Bilkent University
Department of Computer Engineering

Senior Design Project

T2307
Travela

Final Report

Group members

21903224, Yağmur Eryılmaz, yagmur.eryilmaz@ug.bilkent.edu.tr
21901972, Cenk Duran, cenk.duran@ug.bilkent.edu.tr
21902620, Efe Ertürk, efe.erturk@ug.bilkent.edu.tr
21902820, Çağla Ataoğlu, cagla.ataoglu@ug.bilkent.edu.tr
21903248, Efe Şaman, efe.saman@ug.bilkent.edu.tr

Supervisor

Cevdet Aykanat

19.5.2023

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Contents

Final Report

Travela

1 Introduction

Vacations are fun, but planning them, not so much. Especially if you are responsible for planning a holiday trip that suits many people with different interests, budget and available times. This is a serious task for everyone, and on average, a person spends 10 hours planning their holiday, as the survey of Independent Newspaper claims [1]. Moreover, the results of the poll that includes more than 7800 people's opinions from 26 different countries show that over 25 percent of people think holiday planning is one of life's 'biggest stressors'. Thus, our application, "Travela" focuses on solving this problem, while being a total travel companion.

Travela is an app that provides a unique way of planning your holiday trips with your friends. With the machine learning systems we use, the app suggests the best location or route of vacation for your trip group, taking all of the group members' holiday interests, budget and time constraints into consideration. While doing that, the app also suggests to you the popular landmarks or recommended activities in your route, to make your trip well-worth. Also, you can share these holiday plans and your holiday memories on the app, and highlight the places visited in your own unique Globe. By this way, you can see your previous holidays with your most fun memories, and also your friends' too. Thus, it is also a social media app for travel lovers!

In this report, we will start by talking about the functional and nonfunctional requirements we established to guide us in building this project. Then, we provide a detailed view of the final architecture and design of the project, including UML diagrams and the user interface. Then, we explain the development and implementation details and outline which technologies we used and how. Following that, we provide a comprehensive collection of test cases. Then, in the next section, we discuss the maintenance of our application. Finally, we have a section for other elements of the project, such as various factors we considered in our design, our responsibilities and some details about the teamwork aspect of the project. After that we conclude the report.

2 Requirements Details

2.1 Overview

Travela is both a travel companion, and a new social media platform. As for the travel companion aspect, the most important features can be named as enabling users to form travel groups via sending an invitation on the app to arrange a trip together with friends, suggesting users the most suitable vacation by taking each group members holiday interests, budget and time constraints in mind, and suggesting suitable hotels, transportation opportunities, popular landmarks and activities in the vicinity of the travel. For the social media aspect, the most important features can be described as allowing users to see previous trips of both themselves and their friends on unique personalized Globes, create routes and plans according to their map and allowing users to post and attach photos of their trips to their world map which can be seen by the others. The features that are supported by Travela are illustrated in depth in part 3.2 “Functional Requirements" of this report.

To give the customers the best experience, it is vital to gather as much information about a user as possible, to suggest the most suitable vacations. For this reason, it is a good idea to ask the users to solve some sort of a questionnaire, involving questions like “Which of these vacations below do you see yourself in?”, and letting the user choose its own preference. Moreover, integrations with the third party applications that provide the data of nearby hotels, suitable flights or transportation opportunities, like Google hotels, Skyscanner, Amadeus, Travelport, Tripadvisor or Google map reviews will surely increase our service quality as a total travel companion.

Travela is planned to bring innovation of service and customer engagement, as the provided service and functionality of recommending vacations to a group of users by taking account their interests, is different from the already existing products on the market. Also, this app makes planning trips easier and more enjoyable, thus it can be also considered as an innovation of service that increases user experience. The innovation that is planned is also incremental, as it is not achieved by leveraging a radical technology, but it is something that fulfills a need in the market for many users. The machine learning recommender system that we plan on using is similar to the preexisting ones in terms of creating a feature matrix of a person's interests, but whenever a group is formed, these matrices will be taken into consideration together, which is a new technique, but not a radical one. Travela’s business strategy is Digital Business Optimization, which focuses on enhancing customer experiences. Also, it provides a new way of engaging the customers in the tourism / travel planning domain, thus, it can be said that it creates a new digital business model.

As Travela is an app whose success depends on the experience of users within the app, it is essential that the recommender systems are fast and accurate. It is important to mention that the target user group is anyone that plans trips online, especially young people that try to plan holidays through an app with their friend group. Baring that in mind, having an easy and usable user interface for the target customer base is a requirement, which adds to the user experience. One should easily search for trips, form travel groups, view possible vacations for the group and view the activities around the travel route, almost intuitively and without any issue. Our goal as the Travela team is that this app is used internationally, has lots of users in holiday seasons (as it is not realistic to expect people to browse through a vacation app every day), and recognized in the market as a new social media app.

It is important to note that Travela is an intermediary application that helps people to find suitable vacations for themselves, and also, helps hotels, transportation companies, and local landmarks to find their customers. Once a group decides on their travel route, involving where to stay and which transportation unit they will use, they will be directed to actual websites of these services. For example, they will be directed to the web site of the hotel they have chosen for reservation, and the web site of the airline company for buying the seats. There is no reservation or payment functionality on Travela. The advertisements on the app will help Travela to continue giving high quality service to its customers.

2.2 Functional Requirements

2.2.1 Sign up - Login

- Allows users to sign up and login to the system by using their email and password that is determined in the sign up process.
- Enables users to renew their password when it is forgotten using their email.
- Collects data from the user during the sign up process to see his/her preferences like favorite location, preferred routes or activities via a quick questionnaire.

2.2.2 Holiday Arrangement

- Enables users to send a friendship request to other users.
- Enables users to form travel groups via sending an invitation on the app to friends in order to arrange a trip together.
- Enables users to integrate their calendar hence suitable dates to the system.
- Shows the common available dates of the group members via calendar usage.
- Allows users to search destinations that they want to visit.
- Shows suitable hotels and transportation opportunities for the given dates and place.
- Allows users to share destinations and related information (hotel, transportation) with their travel group via chat feature.
- Suggests travel routes according to groups' overall preferences (budget, date) by using ML algorithms.
- Suggests activities (concerts, festivals, etc.) between the given dates in selected locations.
- Allows users to see their upcoming trips that are finalized by the travel group.
- Shows reviews and ratings of the destinations and activities.
- Allows users to see their current travel groups in their account.
- Enables users to create polls among their group to determine possible routes, plans, itineraries, activities, etc.

2.2.3 Social Media Aspect

- Allows users to see previous trips of both themselves and their friends in a World map, create routes and plans according to their map.
- Allows users to post photos belonging to a specific trip to their world map which can be seen from the others.
- Enables users to vote the destinations that they see in the platform as dislike,like and love.
- Enables users to rate their previous trips out of 5 to be able to collect data for suggestion algorithms and also add reviews for the trip.
- Enables users to delete their account if preferred.
- Allows users to take a questionnaire to his/her traveler personality.

2.3 Non-functional Requirements

2.3.1 Usability

- User interface should be understandable and not complex, also catchy to impress users.
- Headings should be in larger font size than the other parts to draw attention.
- Font size for the body should be a minimum 16 px.
- Interfaces should be responsive for both mobile and web.
- Design should be made with a mobile first approach as mobile apps are thought to be used more commonly.
- To apply mobile first design, for iOS the Human Interface Design should be considered and for Android Material Design should be considered [2] [3].
- While fetching data, refreshing animations should be attractive not to distract users.
- It should address all kinds of users from various ages.
- The language should be English to be universal.
- The user shouldn't have to re-login if he/she refreshes the page.

2.3.2 Security

- Users will not be able to enter the system without a password as the app also has a social media aspect.
- Passwords will be saved in the database in an encrypted version.
- Loss of data in a possible crash should be prevented.
- Users' all previous trip data should be removed if the user deletes his/her account.
- An authentication token should be implemented to determine the specifications of a session [4].
-

2.3.3 Maintainability

- Updating the system should not affect the end user's experience for a long time.
- Encapsulation should be used to facilitate detection of problems.
- Consistent commenting method will be used to follow the process of code and make it more understandable.

2.3.4 Performance

- While logging in and logging out, the processes should not take more than 5-10 seconds.
- Navigation between pages should happen fast enough to keep users in the system and should happen at the same speed on all platforms.

2.3.5 Extensibility

- Travela should be available on multiple platforms (web, mobile).
- Travela should work well with external APIs. (Skyscanner, Google Hotels, etc.)

2.3.6 Scalability

- It is aimed to reach many people so Travela should be scalable and handle data properly.

2.3.7 Accessibility

- Travela’s mobile app should be downloadable for free from the App Store or Google Play Store.
- Travelable should be easily visited through its URL.

2.3.8 Supportability

- For mobile, Travela should support iOS and Android operating systems.
- For web Travela should support Firefox, Chrome, Opera, Microsoft Edge, and Safari browsers.

2.4 Pseudo Requirements

- Application will be available on multiple platforms which are Web, Android and iOS.
- GitHub will be used for version control to track changes.
- Google Meets will be used for group meetings.
- Google Docs will be used for further reports.

2.5 System Models

2.5.1 Scenarios

1. Name: Sign up

2. Participating Actor: User

3. Entry Conditions: Web page is browsed or Travela is downloaded

4. Exit Conditions: Registration is completed or failed

5. Flow of Events:

5.1 User clicks the “Sign Up” button

5.2 User fills the registration form.

5.3 System checks the provided information if data is missing, invalid or irrelevant it gives an error, else it creates a new account.

5.4 User fills a questionnaire to mention his/her favorite location, preferred routes or activities etc.

---

1. Name: Login

2. Participating Actor: User

3. Entry Conditions: User should have a valid password and id

4. Exit Conditions: User logins to the system or the login fails

5. Flow of Events:

5.1 User clicks the “Login” button

5.2 User fills login information

5.3 System checks the provided information if data is missing, invalid or the user does not exist. The system gives an error .

---

1. Name: Form travel group

2. Participating Actor: User

3. Entry Conditions:

3.1 User should be logged in

3.2 User should already have friends

4. Exit Conditions: Invitations are accepted by users who received requests

5. Flow of Events:

5.1 User navigates to the Profile Page

5.2 User clicks “Form Group” button

5.3 User selects Friends to send an invitation and presses the “Send Invitation” button

5.4 User clicks “Close” button when he/she is done sending invitations

5.5 Invitations are accepted by users who received requests

---

1. Name: Integrate calendar (important dates) to the system

2. Participating Actor: User

3. Entry Conditions: User should be logged in

4. Exit Conditions: User integrates his/her calendar

5. Flow of Events:

User:

5.1 Switches to the profile page.

5.2 Clicks “Integrate Calendar” button.

5.3 Add suitable dates through the year to the system.

5.4 Clicks the “Close” button when integration is done.

---

1. Name: Arrange trip in a travel group

2. Participating Actor: Users in the group

3. Entry Conditions:

3.1 Users should be logged in to the system.

3.2 Users should be in the same travel group.

3.3 Users should already have added the suitable dates to their calendars.

4. Exit Conditions: Users are navigated to the related sites for purchasing tickets or reserve the determined place.

5. Flow of Events:

5.1 Users navigate to the profile page.

5.2 Users select the travel group that they want to have a trip with.

5.3 Each user views the common available dates of the group members via calendar integration and prefers one of the dates.

