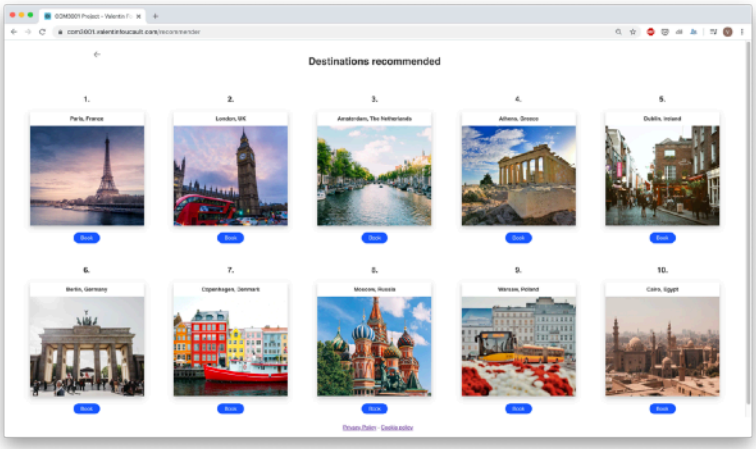

Figure 9: 10 recommended destinations

Result: The requirement is fulfilled.

5.1.4 F4 Requirement test (Book destination)

Expected outcome: “The requirement is fulfilled if, by clicking on a “Book” button below the recommended destination, the user is redirected on the destination’s page on a travel website.”

Test: As an example, I clicked on the “Book” button under the recommended “Paris, France” destination, and ended up on this page:

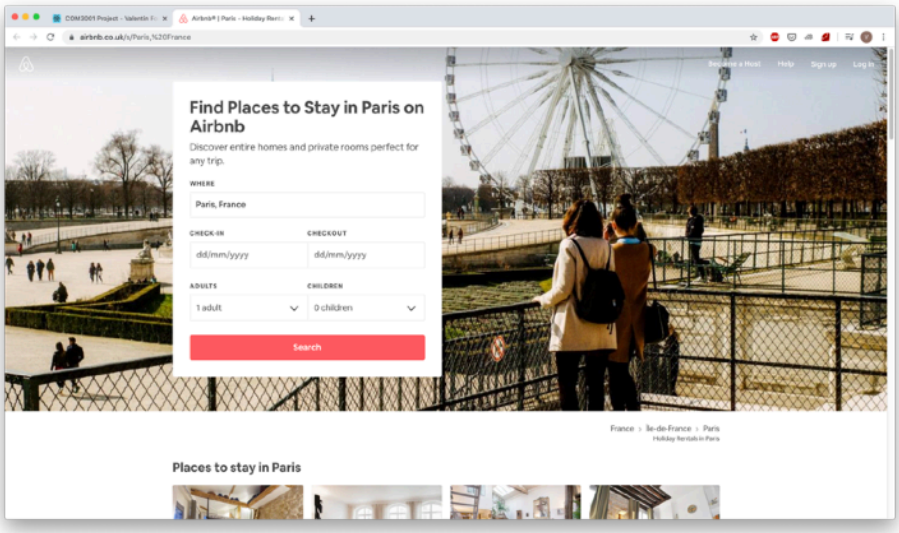

Figure 10: AirBnB page of the Paris destination

This page enables the user to plan a trip to Paris on the travel website AirBnB.

Result: The requirement is fulfilled.

5.1.5 F5 Requirement test (Legal terms)

Expected outcome: “The requirement is fulfilled if the system provides links to a privacy policy and a cookie policy in the footer.”

Test: The footer of the website looks as follows:

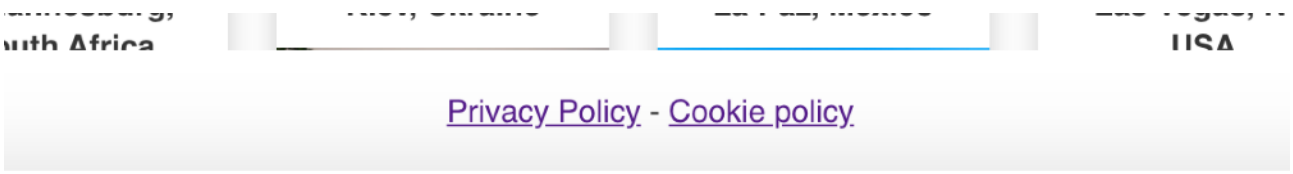

Figure 11: Legal links in the website’s footer

I tried clicking on each of the links. They worked correctly by opening a new tab with the privacy policy and cookie policy of the website.

