

Travel Planning

Application

Name: Cucos-Artene George Emanuel

Group: 6

Table of Contents

[Deliverable 1 3](#_Toc64843130)

[Project Specification 3](#_Toc64843131)

[Functional Requirements 3](#_Toc64843132)

[Use Case Model 3](#_Toc64843133)

[Use Cases Identification 3](#_Toc64843134)

[UML Use Case Diagrams 3](#_Toc64843135)

[Supplementary Specification 3](#_Toc64843136)

[Non-functional Requirements 3](#_Toc64843137)

[Design Constraints 3](#_Toc64843138)

[Glossary 3](#_Toc64843139)

[Deliverable 2 3](#_Toc64843140)

[Domain Model 3](#_Toc64843141)

[Architectural Design 4](#_Toc64843142)

[Conceptual Architecture 4](#_Toc64843143)

[Package Design 4](#_Toc64843144)

[Component and Deployment Diagram 4](#_Toc64843145)

[Deliverable 3 4](#_Toc64843146)

[Design Model 4](#_Toc64843147)

[Dynamic Behavior 4](#_Toc64843148)

[Class Diagram 4](#_Toc64843149)

[Data Model 4](#_Toc64843150)

[System Testing 4](#_Toc64843151)

[Future Improvements 4](#_Toc64843152)

[Conclusion 4](#_Toc64843153)

[Bibliography 4](#_Toc64843154)

# Deliverable 1

## Project Specification

The travel planning application will be a web-based platform that helps users plan their travel itineraries by providing them with various options for flights, hotels, and activities. The application will also allow users to create and save their travel plans and share them with others.

## Functional Requirements

1. *User Registration and Login: Users will be able to create an account and log in to the platform using their email and password.*
2. *Travel Search: Users will be able to search for flights, hotels, and activities based on their travel dates, budget, and preferred location.*
3. *Flight Booking: The application will provide users with flight options from different airlines based on their travel dates and budget.*
4. *Hotel Booking: The application will provide users with a list of hotels in the selected location based on their budget and preferred amenities.*
5. *Activity Booking: The application will provide users with a list of activities to do in the selected location.*
6. *Travel Itinerary Planner: Users will be able to create and save their travel plans on the platform. The application will also allow users to customize their travel plans by adding or removing activities.*
7. *Travel Plan Sharing: Users will be able to share their travel plans with others through email.*
8. *User Profile: Users will have their profile page where they can view their travel history, saved travel plans, and account settings.*
9. *Customer Support: The application will have a customer support section where users can submit their queries or complaints, and the support team will provide them with assistance.*

## Use Case Model 1

### Use Cases Identification

### 1.Use-Case: User Registration

### Level: Primary

### Actor: New User

### Main success scenario:

### User navigates to the registration page.

### User enters its personal information and creates a username and password.

### User submits the form, and the system confirms the registration.

### Extensions:

### If the user enters invalid information, the system prompts the user to correct the errors and resubmit the form.

### If the username is already taken, the system prompts the user to choose a different username

### 2.Use-Case: User Login

### Level: Primary

### Actor: Registered User

### Main success scenario:

### User navigates to the login page.

### User enters its username and password.

### System verifies the credentials and logs the user in

### System redirects the user to their profile page.

### Extensions:

### If the user enters incorrect login information, the system prompts the user to enter the correct username and password.

### If the user forgets their password, the system provides an option to reset their password through email verification.

3.Use-Case: Search for Flights

Level: Primary

Actor: User

### Main success scenario:

* User navigates to the flight search page.
* User enters their travel details such as destination, dates, and number of passengers.
* System retrieves a list of available flights based on the search criteria.
* User selects the desired flight and proceeds to the booking page.
* Extensions:
* If there are no available flights, the system informs the user and suggests alternative dates or destinations.
* If the user needs to make changes to their search criteria, the system provides an option to go back and modify the search parameters.