5.4 Users select a place among ML suggestions that is given by the system or according to their own poll results that is created by the users.

5.5 After selection of the place each user views and votes for the suitable hotels and transportation opportunities for the given available dates and place.

5.8 Users are navigated to the related sites for purchasing tickets or reserve the determined place.

---

1. Name: Send chat to your travel group

2. Participating Actor: Users in the group

3. Entry Conditions:

3.1 Users should be logged in

3.2 Users should be in the same travel group

4. Exit Conditions: Users are navigated to the related sites for purchasing tickets or reserve the determined place.

5. Flow of Events:

5.1 Users navigate to the profile page

5.2 Users select the travel group that they want to have a trip with

5.3 Users send a chat to that group about the trip arrangement

---

1. Name: Search places to visit

2. Participating Actor: User

3. Entry Conditions:

3.1 User should be logged in

4. Exit Conditions: "Search" button is pressed with filled textfield

5. Flow of Events:

User:

5.1 Writes a place he/she wishes to search in the TextField at the top of the Home page.

5.2 Press the "Search" button.

---

1. Name: View selected destination info

2. Participating Actor: User

3. Entry Conditions:

3.1 User should be logged in to the system.

3.2 User should be on the selected Destination page.

4. Exit Conditions: User closes the selected Destination page.

5. Flow of Events:

User:

5.1 Searches and selects a place he/wishes to view.

5.2 View Preferred season information of a destination.

5.3 Scrolls down and views the review and rating section.

5.4 Clicks the "Back" button when he/she is done viewing.

---

1. Name: View route and destination suggestions

2. Participating Actor: User

3. Entry Conditions:

3.1 User should be logged in to the system.

4. Exit Conditions: User closes the Home page.

5. Flow of Events:

5.1 User switches to Home page.

5.2 User sees the most suitable destination options in the Home page.

---

1. Name: View activity suggestions

2. Participating Actor: User

3. Entry Conditions:

3.1 User should be logged in.

4. Exit Conditions: User closes the Home page.

5. Flow of Events:

5.1 User switches to Home page.

5.2 User selects a destination.

5.5 User sees the popular activities that are done in that destination.

1. Name: Rate previous trip location

2. Participating Actor: User

3. Entry Conditions:

3.1 User should be logged into the system.

3.2 User should have a previous trip that is planned through Travela.

4. Exit Conditions: User rates the trip location.

5. Flow of Events:

User:

5.1 Switches to the Maps page.

5.2 Clicks to the trip that is seen in the World map.

5.3 Clicks to the “Rate” button.

5.4 Rates the trip location out of 5.

---

1. Name: Vote trip locations that is seen in Home page

2. Participating Actor: User

3. Entry Conditions: User should be logged into the system.

4. Exit Conditions: User rates the trip location.

5. Flow of Events:

User:

5.1 Switches to the Home page.

5.2 Clicks to the trip that is seen in the Home page.

5.3 Clicks to one of the “Dislike”, “Like” or “Love” buttons (like Netflix).

---

1. Name: See current travel groups

2. Participating Actor: User

3. Entry Conditions: User should be logged into the system.

4. Exit Conditions: User presses the “Logout” button or switches pages.

5. Flow of Events: User switches to the Profile page.

---

1. Name: Create polls among group

2. Participating Actor: User

3. Entry Conditions:

3.1 User should be logged in to the system.

3.2 User should be in a travel group.

4. Exit Conditions: Poll is created

5. Flow of Events:

User:

5.1 Switches to the Profile page.

5.2 Clicks to the travel group that he/she wants to create a poll.

5.3 Clicks to the “Create Poll” button.

5.4 Adds questions to the poll.

5.5 Clicks to the “Publish” button to send the poll to the group members.

---

1. Name: See upcoming trips

2. Participating Actor: User

3. Entry Conditions: User should be logged into the system.

4. Exit Conditions: User presses the “Logout” button or switch pages.

5. Flow of Events:

5.1 User switches to the Trips page.

5.2 User sees the upcoming trips that is planned by their travel groups.

---

1. Name: See previous trips of him/herself or his/her friends in a World map

2. Participating Actor: User

3. Entry Conditions:

3.1 User should be logged into the system.

3.2 User should be on the Maps page.

4. Exit Conditions: User presses the “Logout” button or switches pages.

5. Flow of Events:

5.1 User switches to the Maps page.

5.2 User views his/her map directly.

5.2 If the user wants to see their friends’ previous trips.

5.2.1 User views the friend list that is found in the Maps page.

5.2.2 User clicks to the friend name that has their own map.

5.3.3 User sees the related info (photo,review) of their friends.

---

1. Name: Send friend request

2. Participating Actor: User

3. Entry Conditions: User should be logged into the system.

4. Exit Conditions: Friend request is sent

5. Flow of Events:

5.1 User searches for a user name which the request will be sent.

5.2 User clicks the “Follow” button.

---

1. Name: Post photos belonging to the previous trips to the World map.

2. Participating Actor: User

3. Entry Conditions:

3.1 User should be logged into the system.

3.2 User should have a previous trip that is planned through Travela.

4. Exit Conditions: Trip (photos) is posted to the World map

5. Flow of Events:

User:

5.1 Switches to the Maps page.

5.2 Clicks to the trip that is seen in the World map.

5.3 Clicks to the “Add Photo” button.

5.4 Selects the photos to be posted.

5.5 Clicks “Post” button.

---

1. Name: Add a review of previous trips to the World map.

2. Participating Actor: User

3. Entry Conditions:

3.1 User should be logged into the system.

3.2 User should have a previous trip that is planned through Travela

4. Exit Conditions: Add review to trip

5. Flow of Events:

User:

5.1 Switches to the maps page

5.2 Clicks to the trip that is seen in the World map

5.3 Clicks to the “Add Review” button

5.4 Writes review

5.5 Clicks “Post” button

---

1. Name: Delete account

2. Participating Actor: User

3. Entry Conditions:

:3.1 User should be logged into the system.

3.2 User should have an active account.

4. Exit Conditions: User deletes the account.

5. Flow of Events:

User:

5.1 Switches to the profile page.

5.2 Clicks to the “Settings” button.

5.3 Clicks to the “Account” button.

5.4 Clicks to the “Delete Account” button.

5.5 Confirms to delete the account.

---

1. Name: Participate in questionnaire

2. Participating Actor: User

3. Entry Conditions: User should be logged into the system.

4. Exit Conditions: User completes the questionnaire.

5. Flow of Events:

User:

5.1 Switches to the profile page.

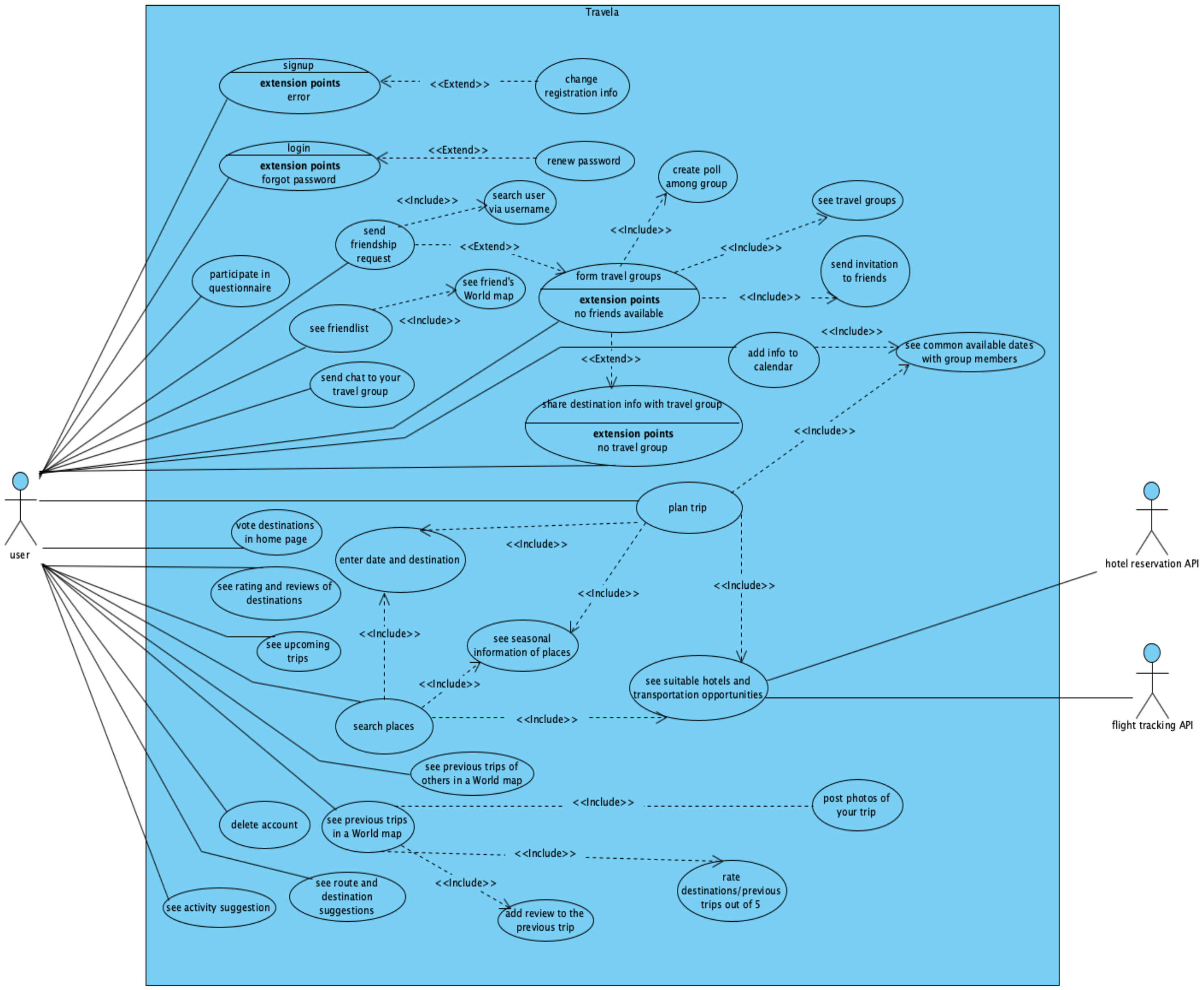
5.2 Clicks to the “Questionnaire” button.