Result: The requirement is fulfilled.

5.1.6 F6 Requirement test (Block nonessential cookies)

Expected outcome: "The requirement is fulfilled if a "GDPR cookie banner" is shown to the user when he first loads the website, allowing him to accept or refuse nonessential cookies."

Test: When loading the website for the first time, the following appeared at the bottom:

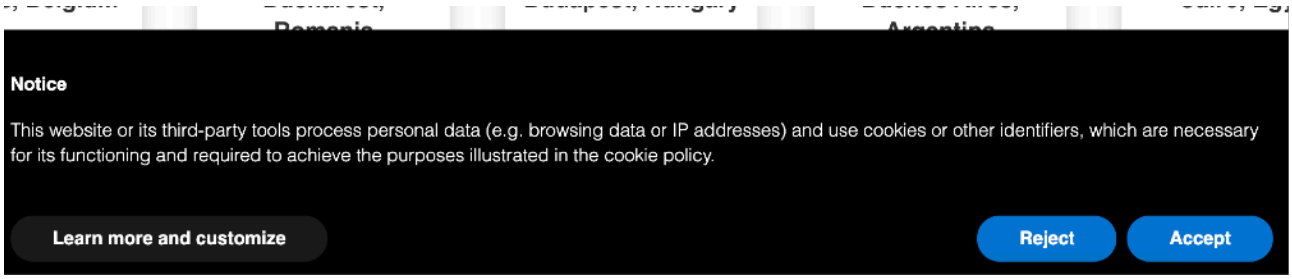

Figure 12: GDPR Banner on the website

Clicking on the "Accept" or the "Reject" button blocks or unblocks the "nonessential cookies" and hides the banner.

Result: The requirement is fulfilled.

5.2 Non-functional requirements testing

5.2.1 N1 Requirement test (Identify user)

Expected outcome: "This requirement is fulfilled if a cookie that consists of an automatically generated ID is saved on the user's device."

Test: After loading the website, the following "userid" cookie was set on the user's device:

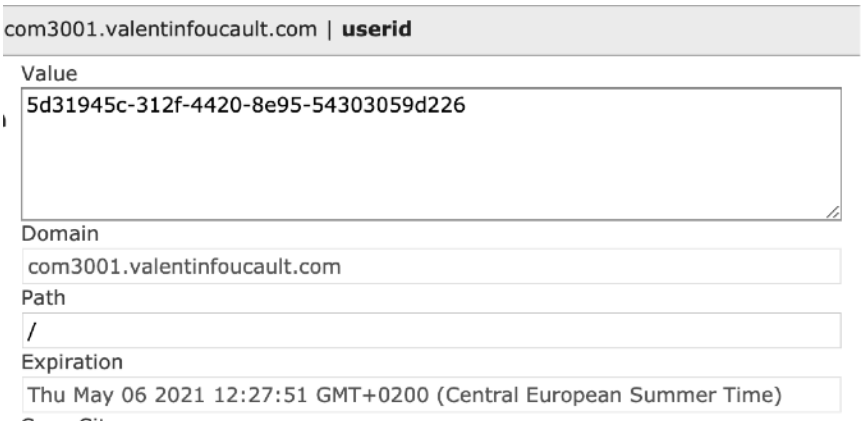

Figure 13: "userid" cookie on the user's device

The cookie consists of a Universally Unique IDentifier (UUID) that is automatically generated.

Result: The requirement is fulfilled.

5.2.2 N2 Requirement test (Security of the data)

Expected outcome: "Requirement is fulfilled if the website uses an SSL certificate to encrypt the user's transmitted data, and access to the database is properly controlled with a password. Regular backups should also be made."

Test: An SSL certificate is present on the website:

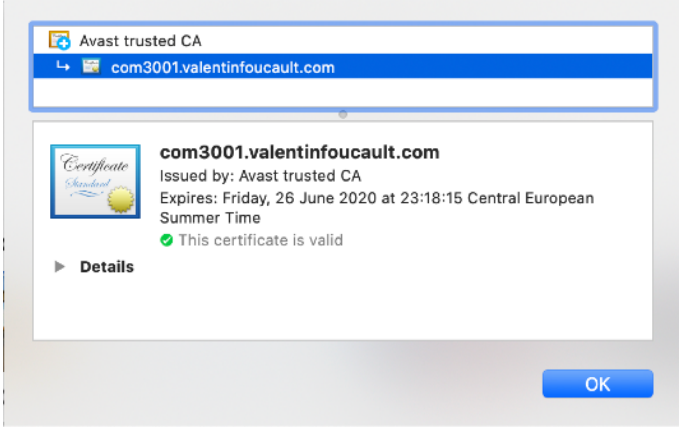

Figure 14: SSL certificate on the website

The database, hosted on the Google Cloud Platform, is protected by a password and its security is fully managed by the MongoDB Atlas service.