4.Use-Case: Search for Hotels

Level: Primary

Actor: User

Main success scenario:

* User navigates to the hotel search page.
* User enters their travel details such as destination, dates, and number of guests.
* System retrieves a list of available hotels based on the search criteria.
* User selects the desired hotel and proceeds to the booking page.
* Extensions:
* If there are no available hotels, the system informs the user and suggests alternative dates or destinations.
* If the user needs to make changes to their search criteria, the system provides an option to go back and modify the search parameters.

5.Use-Case: Search for Activities

Level: Primary

Actor: User

Main success scenario:

* User navigates to the activity search page.
* User enters their travel details such as destination, dates, and number of participants.
* System retrieves a list of available activities based on the search criteria.
* User selects the desired activity and proceeds to the booking page.
* Extensions:
* If there are no available activities, the system informs the user and suggests alternative dates or destinations.
* If the user needs to make changes to their search criteria, the system provides an option to go back and modify the search parameters.

6.Use-Case: View Itinerary

Level: Primary

Actor: User

Main success scenario:

* User navigates to the itinerary page.
* System retrieves and displays the user's upcoming travel bookings including flights, hotels, and activities.
* Extensions:
* If the user has no upcoming travel bookings, the system informs the user and suggests they search for and book travel arrangements.
* If the user needs to make changes to their bookings, the system provides an option to modify or cancel the bookings.

7.Use-Case: Customer Support

Level: Primary

Actor: User

Main success scenario:

* User navigates to the customer support page.
* User submits a request for assistance or information.
* System confirms receipt of the request and assigns it to a customer support representative.
* Customer support representative responds to the request and provides assistance or information to the user.
* Extensions:
* If the user is not satisfied with the customer support, the system provides an option to escalate the request to a higher level of support or to submit a complaint or feedback.

8.Use-Case: Manage Content

Level: Admin

Actor: Admin

Main success scenario:

* The admin logs into the admin panel of the travel planning application.
* The admin navigates to the content management section of the admin panel.
* The admin selects the type of content they wish to manage, such as destinations or activities.
* The travel planning application displays a list of content items.
* The admin selects the content item they wish to manage.
* The travel planning application displays the content details and allows the admin to edit or delete the content.
* Extensions:
* If the admin attempts to delete a content item that is associated with existing bookings, the travel planning application notifies the admin that the content cannot be deleted.
* If the admin creates a new content item, the travel planning application prompts the admin to enter all required information.

9.Use-Case: Share Travel Plan

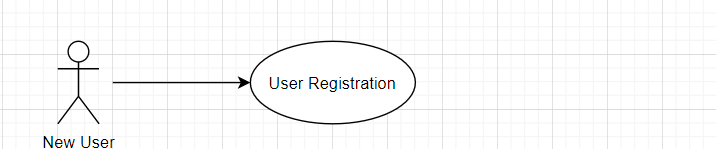
Level: Primary

Actor: User

Main success scenario:

* The user navigates to the travel plan they wish to share.
* The user selects the "share" button or icon associated with the travel plan.
* The travel planning application presents the user with sharing options, such as email, text message, or social media.
* The user selects the desired sharing option and enters the necessary information (e.g., email address, phone number, social media handle).
* The travel planning application sends the travel plan information to the selected recipient.
* The recipient can view the travel plan details and may have the option to save or book the travel plan.
* Extensions:
* If the recipient does not have an account with the travel planning application, the travel planning application may prompt them to create an account in order to view or book the travel plan.

### UML Use Case Diagrams



Diagram

Description automatically generated

A picture containing text, shoji

Description automatically generated

## Supplementary Specification

### Non-functional Requirements

1.Performance

The performance non-functional requirement would ensure that the travel planning application is able to handle a high volume of users and provide a fast response time to user requests. This is particularly important for a travel planning application, as users may become frustrated if they experience delays or long loading times while trying to book travel accommodations or activities. To meet this requirement, the system could implement various techniques such as caching, load balancing, and efficient database queries to optimize performance.

2.Security

The security non-functional requirement would ensure that the travel planning application is secure from unauthorized access, data breaches, and other security threats. This is important as the application will be handling sensitive user information such as personal details. To meet this requirement, the system could implement measures such as encryption of data at rest and in transit, role-based access control, and regular security audits and testing.