5.3 Completes the questionnaire.

5.4 Clicks to the “Send” button.

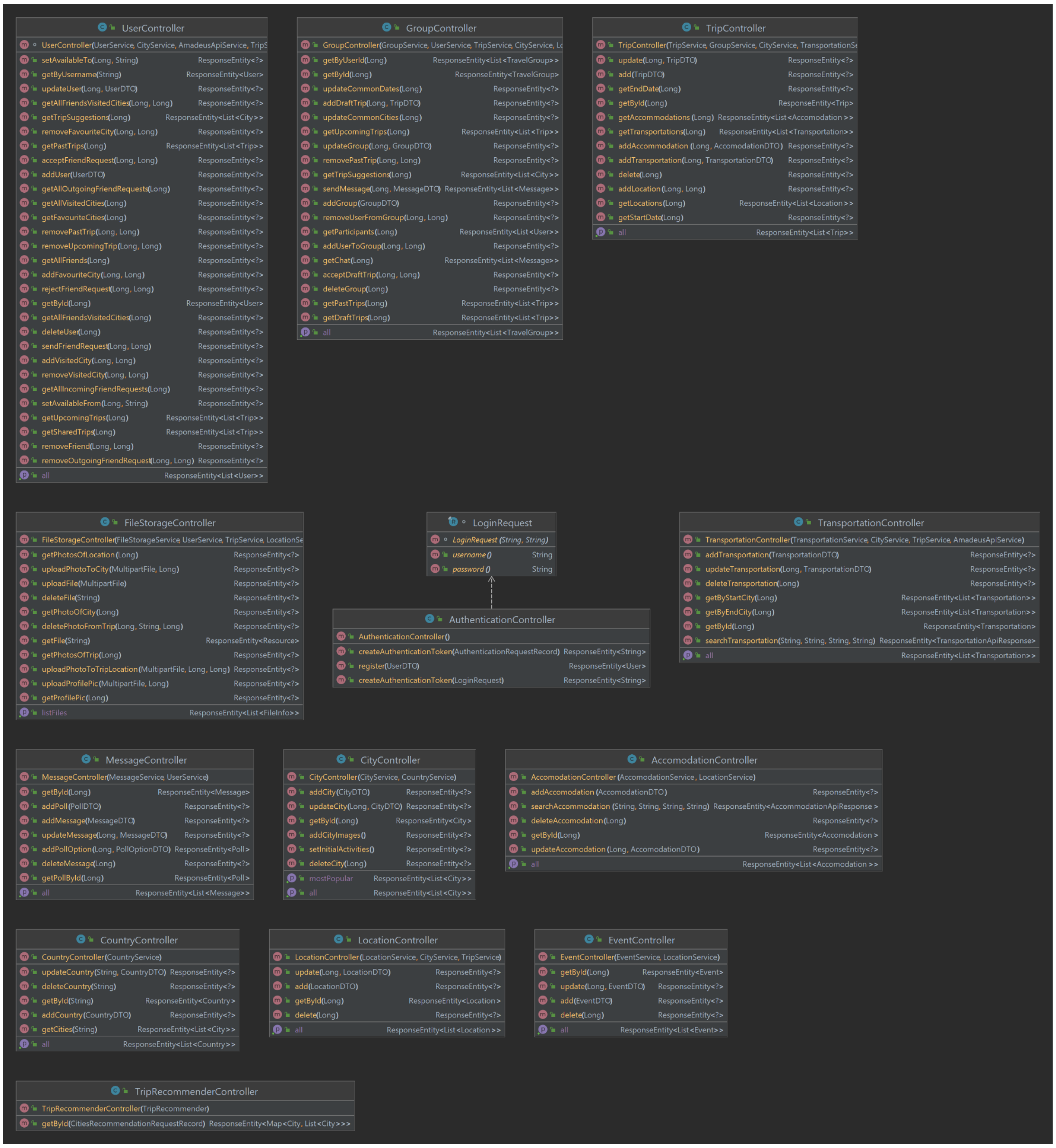
3 Final Architecture and Design Details

3.1.1 Use-Case Model



3.1.2 Object and Class Model

3.1.2.1 Controllers



UserController	
UserController(UserService, CityService, AmadeusApiService, TripS	
setAvailableTo(Long, String)	ResponseEntity<?>
getByUsername(String)	ResponseEntity<User>
updateUser(Long, UserDTO)	ResponseEntity<?>
getAllFriendsVisitedCities(Long, Long)	ResponseEntity<?>
getTripSuggestions(Long)	ResponseEntity<List<City>>
removeFavouriteCity(Long, Long)	ResponseEntity<?>
getPastTrips(Long)	ResponseEntity<List<Trip>>
acceptFriendRequest(Long, Long)	ResponseEntity<?>
addUser(UserDTO)	ResponseEntity<?>
getAllOutgoingFriendRequests(Long)	ResponseEntity<?>
getAllVisitedCities(Long)	ResponseEntity<?>
getFavouriteCities(Long)	ResponseEntity<?>
removePastTrip(Long, Long)	ResponseEntity<?>
removeUpcomingTrip(Long, Long)	ResponseEntity<?>
getAllFriends(Long)	ResponseEntity<?>
addFavouriteCity(Long, Long)	ResponseEntity<?>
rejectFriendRequest(Long, Long)	ResponseEntity<?>
getById(Long)	ResponseEntity<User>
getAllFriendsVisitedCities(Long)	ResponseEntity<?>
deleteUser(Long)	ResponseEntity<?>
sendFriendRequest(Long, Long)	ResponseEntity<?>
addVisitedCity(Long, Long)	ResponseEntity<?>
removeVisitedCity(Long, Long)	ResponseEntity<?>
getAllIncomingFriendRequests(Long)	ResponseEntity<?>
setAvailableFrom(Long, String)	ResponseEntity<?>
getUpcomingTrips(Long)	ResponseEntity<List<Trip>>
getSharedTrips(Long)	ResponseEntity<List<Trip>>
removeFriend(Long, Long)	ResponseEntity<?>
removeOutgoingFriendRequest(Long, Long)	ResponseEntity<?>
all	ResponseEntity<List<User>>

GroupController	
GroupController(GroupService, UserService, TripService, CityService, Lo	
getByUserId(Long)	ResponseEntity<List<TravelGroup>>
getById(Long)	ResponseEntity<TravelGroup>
updateCommonDates(Long)	ResponseEntity<?>
addDraftTrip(Long, TripDTO)	ResponseEntity<?>
updateCommonCities(Long)	ResponseEntity<?>
getUpcomingTrips(Long)	ResponseEntity<List<Trip>>
updateGroup(Long, GroupDTO)	ResponseEntity<?>
removePastTrip(Long, Long)	ResponseEntity<?>
getTripSuggestions(Long)	ResponseEntity<List<City>>
sendMessage(Long, MessageDTO)	ResponseEntity<List<Message>>
addGroup(GroupDTO)	ResponseEntity<?>
removeUserFromGroup(Long, Long)	ResponseEntity<?>
getParticipants(Long)	ResponseEntity<List<User>>
addUserToGroup(Long, Long)	ResponseEntity<?>
getChat(Long)	ResponseEntity<List<Message>>
acceptDraftTrip(Long, Long)	ResponseEntity<?>
deleteGroup(Long)	ResponseEntity<?>
getPastTrips(Long)	ResponseEntity<List<Trip>>
getDraftTrips(Long)	ResponseEntity<List<Trip>>
all	ResponseEntity<List<TravelGroup>>

TripController	
TripController(TripService, GroupService, CityService, TransportationSe	
update(Long, TripDTO)	ResponseEntity<?>
add(TripDTO)	ResponseEntity<?>
getEndDate(Long)	ResponseEntity<?>
getById(Long)	ResponseEntity<Trip>
getAccommodations (Long)	ResponseEntity<List<Accomodation>>
getTransportations(Long)	ResponseEntity<List<Transportation>>
addAccommodation (Long, AccomodationDTO)	ResponseEntity<?>
addTransportation(Long, TransportationDTO)	ResponseEntity<?>
delete(Long)	ResponseEntity<?>
addLocation (Long, Long)	ResponseEntity<?>
getLocations (Long)	ResponseEntity<List<Location>>
getStartDate(Long)	ResponseEntity<?>
all	ResponseEntity<List<Trip>>

FileStorageController	
FileStorageController(FileStorageService, UserService, TripService, LocationSe	
getPhotosOfLocation (Long)	ResponseEntity<?>
uploadPhotoToCity (MultipartFile, Long)	ResponseEntity<?>
uploadFile(MultipartFile)	ResponseEntity<?>
deleteFile(String)	ResponseEntity<?>
getPhotoOfCity (Long)	ResponseEntity<?>
deletePhotoFromTrip(Long, String, Long)	ResponseEntity<?>
getFile(String)	ResponseEntity<Resource>
getPhotosOfTrip(Long)	ResponseEntity<?>
uploadPhotoToTripLocation (MultipartFile, Long, Long)	ResponseEntity<?>
uploadProfilePic (MultipartFile, Long)	ResponseEntity<?>
getProfilePic (Long)	ResponseEntity<?>
listFiles	ResponseEntity<List<FileInfo>>

LoginRequest	
LoginRequest (String, String)	
username ()	String
password ()	String

AuthenticationController	
AuthenticationController()	
createAuthenticationToken(AuthenticationRequestRecord)	ResponseEntity<String>
register(UserDTO)	ResponseEntity<User>
createAuthenticationToken(LoginRequest)	ResponseEntity<String>

TransportationController	
TransportationController(TransportationService, CityService, TripService, AmadeusApiService)	
addTransportation(TransportationDTO)	ResponseEntity<?>
updateTransportation(Long, TransportationDTO)	ResponseEntity<?>
deleteTransportation(Long)	ResponseEntity<?>
getByStartCity(Long)	ResponseEntity<List<Transportation>>
getByEndCity(Long)	ResponseEntity<List<Transportation>>
getById(Long)	ResponseEntity<Transportation>
searchTransportation(String, String, String, String)	ResponseEntity<TransportationApiResponse>
all	ResponseEntity<List<Transportation>>

MessageController	
MessageController(MessageService, UserService)	
getById(Long)	ResponseEntity<Message>
addPoll (PollDTO)	ResponseEntity<?>
addMessage(MessageDTO)	ResponseEntity<?>
updateMessage(Long, MessageDTO)	ResponseEntity<?>
addPollOption(Long, PollOptionDTO)	ResponseEntity<Poll>
deleteMessage(Long)	ResponseEntity<?>
getPollById(Long)	ResponseEntity<Poll>
all	ResponseEntity<List<Message>>

CityController	
CityController(CityService, CountryService)	
addCity(CityDTO)	ResponseEntity<?>
updateCity(Long, CityDTO)	ResponseEntity<?>
getById(Long)	ResponseEntity<City>
addCityImages()	ResponseEntity<?>
setInitialActivities()	ResponseEntity<?>
deleteCity(Long)	ResponseEntity<?>
mostPopular	ResponseEntity<List<City>>
all	ResponseEntity<List<City>>

AccomodationController	
AccomodationController (AccomodationService, LocationService)	
addAccomodation (AccomodationDTO)	ResponseEntity<?>
searchAccommodation (String, String, String, String)	ResponseEntity<AccommodationApiResponse>
deleteAccomodation(Long)	ResponseEntity<?>
getById(Long)	ResponseEntity<Accomodation>
updateAccomodation (Long, AccomodationDTO)	ResponseEntity<?>
all	ResponseEntity<List<Accomodation>>