Backups can be made very easily thanks to the MongoDB CLI by running the "mongodump" command regularly.

Result: The requirement is fulfilled.

5.2.3 N3 Requirement test (Limited server usage)

Expected outcome: "This requirement is fulfilled if the three servers are automatically started when a user accesses the website, and are automatically turned off after the user leaves it."

Test: By using the "standard" environment of the Google App Engine module with the right settings in the configuration file ("min_instances: 0" on the picture below) for all the servers we can ensure that the instances are automatically started and stopped when needed.

```
# [START scaling]
automatic_scaling:
  min_instances: 0
  max_instances: 1
  min_idle_instances: 0
  max_idle_instances: 1
# [END scaling]
```

Figure 15: General Google App Engine configuration file (app.yaml)

This can be confirmed by looking at the logs of the com3001-backend project, for example:

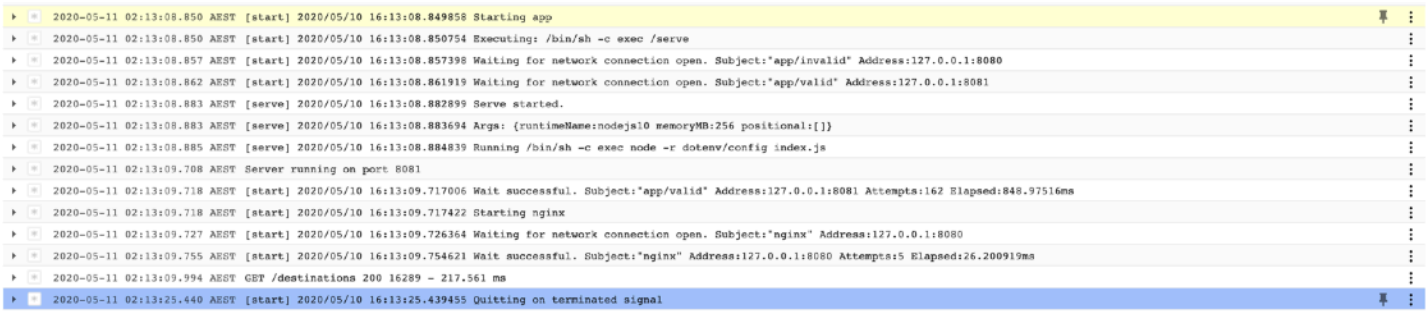

Figure 16: Example log of the com3001-backend project

Here we can see that, as soon as a new request is received (in yellow), a new instance is spawned to handle the new request. If there's no more request afterwards, the instance is automatically shut down (in blue).

Result: The requirement is fulfilled.

5.2.4 N4 Requirement test (Scalability)

Expected outcome: "We consider this requirement to be fulfilled if new instances of com3001-frontend, com3001-backend, and com3001-recommender can be automatically deployed whenever there's a surge in traffic on the website."

Test: Same as for the N3 requirement, deploying on the Google App Engine enables us to scale the system automatically based on the demand. Although I have made the choice to disable this feature for now as it is quite expensive to use, it can be re-enabled at any time.

Result: The requirement is fulfilled.

5.2.5 N5 Requirement test (Low loading times)

Expected outcome: "Requirement fulfilled if the website can load in less than 5 seconds, and the recommendations can be generated in less than 15 seconds."

Test: I tried to load the website and generate recommendations without taking advantage of the web browser's cache. The results were as follows:

Time to load the website: 4.01 seconds
Time to generate a recommendation: 12.12 seconds

The time that it took to load the website was under 5 seconds, and the time that it took to generate recommendations was under 15 seconds.

Result: The requirement is fulfilled.

5.2.6 N6 Requirement test (Block nonessential cookies)

Expected outcome: “The requirement is fulfilled if we can observe that nonessential cookies are effectively not saved on the user’s device after clicking on the “reject” button of the GDPR banner.”