3.Scalability

The scalability non-functional requirement would ensure that the travel planning application is able to handle an increasing number of users and transactions without experiencing a degradation in performance or stability. This is important as the application may experience sudden spikes in traffic during peak travel seasons or promotions. To meet this requirement, the system could use scalable cloud infrastructure, such as AWS or Azure, that can dynamically allocate computing resources as needed.

4.Usability

The usability non-functional requirement would ensure that the travel planning application is easy to use and navigate and provides a positive user experience. This is important as users may become frustrated if they have difficulty finding information or completing tasks within the application. To meet this requirement, the system could use user-centered design principles, such as clear and intuitive navigation, prominent call-to-action buttons, and consistent design patterns throughout the application. The system could also conduct user testing and feedback to identify and address any usability issues.

### Design Constraints

1. Technology stack: A constraint is that the application on the server side will be built using Java and the Spring Framework. On client side, the application will be built using Typescript and React alongside HTML and CSS.
2. Integration requirements: The application will use the Sky Scanner API for flight booking and Booking.com API for accommodation booking.
3. User experience guidelines: The application will have a base color scheme consisting of white and blue.
4. Development tools: The application will be developed using the IntelliJ IDEA IDE.

## Glossary

User

Definition: A person who interacts with the travel planning application to search for, book, and manage travel accommodations and activities.

Admin

Definition: A user with administrative privileges who can manage the content and settings of the travel planning application.

Destination

Definition: A location that users can search for and book travel accommodations and activities.

Accommodation

Definition: A place to stay, such as a hotel, hostel, or vacation rental.

Activity

Definition: An experience or event that users can book, such as tours, attractions, or restaurants.

API

Definition: Application Programming Interface, a set of protocols and tools for building software applications.

UI

Definition: User Interface, the visual and interactive elements of the travel planning application that users interact with.

UX

Definition: User Experience, the overall experience and satisfaction that users have when using the travel planning application.

API Key

Definition: A unique identifier that allows a user or application to access a specific API.

Authentication

Definition: The process of verifying the identity of a user or application attempting to access the travel planning application.

Authorization

Definition: The process of granting or denying access to specific resources or features of the travel planning application based on a user's role or permissions.

Cache

Definition: A temporary storage location used to store frequently accessed data or resources in order to improve application performance.

Database

Definition: A structured collection of data used to store and organize information for the travel planning application.

Encryption

Definition: The process of converting sensitive data into a secure format using cryptographic algorithms to prevent unauthorized access.

REST

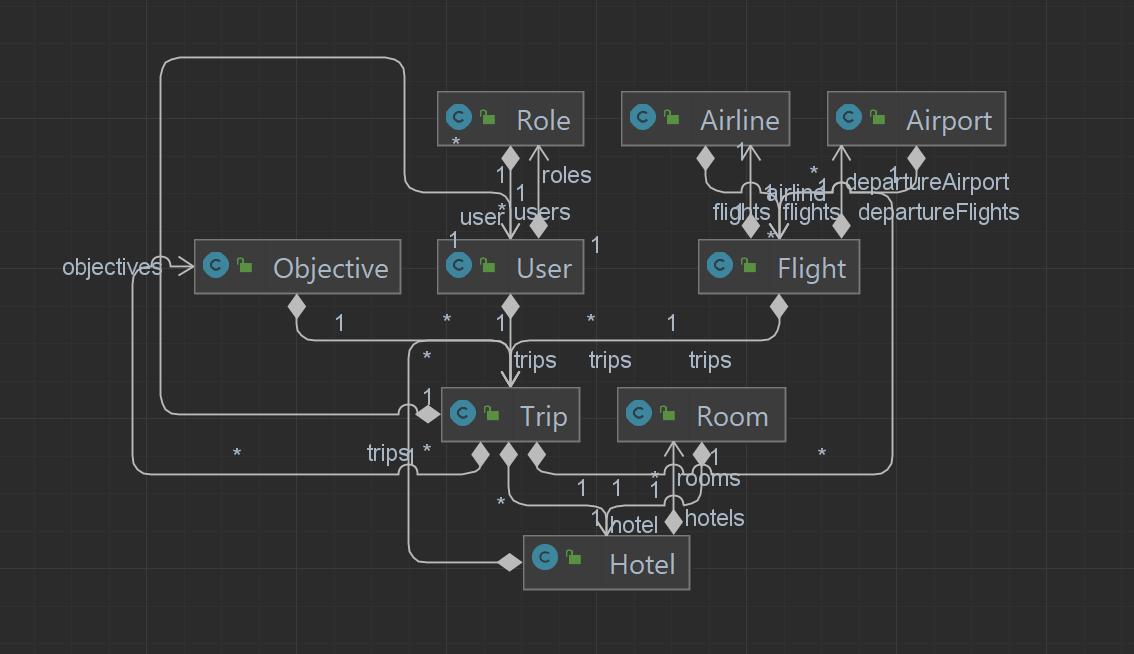
Definition: Representational State Transfer, a software architecture style used to create scalable and flexible web services.