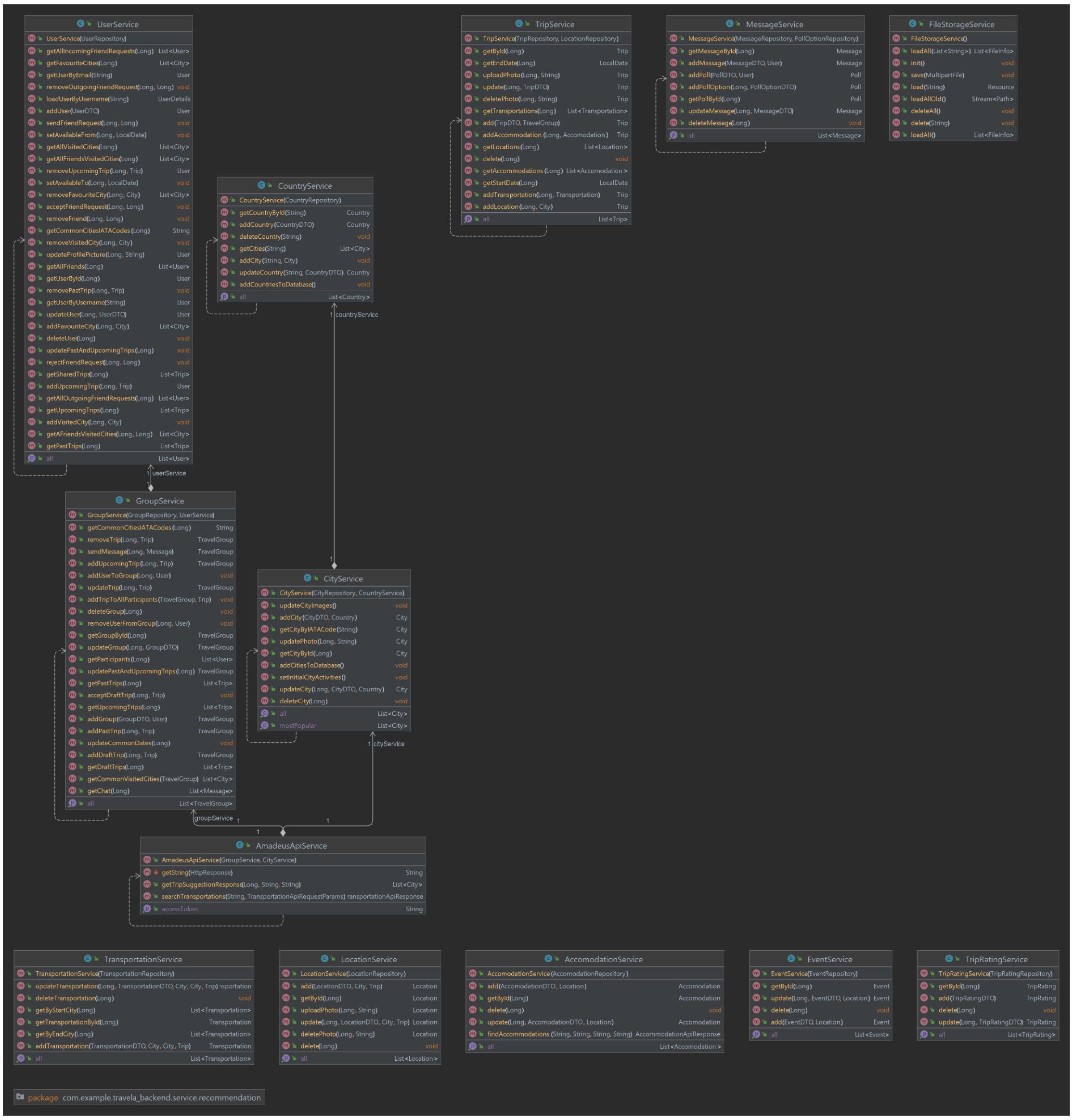
CountryController	
CountryController(CountryService)	
updateCountry(String, CountryDTO)	ResponseEntity<?>
deleteCountry(String)	ResponseEntity<?>
getById(String)	ResponseEntity<Country>
addCountry(CountryDTO)	ResponseEntity<?>
getCities(String)	ResponseEntity<List<City>>
all	ResponseEntity<List<Country>>

LocationController	
LocationController(LocationService, CityService, TripService)	
update(Long, LocationDTO)	ResponseEntity<?>
add(LocationDTO)	ResponseEntity<?>
getById(Long)	ResponseEntity<Location>
delete(Long)	ResponseEntity<?>
all	ResponseEntity<List<Location>>

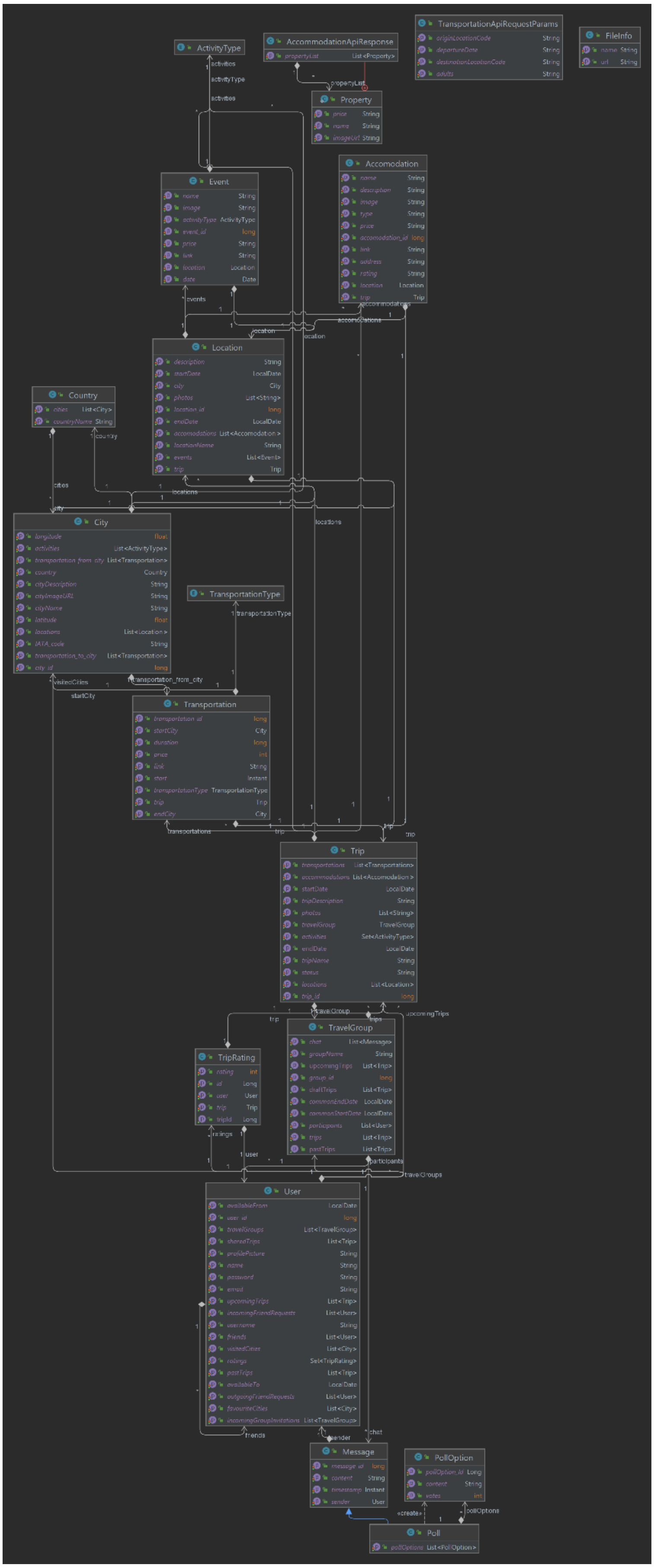
EventController	
EventController(EventService, LocationService)	
getById(Long)	ResponseEntity<Event>
update(Long, EventDTO)	ResponseEntity<?>
add(EventDTO)	ResponseEntity<?>
delete(Long)	ResponseEntity<?>
all	ResponseEntity<List<Event>>

TripRecommenderController	
TripRecommenderController(TripRecommender)	
getById(CitiesRecommendationRequestRecord)	ResponseEntity<Map<City, List<City>>>