Test: By using the GDPR banner script provided by the company “Iubenda”, the nonessential cookies (such as Google Analytics tracking) get automatically blocked if the user clicks on the “Reject” button to reject the cookies.

Result: The requirement is fulfilled.

5.2.7 N7 Requirement test (Generate recommendations)

Expected outcome: “Requirement is fulfilled if the deployed instance of com3001-recommender is able to return 10 recommended destinations based on a list of ratings from a user.”

Test: To test this requirement, I sent a request to the “com3001-recommender” server with “num-recommendations: 10” as a parameter and the id of a user. The response to that request was as follows:


```
[{"id":"id85","name":"Sydney, Australia","score":0.9043170213699341},
{"id":"id16","name":"Brussels, Belgium","score":0.6252530217170715},
{"id":"id40","name":"Innsbruck, Austria","score":0.6185912489891052},
{"id":"id35","name":"Geneva, Switzerland","score":0.5423282384872437},
{"id":"id34","name":"Florence, Italy","score":0.4945635199546814},
{"id":"id88","name":"Tenerife, Spain","score":0.4175455868244171},
{"id":"id70","name":"Playa del Carmen, Mexico","score":0.30619776248931885},
{"id":"id52","name":"Lyon, France","score":0.2576817274093628},
{"id":"id28","name":"Cuba, Caribbean","score":0.23875287175178528},
{"id":"id57","name":"Mauritius, Africa","score":0.2207687795162201}]
```


Figure 17: Recommendations sent back by the recommender

As we can see here, 10 recommendations were sent back by the server.

Result: The requirement is fulfilled.

5.2.8 N8 Requirement test (Responsiveness)

Expected outcome: “Requirement is fulfilled if all of the functional requirements are fully met on a mobile device, and the user interface is adapted automatically on mobile.”

Test: Requirements F1 to F8 have been tested on mobile, and they are all fulfilled.

The user interface becomes responsive automatically whenever the website is loaded on a mobile phone.

As an example, on an Android phone the user interface looks as follows:

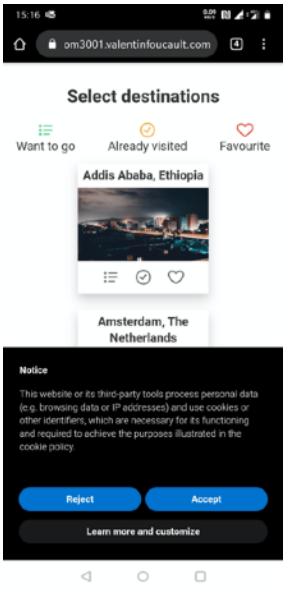

Figure 18: Website's user interface on an Android phone

Result: The requirement is fulfilled.

Section 6: Statement of Ethics

A mandatory element for all of the Final Year Projects' reports, the statement of ethics aims to have a look at all of the Legal, Social, Ethical, and Professional (LSEP) aspects of the project.

We are going to analyse each of those in detail.

6.1 Informed consent & Confidentiality of data - Legal

For the Legal aspect, one of the most important aspects of the project was to ensure that the users of the website could provide an informed consent regarding the usage of their data, and that this same data could be kept confidential at all time.

An important regulation in this domain used to be the "Data Protection Act", voted in 1998 [26]. The Data Protection Act has been updated with the General Data Protection Regulation (GDPR) in May 2018 [27]. Regarding the protection of data, it states the following principles:

- **Lawfulness, fairness, and transparency**
- **Purpose limitations**
- **Data minimisation**
- **Trueness, Accuracy**
- **Storage Limitation**
- **Integrity and Confidentiality**

Following the implementation of this new regulation, the principle of providing informed consent and ensuring the confidentiality of the data that we have on users at all time has been quite reinforced. One of the most important things that came out of this new regulation is the fact that any website which deals with user's data should provide an explicit Privacy Policy and the possibility for the user to refuse having nonessential cookies saved on his device [27].

I managed to comply with these two regulations by implementing a proper privacy policy on the website (available in the footer), and by showing a "GDPR Cookie banner" to the user at the bottom of the screen whenever he first loads the website, allowing him to refuse having nonessential cookies saved on his device.

6.2 Social responsibility (Contribute to society and human well-being) - Social

Social responsibility is defined as the obligation to always act for the benefit of the society at large [28].