SSL/TLS

Definition: Secure Sockets Layer/Transport Layer Security, a protocol used to establish secure communications between a client and server over the internet.

# Deliverable 2

## Domain Model



## Architectural Design

### Conceptual Architecture

The MVC (Model-View-Controller) architectural pattern is a popular design pattern used in software development. It separates an application into three interconnected components: the Model, View, and Controller.

The Model component represents the domain objects or data entities used in the application. It encapsulates the data and the business logic related to that data. The View component represents the user interface of the application. It displays the data and allows the user to interact with it. The Controller component acts as the intermediary between the Model and View components. It receives the user input from the View, processes it, and updates the Model or View accordingly.

For a travel planning application, the MVC pattern can be a suitable choice as it helps to separate the data, logic, and presentation layers. The Model component can contain the data related to travel destinations, accommodations, flights, etc. and the business logic related to travel planning such as finding the cheapest flights, suggesting destinations based on user preferences, etc. The View component can display the travel plans to the user, and the Controller component can handle user input to modify the data and update the View.

Overall, using the MVC pattern can help in creating a maintainable, scalable, and testable travel planning application. It also provides clear separation of concerns and promotes modularity, which can facilitate further development and integration of new features in the future.

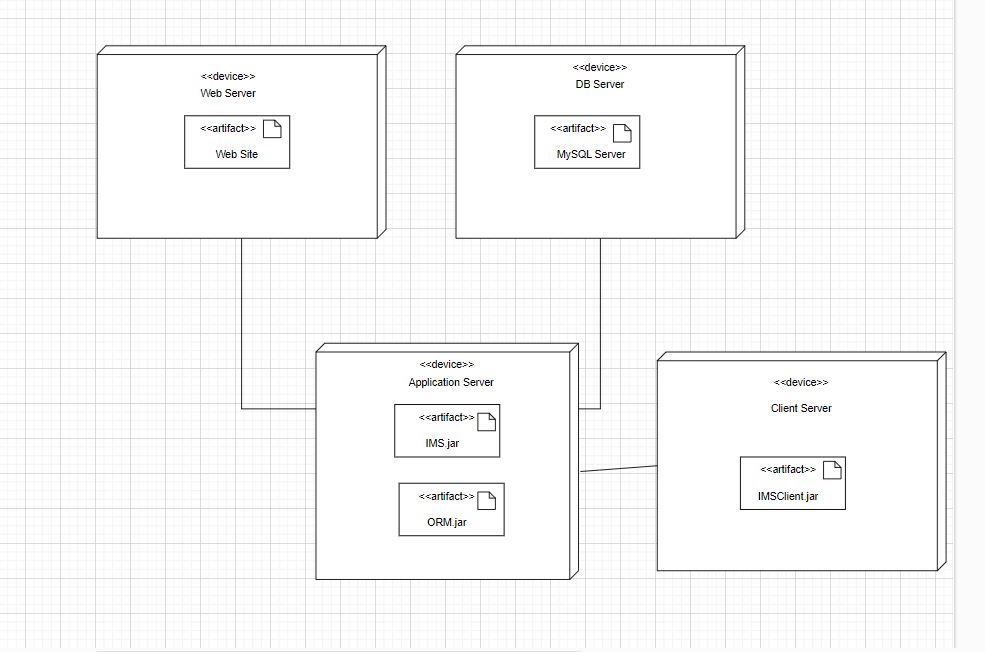
### Package Design

*Chart, box and whisker chart

Description automatically generated*

### 

### Component and Deployment Diagram



# Deliverable 3

## Design Model

### Dynamic Behavior

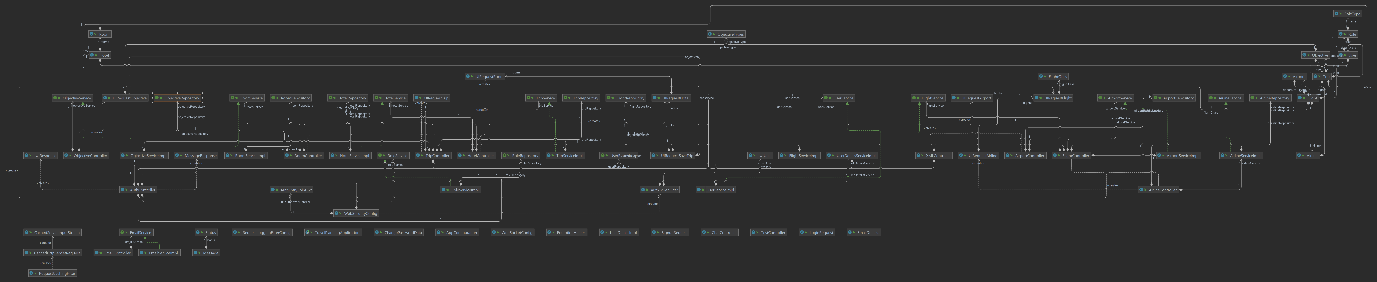
A white rectangular object with black text

Description automatically generated with low confidence

A picture containing text, screenshot, diagram, line

Description automatically generated

### Class Diagram



GoF Patterns:

1. Composite design pattern: Within this project, there are several cases in which this pattern was used. The most relevant of them refer to the Trip class. This pattern is reinforced by the fact that when an employee or a client is deleted, all the related appointments are also deleted.