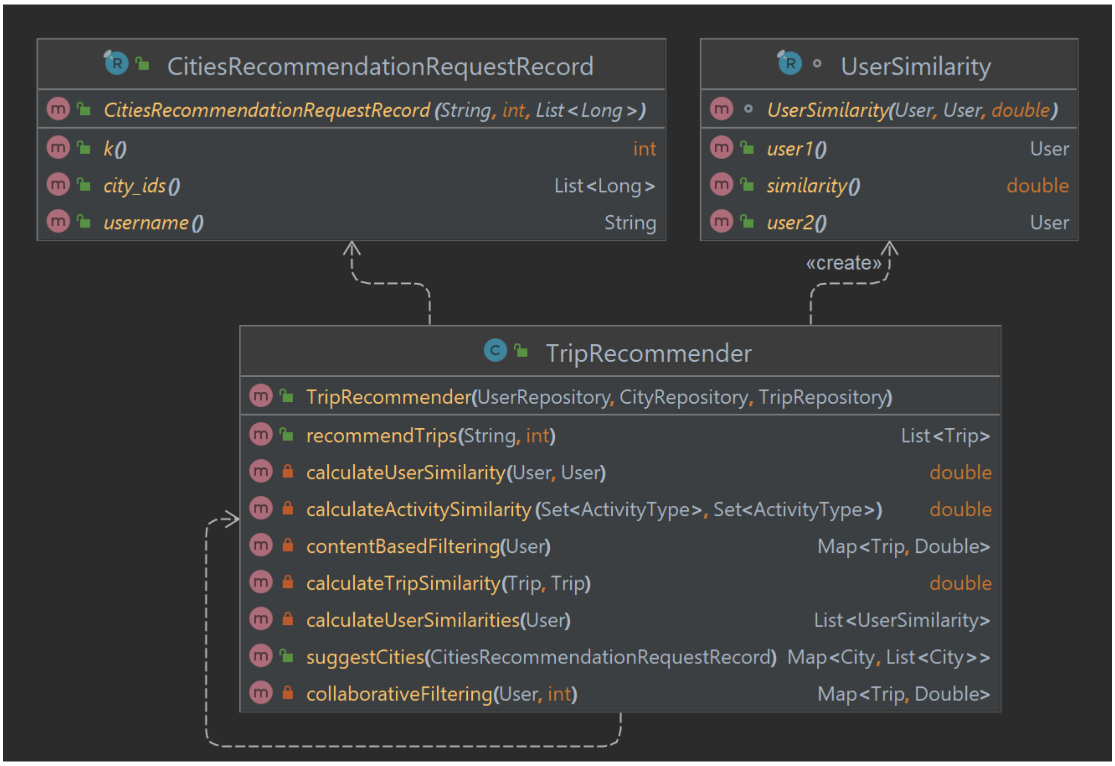
3.1.2.2 Services


UserService
UserService(UserRepository)
getAllIncomingFriendRequests(Long) List<User>
getFavouriteCities(Long) List<City>
getUserByEmail(String) User
removeOutgoingFriendRequest(Long, Long) void
loadUserByUsername(String) UserDetails
addUser(UserDTO) User
sendFriendRequest(Long, Long) void
setAvailableFrom(Long, LocalDate) void
getAllVisitedCities(Long) List<City>
getAllFriendsVisitedCities(Long) List<City>
removeUpcomingTrip(Long, Trip) User
setAvailableTo(Long, LocalDate) void
removeFavouriteCity(Long, City) List<City>
acceptFriendRequest(Long, Long) void
removeFriend(Long, Long) void
getCommonCitiesIATACodes (Long) String
removeVisitedCity(Long, City) void
updateProfilePicture(Long, String) User
getAllFriends(Long) List<User>
getUserById(Long) User
removePastTrip(Long, Trip) void
getUserByUsername(String) User
updateUser(Long, UserDTO) User
addFavouriteCity(Long, City) List<City>
deleteUser(Long) void
updatePastAndUpcomingTrips (Long) void
rejectFriendRequest(Long, Long) void
getSharedTrips(Long) List<Trip>
addUpcomingTrip (Long, Trip) User
getAllOutgoingFriendRequests(Long) List<User>
getUpcomingTrips(Long) List<Trip>
addVisitedCity(Long, City) void
getAFriendsVisitedCities(Long, Long) List<City>
getPastTrips(Long) List<Trip>
all List<User>
1 userService
1
CountryService
CountryService(CountryRepository)
getCountryById(String) Country
addCountry(CountryDTO) Country
deleteCountry(String) void
getCities(String) List<City>
addCity(String, City) void
updateCountry(String, CountryDTO) Country
addCountriesToDatabase() void
all List<Country>
1 countryService
1
TripService
TripService(TripRepository, LocationRepository)
getById(Long) Trip
getEndDate(Long) LocalDate
uploadPhoto (Long, String) Trip
update(Long, TripDTO) Trip
deletePhoto(Long, String) Trip
getTransportations(Long) List<Transportation>
add(TripDTO, TravelGroup) Trip
addAccommodation (Long, Accomodation) Trip
getLocations (Long) List<Location >
delete(Long) void
getAccommodations (Long) List<Accomodation >
getStartDate(Long) LocalDate
addTransportation(Long, Transportation) Trip
addLocation (Long, City) Trip
all List<Trip>
MessageService
MessageService(MessageRepository, PollOptionRepository)
getMessageById(Long) Message
addMessage(MessageDTO, User) Message
addPoll(PollDTO, User) Poll
addPollOption(Long, PollOptionDTO) Poll
getPollById(Long) Poll
updateMessage(Long, MessageDTO) Message
deleteMessage(Long) void
all List<Message>
FileStorageService
FileStorageService()
loadAll(List<String>) List<FileInfo>
init() void
save(MultipartFile) void
load(String) Resource
loadAllOld() Stream<Path>
deleteAll() void
delete(String) void
loadAll() List<FileInfo>
GroupService
GroupService(GroupRepository, UserService)
getCommonCitiesIATACodes (Long) String
removeTrip(Long, Trip) TravelGroup
sendMessage(Long, Message) TravelGroup
addUpcomingTrip (Long, Trip) TravelGroup
addUserToGroup(Long, User) void
updateTrip(Long, Trip) TravelGroup
addTripToAllParticipants (TravelGroup, Trip) void
deleteGroup(Long) void
removeUserFromGroup(Long, User) void
getGroupById(Long) TravelGroup
updateGroup(Long, GroupDTO) TravelGroup
getParticipants (Long) List<User>
updatePastAndUpcomingTrips (Long) TravelGroup
getPastTrips(Long) List<Trip>
acceptDraftTrip(Long, Trip) void
getUpcomingTrips(Long) List<Trip>
addGroup(GroupDTO, User) TravelGroup
addPastTrip(Long, Trip) TravelGroup
updateCommonDates(Long) void
addDraftTrip(Long, Trip) TravelGroup
getDraftTrips(Long) List<Trip>
getCommonVisitedCities(TravelGroup) List<City>
getChat(Long) List<Message>
all List<TravelGroup>
groupService 1
CityService
CityService(CityRepository, CountryService)
updateCityImages() void
addCity (CityDTO, Country) City
getCityByIATACode (String) City
updatePhoto(Long, String) City
getCityById(Long) City
addCitiesToDatabase() void
setInitialCityActivities() void
updateCity(Long, CityDTO, Country) City
deleteCity(Long) void
all List<City>
mostPopular List<City>
1 cityService
1
1
AmadeusApiService
AmadeusApiService(GroupService, CityService)
getString(HttpResponse) String
getTripSuggestionResponse(Long, String, String) List<City>
searchTransportations(String, TransportationApiRequestParams) ransportationApiResponse
accessToken String
TransportationService
TransportationService(TransportationRepository)
updateTransportation(Long, TransportationDTO, City, City, Trip) ısportation
deleteTransportation(Long) void
getByStartCity(Long) List<Transportation>
getTransportationById(Long) Transportation
getByEndCity(Long) List<Transportation>
addTransportation(TransportationDTO, City, City, Trip) Transportation
all List<Transportation>
LocationService
LocationService(LocationRepository)
add(LocationDTO, City, Trip) Location
getById(Long) Location
uploadPhoto (Long, String) Location
update(Long, LocationDTO, City, Trip) Location
deletePhoto(Long, String) Location
delete(Long) void
all List<Location >
AccomodationService
AccomodationService (AccomodationRepository)
add(AccomodationDTO, Location) Accomodation
getById(Long) Accomodation
delete(Long) void
update(Long, AccomodationDTO, Location) Accomodation
findAccommodations (String, String, String, String) AccommodationApiResponse
all List<Accomodation >
EventService
EventService(EventRepository)
getById(Long) Event
update(Long, EventDTO, Location) Event
delete(Long) void
add(EventDTO, Location) Event
all List<Event>
TripRatingService
TripRatingService(TripRatingRepository)
getById(Long) TripRating
add(TripRatingDTO) TripRating
delete(Long) void
update(Long, TripRatingDTO) TripRating
all List<TripRating>
package com.example.travela_backend.service.recommendation