In regards to the social responsibility aspect of the project, I would say that there are no major issues here because the goal of the project is first and foremost to recommend destinations that correspond as much as possible to the user's needs.

Nevertheless, one of the things that may possibly be a problem is the environmental concern. As the system that we've developed aims to recommend a variety of destinations from all around the world, this may possibly have a big environmental footprint. The issue is that the system might recommend destinations to the user that are very far away from his current location, which is something that could have a negative impact on the environment. Over-tourism may also be a problem: what would happen if the system were to be biased towards a specific destination, and most of the tourists would be sent there? This can potentially result in a negative impact on the locals with consequences such as an increase in housing prices [29].

In a future version, we may have to update the system so that it can recommend more destinations that are closer to the user or that can be accessed without using environmentally costly modes of transportation such as air travel. The system should also make sure that the destinations recommended are fairly distributed to prevent over-tourism.

6.3 Public interest (Do not harm) - Ethical

Regarding the ethical side of things, one of the most important things was to ensure that the project minimises harms / risks to the public and respects the appropriate laws in the UK.

One of these laws which is quite relevant is the "Computer Misuse Act" [30]. Voted by the British parliament in 1990, this law introduced the following new offences:

- **"Unauthorised access to computer material":** Penalty of up to six months in prison and/or a or up to a £5,000 fine
- **"Unauthorised access to computer materials with intent to commit a further crime":** Penalty of up to a five-year prison sentence and/or an unlimited fine
- **"Unauthorised modification of data":** Penalty of up to a five-year prison sentence and/or an unlimited fine
- **"Making, supplying or obtaining anything which can be used in computer misuse offences":** Penalty of up to a ten-year prison sentence and/or an unlimited fine

One of the ways the project could potentially cause harm to the public is if it would recommend destinations that are potentially dangerous to the user (such as war zones). But at this stage, I have made sure that the 100 destinations recommended on the website are destinations with no huge risks. As long as the user applies due diligence and follows basic security measures, he should remain safe.

Considering the fact that the project doesn't do any of the offences mentioned above, and that at this stage I haven't found how the project could potentially harm the public, we can consider that we have minimised the harms and risks to the public as much as possible and that the project is compliant with the ethical aspect.

You may find the Self-Assessment for Governance and Ethics (SAGE) report in the Appendix A.

6.4 Professional Competence and Integrity - Professional

Finally, this statement wouldn't be complete without mentioning the importance of showing professional competence and integrity throughout the entire duration of the project.

The BCS (British Computer Society) gives us a code of conduct [31] which can enable us to assess this aspect of the project. As a member of the BCS via my Bachelor in Computer Science at the University of Surrey, I believed that it was fundamental to abide by its code of conduct at all times.

The four main principles of the code of conduct are the following:

- **"You make IT for everyone"**:
- "**Show what you know, learn what you don't"**
- **"Respect the organisation or individual you work for"**
- **"Keep IT real. Keep IT professional. Pass IT on"**

I have made sure to abide by these rules throughout the entire duration of the project.

Section 7: Conclusion

This section aims to conclude the report by reviewing some of the key aspects of the project, and by determining the personal benefits that I got out of it.

We are going to make a review of the aims and objectives of the project, what I have learned throughout its entire duration, what went well (and what didn't), and the potential improvements that could be made to the project in the future.

7.1 Review of the aim and objectives

We are going to make a review of the aim and the objectives of the project.

7.1.1 Objectives of the project

Objective 1: Review the literature available on recommender systems

This objective was met in the section 2 (Literature Review) of the report. We've made a review of some of the literature available on recommender systems (mainly research papers).

Objective 2: Develop a hybrid recommender system for the travel industry by using content-based, collaborative-based, and knowledge-based filtering

As we've seen in the section 3 (System Requirements and Specification) and 4 (System Design & Implementation) of this report, we've managed to develop a recommender system using collaborative-based filtering. As I have explained, I made the choice to not develop the content-based filtering and knowledge-based filtering for now as I had feared that I may not have enough time to develop those in the timeframe that was given to me. The code base that has been developed enables the implementation of content-based filtering and knowledge-based filtering.

So, this objective is partially met.

Objective 3: Design, implement, and test a user-friendly web application that uses the recommender engine to appropriately display the recommended destinations to the user

As we've seen in the sections 4 (System Design & Implementation) and 5 (Testing & Validation) of this report, we've managed to develop a user-friendly web application that is able to connect to the recommender engine to send the ratings of the user and get recommendations for potential destinations to visit in return.