2. Builder pattern: This pattern is found especially at the security level, where complex objects are built step by step. The implementation is done by using the "@Builder" annotation provided by Lombok and then building the objects through .build(). By using the Builder Pattern, we can build complex objects in a clear and flexible way. It allows us to set different attributes in any order and avoid overloaded constructors with a large number of parameters. Additionally, it gives us the ability to extend and customize the construction process without affecting the base class.

## Data Model

*A picture containing text, diagram, line, parallel

Description automatically generated*

# System Testing

The system was tested through junit tests, which include both common cases and exceptional cases. Each service was tested individually in a specially designed class. Within the test classes related to each service, each method was tested, both for cases completed with success and for cases completed with an error.

For the tests related to the services, we used Mocks related to the repositories used in the tests.

# Future Improvements

1. Implementation of the rating and review system: allows users to leave ratings and reviews for flights, destinations, and objectives. There is the possibility to add a rating system with ratings and comments so that users can share their experiences and provide feedback.

2. Implementing a personalized recommendation system: using recommendation algorithms to suggest services based on customer preferences and history. This can help increase customer satisfaction and promote additional services.

# Conclusion

In conclusion, this project is a complex one, which has various usage flows for each type of user. It is a project that offers various checks and ensures data consistency.

The design and implementation stages had a considerable degree of difficulty, as it was necessary to analyze the various factors that can influence a certain flow, but I consider that the entire effort has resulted in a satisfactory result, which harmoniously combines the complex logic in the backend with the imposing appearance of the front-end.

# Bibliography

<https://www.baeldung.com/spring-boot>

<https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#legal>

<https://docs.spring.io/spring-security/reference/index.html>

<https://www.javaguides.net/2019/06/spring-boot-angular-8-websocket-example-tutorial.html>