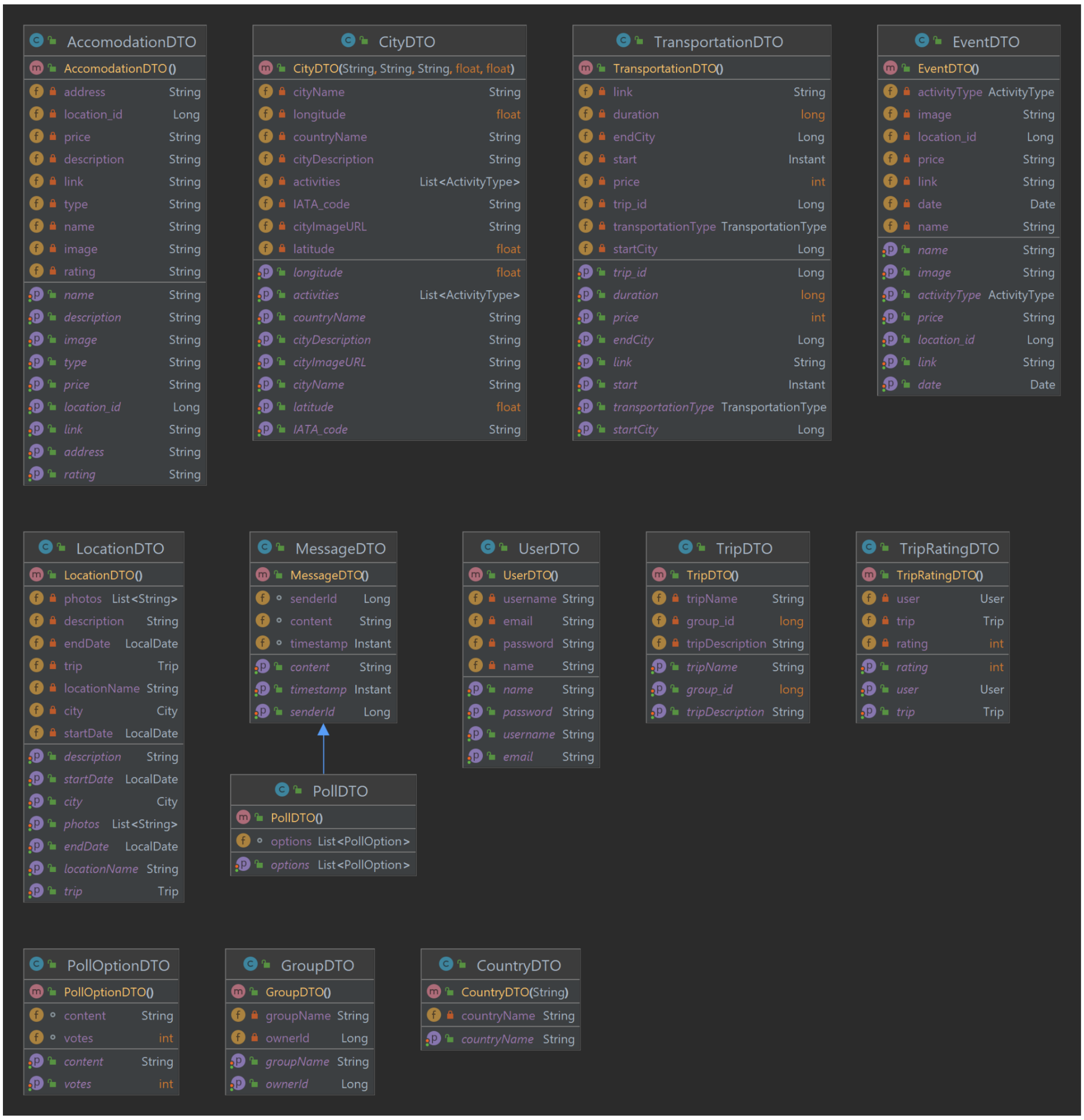
3.1.2.3 Entities



3.1.2.4 Recommendation Logic

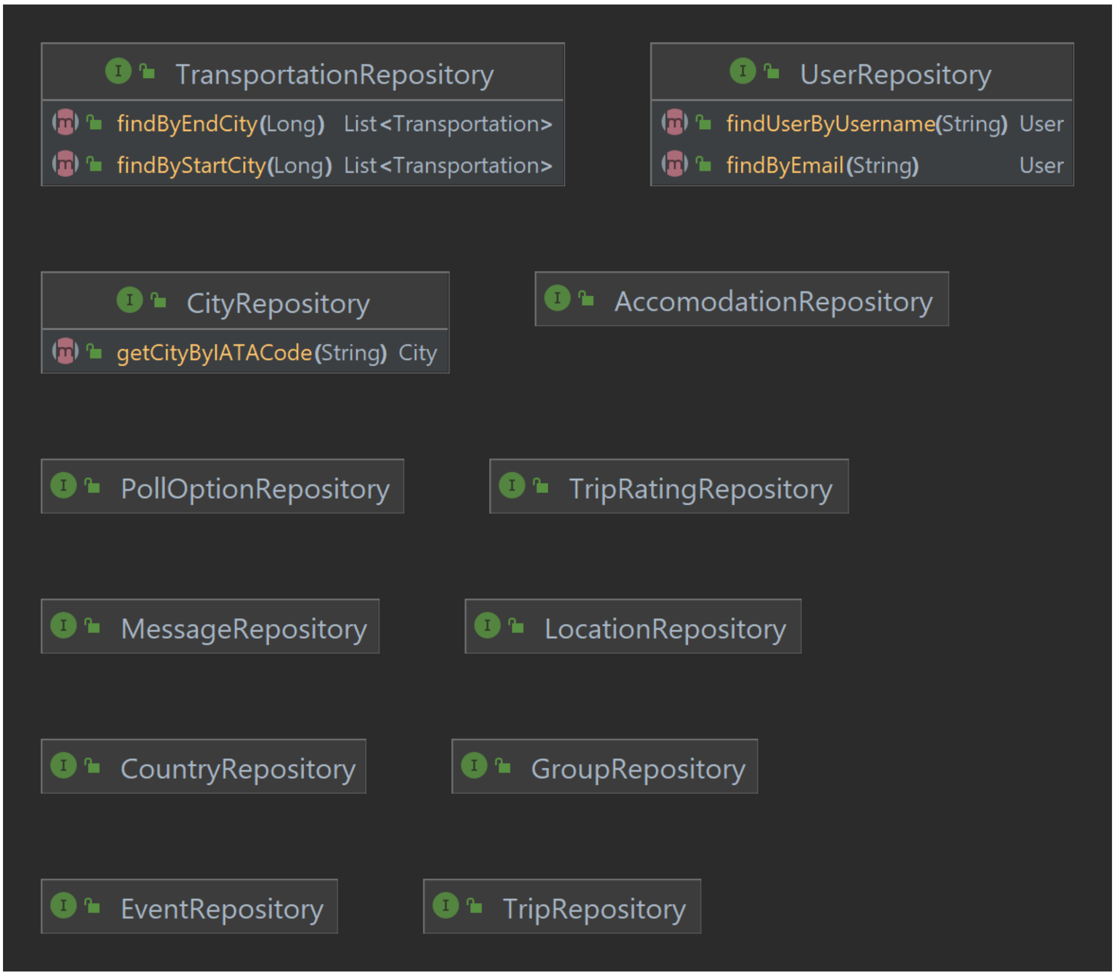

CitiesRecommendationRequestRecord
CitiesRecommendationRequestRecord (String, int, List<Long>)
k() int
city_ids() List<Long>
username() String
UserSimilarity
UserSimilarity(User, User, double)
user1() User
similarity() double
user2() User
«create»
TripRecommender
TripRecommender(UserRepository, CityRepository, TripRepository)
recommendTrips(String, int) List<Trip>
calculateUserSimilarity(User, User) double
calculateActivitySimilarity (Set<ActivityType>, Set<ActivityType>) double
contentBasedFiltering(User) Map<Trip, Double>
calculateTripSimilarity(Trip, Trip) double
calculateUserSimilarities(User) List<UserSimilarity>
suggestCities(CitiesRecommendationRequestRecord) Map<City, List<City>>
collaborativeFiltering(User, int) Map<Trip, Double>

3.1.2.5 DTOs

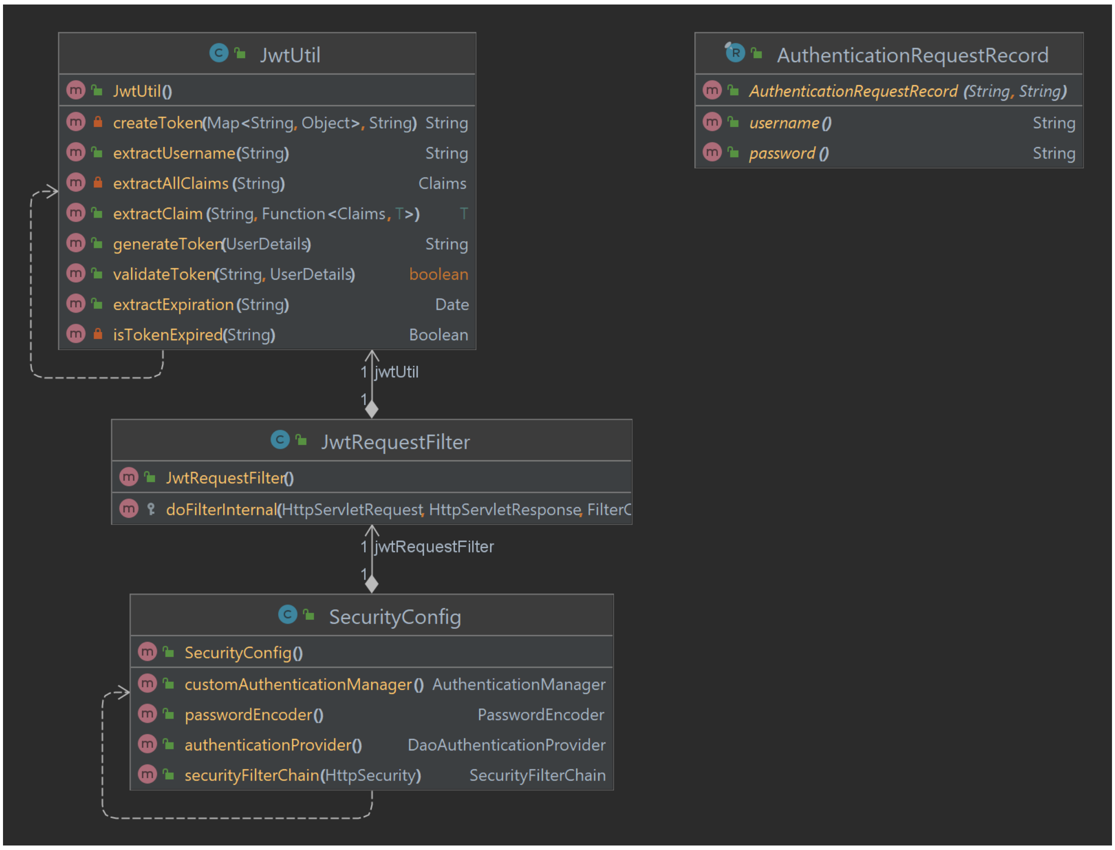

AccomodationDTO
AccomodationDTO()
address String
location_id Long
price String
description String
link String
type String
name String
image String
rating String
name String
description String
image String
type String
price String
location_id Long
link String
address String
rating String
CityDTO
CityDTO(String, String, String, float, float)
cityName String
longitude float
countryName String
cityDescription String
activities List<ActivityType>
IATA_code String
cityImageURL String
latitude float
longitude float
activities List<ActivityType>
countryName String
cityDescription String
cityImageURL String
cityName String
latitude float
IATA_code String
TransportationDTO
TransportationDTO()
link String
duration long
endCity Long
start Instant
price int
trip_id Long
transportationType TransportationType
startCity Long
trip_id Long
duration long
price int
endCity Long
link String
start Instant
transportationType TransportationType
startCity Long
EventDTO
EventDTO()
activityType ActivityType
image String
location_id Long
price String
link String
date Date
name String
name String
image String
activityType ActivityType
price String
location_id Long
link String
date Date
LocationDTO
LocationDTO()
photos List<String>
description String
endDate LocalDate
trip Trip
locationName String
city City
startDate LocalDate
description String
startDate LocalDate
city City
photos List<String>
endDate LocalDate
locationName String
trip Trip
MessageDTO
MessageDTO()
senderId Long
content String
timestamp Instant
content String
timestamp Instant
senderId Long
PollDTO
PollDTO()
options List<PollOption>
options List<PollOption>
UserDTO
UserDTO()
username String
email String
password String
name String
name String
password String
username String
email String
TripDTO
TripDTO()
tripName String
group_id long
tripDescription String
tripName String
group_id long
tripDescription String
TripRatingDTO
TripRatingDTO()
user User
trip Trip
rating int
rating int
user User
trip Trip
PollOptionDTO
PollOptionDTO()
content String
votes int
content String
votes int
GroupDTO
GroupDTO()
groupName String
ownerId Long
groupName String
ownerId Long
CountryDTO
CountryDTO(String)
countryName String
countryName String

3.1.2.6 Repositories

These repositories extend base JPARepository



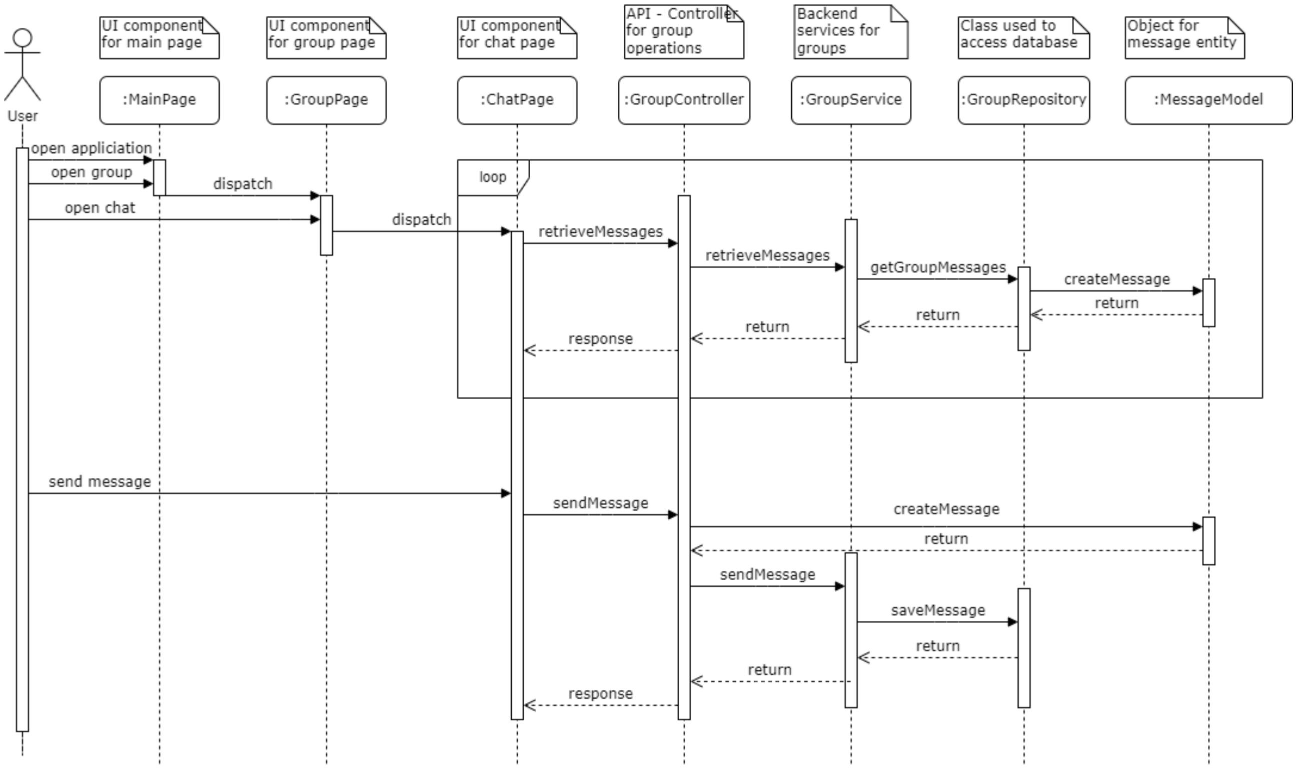
3.1.2.7 Security


JwtUtil
JwtUtil()
createToken(Map<String, Object>, String) String
extractUsername(String) String
extractAllClaims (String) Claims
extractClaim (String, Function<Claims, T>) T
generateToken(UserDetails) String
validateToken(String, UserDetails) boolean
extractExpiration (String) Date
isTokenExpired(String) Boolean
AuthenticationRequestRecord
AuthenticationRequestRecord (String, String)
username () String
password () String
1 jwtUtil
1
JwtRequestFilter
JwtRequestFilter()
doFilterInternal(HttpServletRequest, HttpServletResponse, FilterC
1 jwtRequestFilter
1
SecurityConfig
SecurityConfig()
customAuthenticationManager() AuthenticationManager
passwordEncoder() PasswordEncoder
authenticationProvider() DaoAuthenticationProvider
securityFilterChain(HttpSecurity) SecurityFilterChain