So, this objective is met.

Objective 4: Evaluate the developed system against the requirements of the project

This objective was completed in the section 5 (Testing & Validation) of the report. We've made of a complete review of each of the functional and non-functional requirements of the project. All of the requirements were fulfilled.

So, this objective is met.

Objective 5: Review the Legal, Social, Ethical, and Professional aspects of the project

This objective was met in the section 6 (Statement of ethics) of this report. We've had a proper look at all of the LSEP aspects of the project.

7.1.2 Aim of the project

Aim: To develop a Destination Recommendation System (DRS) that takes advantage of the latest research on recommender systems and web development techniques to enhance the customer experience within the travel industry.

Based on the previous subsection in which we've seen that most of the objectives were met, based on the fact that we've managed to develop a real destination recommendation system for the travel industry that took advantage of the latest research on recommender systems and web development techniques, and based on the fact that all of the requirements of the project are fulfilled, I consider the aim of the project to be fully met.

7.2 What I have learned

I am grateful to have had the chance to work on this project for several months. I managed to learn a lot of knowledge on a variety of areas which will be quite useful for my future career.

The following are the key learnings that I got out of this project:

- I depthened my knowledge in AI by researching and understanding all of the key aspects of recommender systems
- I got a better understanding of how to deploy a web app on the cloud and scale it.
- I depthened my knowledge of the various web architectures and software development methodologies that exist
- I got new skills in software development by learning how to set up and administer a MongoDB database and the UI library Grommet
- I've gained a better understanding of the regulations surrounding the usage of data that are currently in place
- I've had the opportunity to get a deeper understanding of how the travel industry works

7.3 What went well, what didn't

In this subsection, I am going to review what went well in the project, and what didn't.

7.3.1 What went well

To begin with, I would say that the development of the front-end and the back-end with React.js and Node.js went pretty well because I already had a good experience in developing with technologies from the JavaScript ecosystem.

Writing the literature review section was quite easy and straightforward as well, because I made sure to keep a daily journal with all of the interesting research papers and articles that I encountered throughout the entire duration of the project.

As the system was pretty straightforward to test, writing the section 5 of the report (Testing & Validation) also went quite smoothly and in a timely manner.

Finally, one of my personal goals with this project was to learn as many things as possible for my future career, which I managed to do.

7.3.2 What didn't go well

When thinking about what didn't go so well, the first thing that came to my mind is the development of the recommender engine. As I didn't have a lot of experience in developing using Python and machine learning libraries, I had to spend quite some time reading the documentation before being able to develop this component of the project.

The second thing that didn't go well, which still relates to the recommender engine of the project, is finding out how I would be able to find a proper dataset to train the recommender engine, as there was none available publicly that could suit the project's needs. I then came to the conclusion that I should build it by myself, using public data from the TripAdvisor website.

Another aspect of the project that didn't go quite well is the question of if whether or not I could use the data that was publicly available on some of TripAdvisor's users. I had to spend quite a big amount of time doing research on that issue by contacting a variety of academics and filling the SAGE statement.

Finally, due to the "Covid-19 crisis" that happened in the middle of the project, I have been forced to move out of the UK to return to my home country (France), which is something that made me lose quite some time for the project.

7.4 Future work

Although I have not been able to develop all of the features that I first thought about developing because of a lack of time, I think that it could be interesting to reflect on the potential direction of the project after June.

Going forward, I believe that it would be great to implement the features that I presented as "optional features" in the Section 3 (System Requirements and Specification) of the report. These include:

- Instead of visualising the destinations as a plain "list", having the possibility to visualise them on an interactive map directly, with additional appreciable functionalities such as being able to filter the destinations, access more information on them, and leave comments
- Enable the user to register an account on the website so that his ratings and recommended destinations can be saved and retrieved later
- Instead of forcing the user to enter all of his destinations' ratings manually each time he wants to get recommendations, enable him to connect to the website through his Facebook account, so that the ratings can be fetched directly from his "pages liked" and "places visited"
- Improve the recommender engine by replacing the collaborative filtering with a "hybrid" filtering that would pass the ratings of the user through several filters (collaborative-based,

content-based, knowledge-based...) at once in order to make the most precise recommendations possible

- Add a proper branding to the website to make it look more “professional”

Although this is not a priority at the moment, I believe that, if the system were to cost more in the future, it could also be interesting to think about the potential ways to monetise it. This could be either on the B2C (Business To Customer) side, or on the B2B (Business to Business) side.

Finally, I believe that it could be interesting to have a look at the other industries in which this recommender system using collaborative-based filtering could potentially be useful.

References

[1] L. Ravi and S. Vairavasundaram, "A Collaborative Location Based Travel Recommendation System through Enhanced Rating Prediction for the Group of Users", 2016.

[2] H. Al-bashiri, "An improved memory-based collaborative filtering method based on the TOPSIS technique (Abstract)", 2018.

[3] "Collaborative filtering", En.wikipedia.org, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Collaborative_filtering. [Accessed: 12- May- 2020].

[4] X. Zhao, "Cold-Start Collaborative Filtering", PhD, University College London, 2016.

[5] Robin Burke. Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction, 12(4):331–370, 2002.

[6] C. Aggarwal, "Chapter 2: Neighbourhood-based Collaborative Filtering", in Recommender Systems: The textbook, C. Aggarwal, Ed. 2016.

[7] R. van Meteren and M. van Someren, "Using Content-Based Filtering for Recommendation.", 2020.

[8] "Recommender system", En.wikipedia.org, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Recommender_system#Content-based_filtering. [Accessed: 12- May- 2020].

[9] F. Moghaddam and M. Elahi, "Cold Start Solutions For Recommendation Systems", 2019.

[10] R. Burke, "Knowledge-based recommender systems.", 2020.

[11] P. Thiengburanathum, "An Intelligent Destination Recommendation System for Tourists", PhD, Bournemouth University, 2018.

[12] R. Burke, "Hybrid Web Recommender Systems", 2007.

[13] A. Azaria, A. Hassidim, S. Kraus, I. Netanely, O. Weintraub and A. Eshkol, "Movie recommender system for profit maximization", 2013.

[14] "CS50 - Recommender Systems", Harvard University, 2015.

[15] "Netflix Prize", En.wikipedia.org, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Netflix_Prize. [Accessed: 12- May- 2020].

[16] E. Teppan, A. Felfernig, S. Gordea, D. Jannach and M. Zanker, "A Short Survey of Recommendation Technologies in Travel and Tourism", University Klagenfurt, 2007.

[17] M. Starkov, "Recommendation Technology in Travel and Hospitality: Or How to Turn Lookers into Bookers", Hospitality Net, 2001. [Online]. Available: https://www.hospitalitynet.org/news/4008935.html. [Accessed: 12- May- 2020].

[18] "Personalize Expedia Hotel Searches - ICDM 2013 | Kaggle", Kaggle.com, 2020. [Online]. Available: https://www.kaggle.com/c/expedia-personalized-sort. [Accessed: 01- Apr- 2020].

[19] "Single-page application", En.wikipedia.org, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Single-page_application. [Accessed: 12- May- 2020].

[20] "Server-side", En.wikipedia.org, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Server-side. [Accessed: 12- May- 2020].

[21] "Monolithic application", En.wikipedia.org, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Monolithic_application. [Accessed: 12- May- 2020].

[22] "Microservices", En.wikipedia.org, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Microservices. [Accessed: 12- May- 2020].

[23] "Waterfall model", En.wikipedia.org, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Waterfall_model. [Accessed: 12- May- 2020].

[24] "Manifesto for Agile Software Development", Agilemanifesto.org, 2020. [Online]. Available: https://agilemanifesto.org/. [Accessed: 12- May- 2020].

[25] "Software prototyping", En.wikipedia.org, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Software_prototyping#Evolutionary_systems_development. [Accessed: 12- May- 2020].

[26] UK Public General Acts, "Data Protection Act 2018", 2018.

[27] European Parliament, "Regulation (EU) 2016/679 of the European Parliament and of the council of 27 April 2016", 2018.

[28] "Social responsibility", En.wikipedia.org, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Social_responsibility. [Accessed: 11- May- 2020].

[29] M. Zemła, "Reasons and Consequences of Overtourism in Contemporary Cities—Knowledge Gaps and Future Research" (p.7), 2020.

[30] UK Public General Acts, "Computer Misuse Act 1990", 1990.

[31] British Computer Society (BCS), "BCS Code of Conduct", 2011.