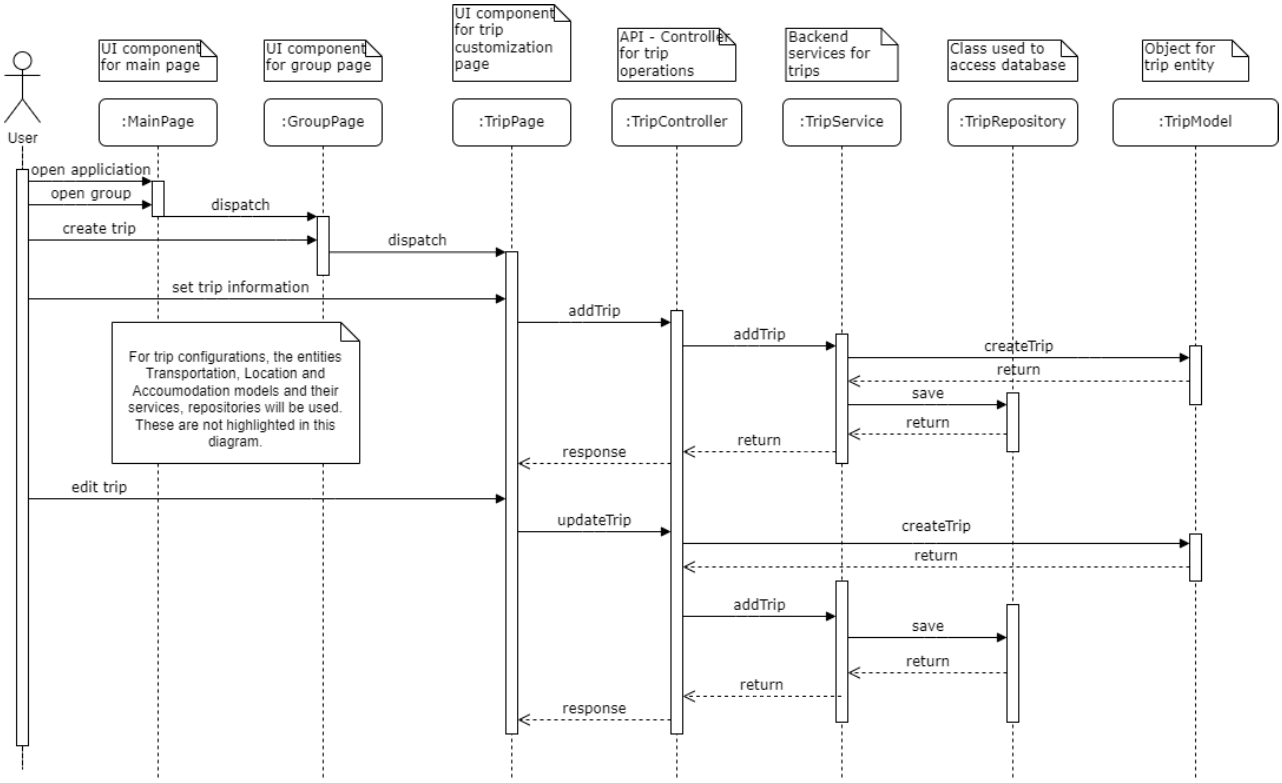
3.1.3 Dynamic Models

3.1.3.1 Sequence Diagrams

Open travel group chat and send messages

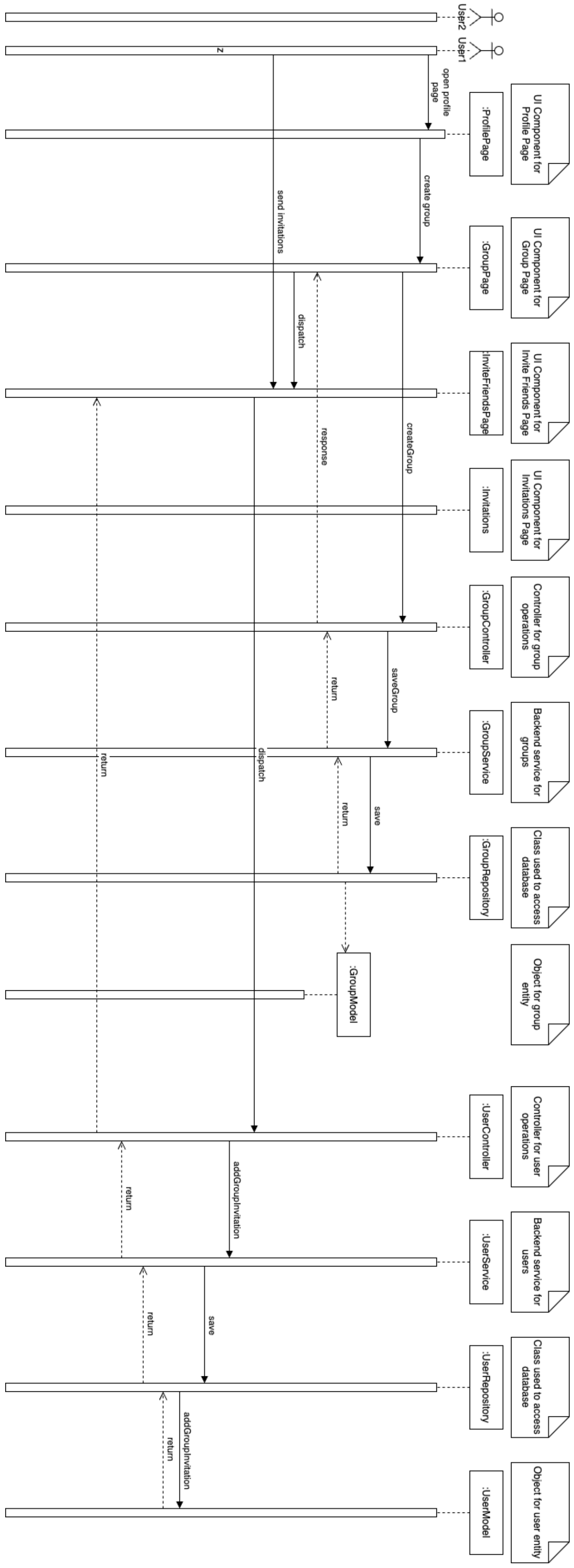


Arrange trip and edit trip

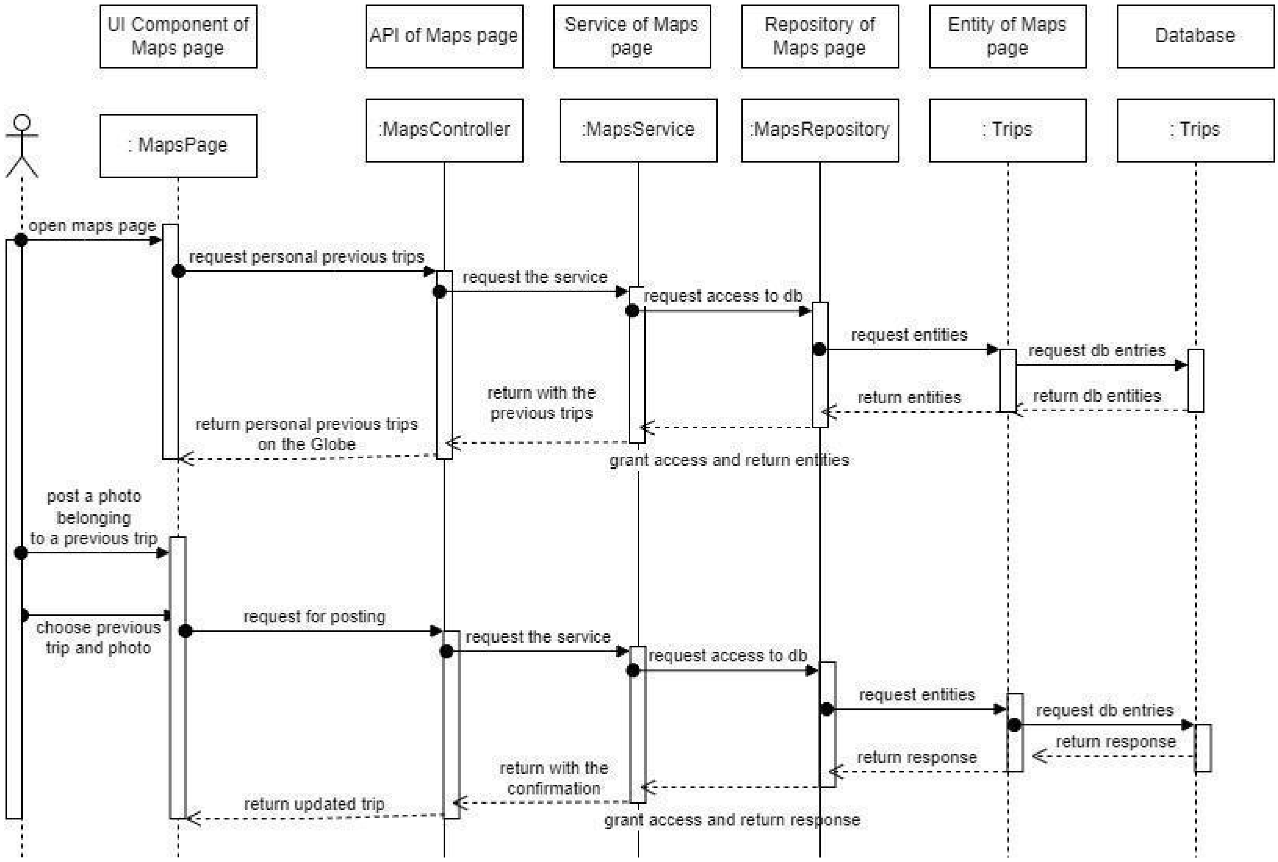


For trip configurations, the entities Transportation, Location and Accoumodation models and their services, repositories will be used. These are not highlighted in this diagram.