Appendices

Appendix A: SAGE statement

SAGE

Response ID	Completion date
514292-514283-55636751	29 Mar 2020, 16:27 (BST)

1	**Applicant Name**	Valentin Foucault
1.a	**University of Surrey email address**	vf00070@surrey.ac.uk
1.b	**Level of research**	Undergraduate
1.b.i	**Please enter your University of Surrey supervisor's name. (If you have more than one supervisor, enter the details of the supervisor who will check this submission).**	Prof. Yaochu Jin
1.b.ii	**Please enter your supervisor's University of Surrey email address. (if you have more than one supervisor, enter the details of the supervisor who will check this submission)**	yaochu.jin@surrey.ac.uk
1.c	**School or Department**	Computer Science

2	**Project title**	A destination recommendation system (DRS) to enhance the consumers' experience within the travel industry

3	**For Undergraduate and Masters students, will your student research project be conducted according to a faculty standard study protocol? Your module lead or supervisor can advise if you are unsure.**	NO

4	**Are you making an amendment to a project with a current University of Surrey/NHS REC/other favourable ethical opinion in place?**	NO

5	**Does your research involve any animals, animal data or animal derived tissue, including cell lines?**	NO

6	**This question is deliberately left blank.**	Please click here to continue

7	**Does your project involve* human participants, their data and/or any human tissue?**	YES

8	**Does your funder, collaborator or other stakeholder require a mandatory ethics review (e.g. Institutional Review Board (IRB) review) to take place at the University of Surrey?**	NO

9	**Does your project process personal data1? Processing covers any activity performed with personal data, whether digitally or using other formats, and includes contacting, collecting, recording, organising, viewing, structuring, storing, adapting, transferring, altering, retrieving, consulting, marketing, using, disclosing, transmitting, communicating, disseminating, making available, aligning, analysing, combining, restricting, erasing, archiving, destroying.**	YES

10	**Does your project require the processing of special category2 data?**	NO

11	**If you are an undergraduate or Masters student, are you ONLY using name and contact details for recruitment purposes, and no other personal data is being collected as listed in questions 9 and 10 above?**	YES
11.a	**Will you adhere to the security requirements set out in the 'Data Protection and Security for Undergraduate and Postgraduate Taught Students' Projects'.**	YES

12	**Does your project involve any type of human tissue? This includes Human Tissue Authority (HTA) relevant, or irrelevant tissue (e.g. non-cellular such as plasma or serum), any genetic material, samples that have been previously collected, samples being collected directly from the donor or obtained from another researcher, organisation or commercial source.**	NO
13	**Does your research involve exposure of participants to any hazardous materials e.g. chemicals, pathogens, biological agents or does it involve any activities or locations that may pose**	NO
14	**Will you be accessing any organisations, facilities or areas that may require prior permission? This includes organisations such as schools (Headteacher authorisation), care homes (manager permission), military facilities etc. If you are unsure, please contact RIGO.**	NO
15	**Will you be working with any collaborators or third parties to deliver any aspect of the research project?**	NO
16	**Will you be travelling to non-UK countries for any of your research activities?**	NO
17	**Will any research activities be conducted outside of the UK?**	NO
18	**Does your research involve lone working?**	NO
19	**Certain types of research require ethics approval from a nationally recognised research ethics committee (REC) which operates to standards set out by the Department of Health's Governance Arrangements for Research Ethics Committees. Recognised research ethics committees (REC) include NHS RECs and the MoDREC. Does your research involve any of the following? (select all that apply)**	None of the above
20	**Have you selected any of the options between A-O from question 19?**	NO
21	**Does your project require ethics review from another institution?**	NO
28	**Declarations**	• *I confirm that I have read the University's Code on Good Research Practice and ethics policy and all relevant professional and regulatory and complete information regarding my research project • I understand that a false declaration or providing misleading information will be considered potential research misconduct resulting in a formal investigation and subsequent disciplinary proceedings liable for reporting to external bodies • I understand that if my answers to this form have indicated that I must submit an ethics and governance application, that I will NOT commence my research until a Favourable Ethical Opinion is issued and governance checks are cleared. If I do so, this will be considered research misconduct and result in a formal investigation and subsequent disciplinary proceedings liable for reporting to external bodies. • I understand that if any of my responses to the governance questions have requested additional documents, that these will be provided with my ethics and governance application if my project is to proceed. • I understand that if I have selected any options from Qu 22-27 I MUST submit an ethics and governance application (EGA) for review in order to proceed with this research project UNLESS I am an undergraduate or Masters student, in which case I have completed Qu 29 below.
29	**If I am conducting research as a student:**	I confirm that I have discussed my responses to the questions on this form with my supervisor to ensure they are correct.