Create new group and invite friends

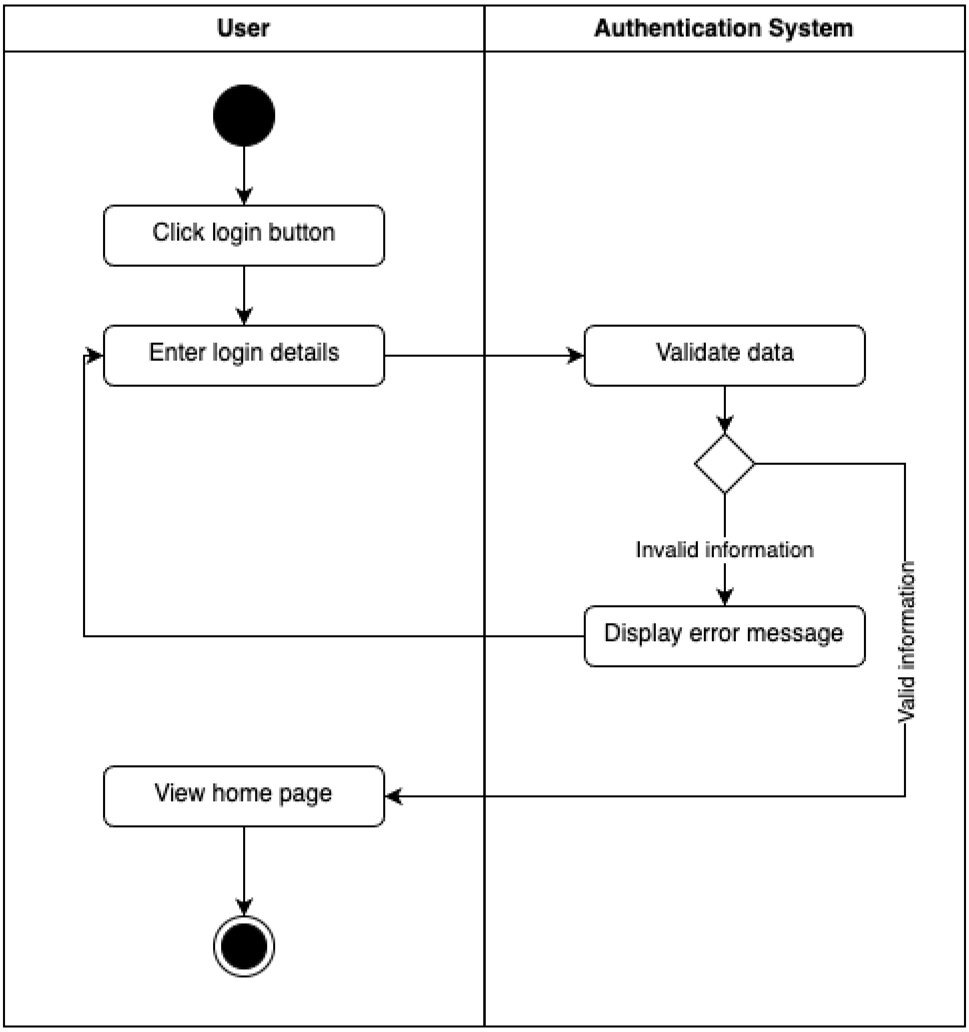


View the previous trips on globe and post photos belonging to previous trips

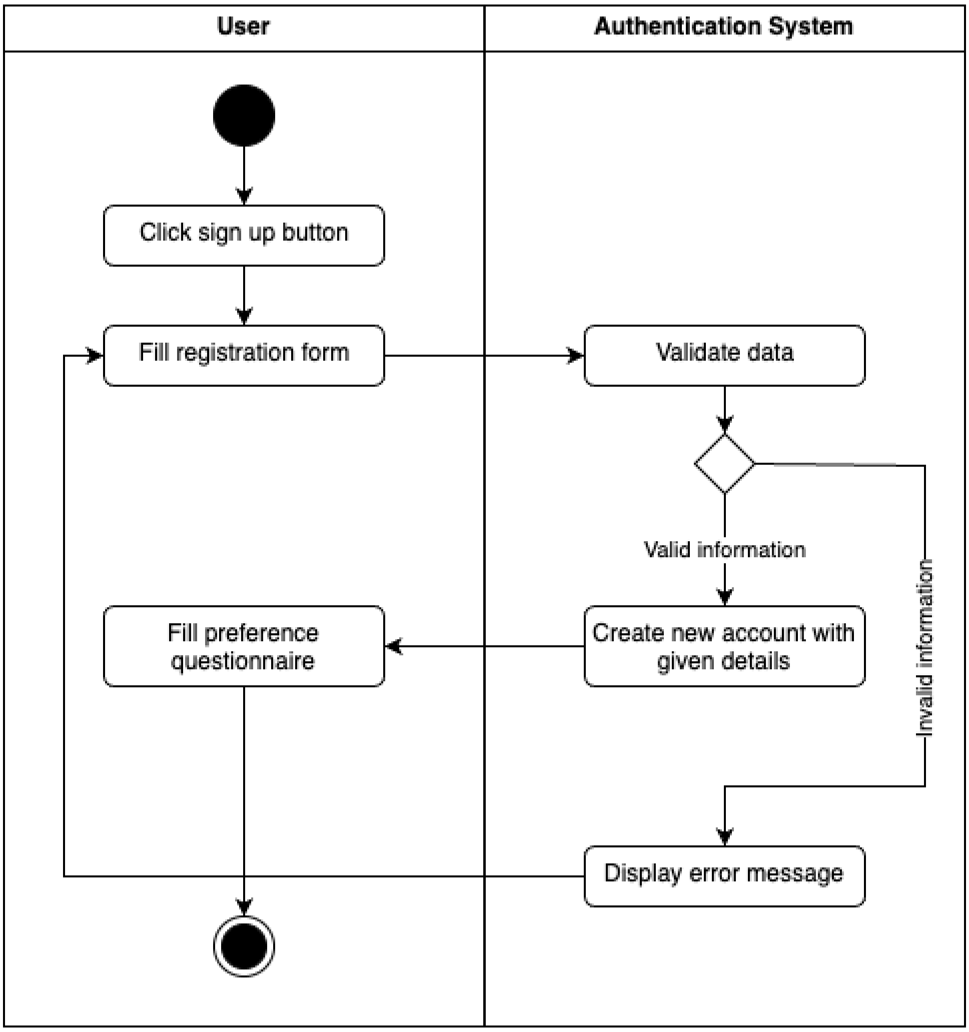


3.1.3.2 Activity Diagrams

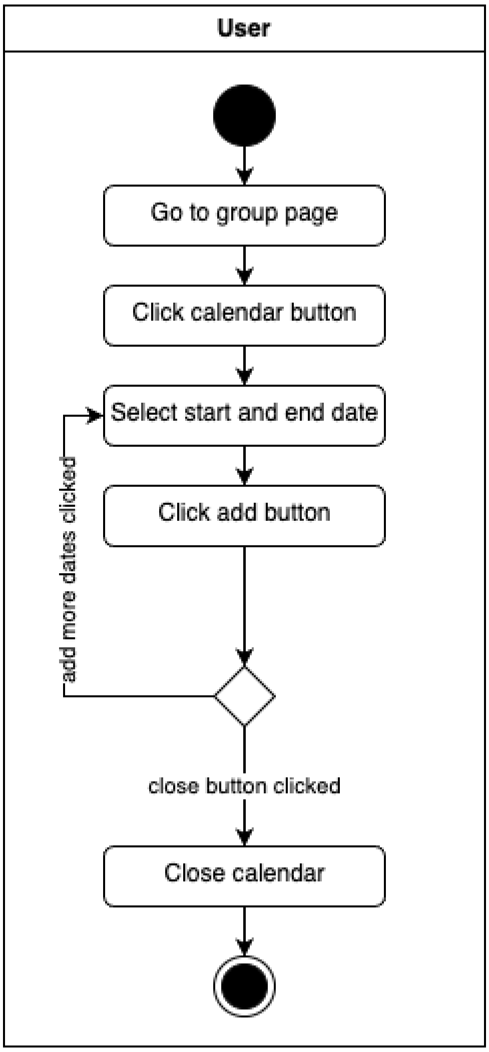
Login



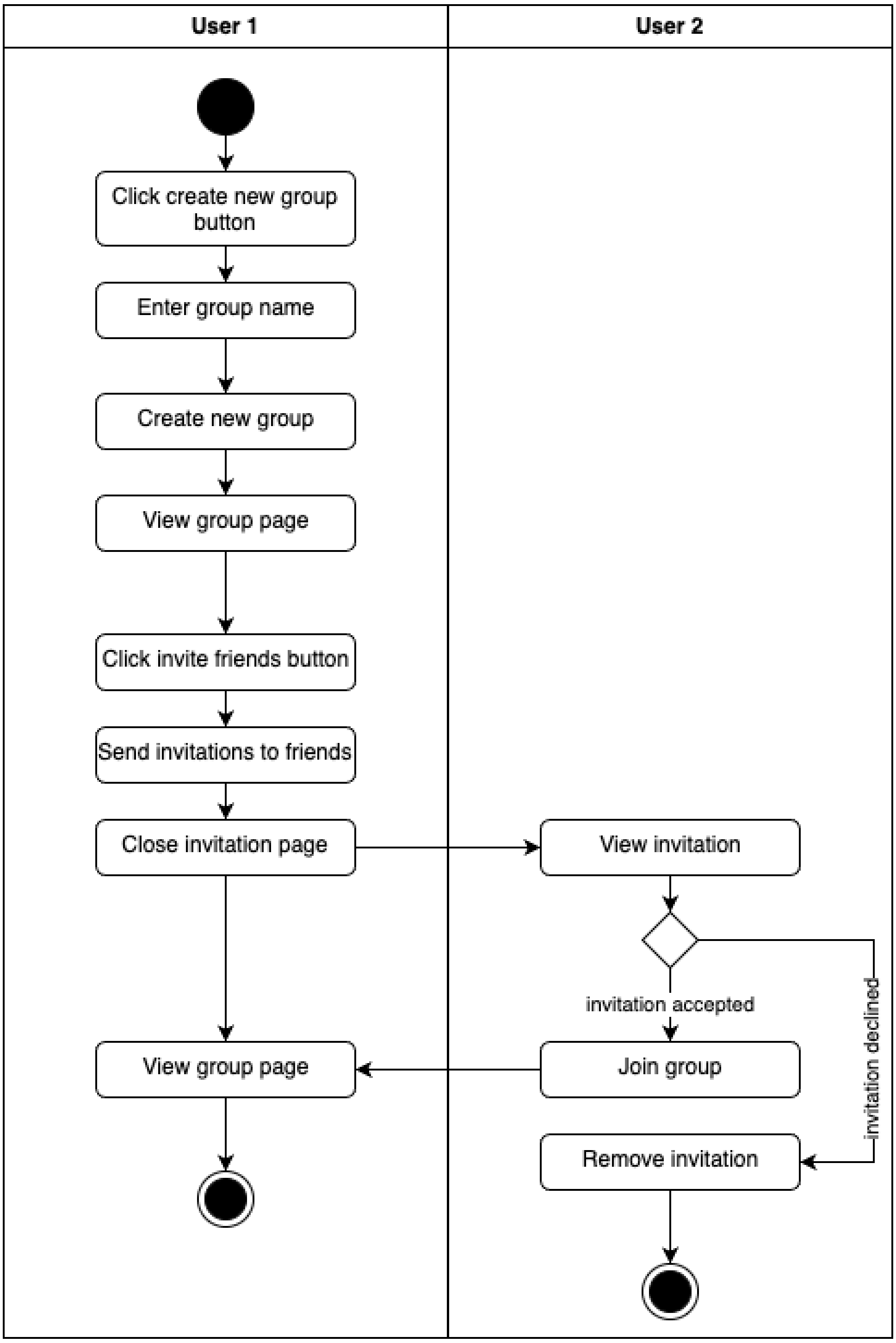
Signup


User
Authentication System
Click sign up button
Fill registration form
Validate data
Valid information
Invalid information
Create new account with given details
Fill preference questionnaire
Display error message

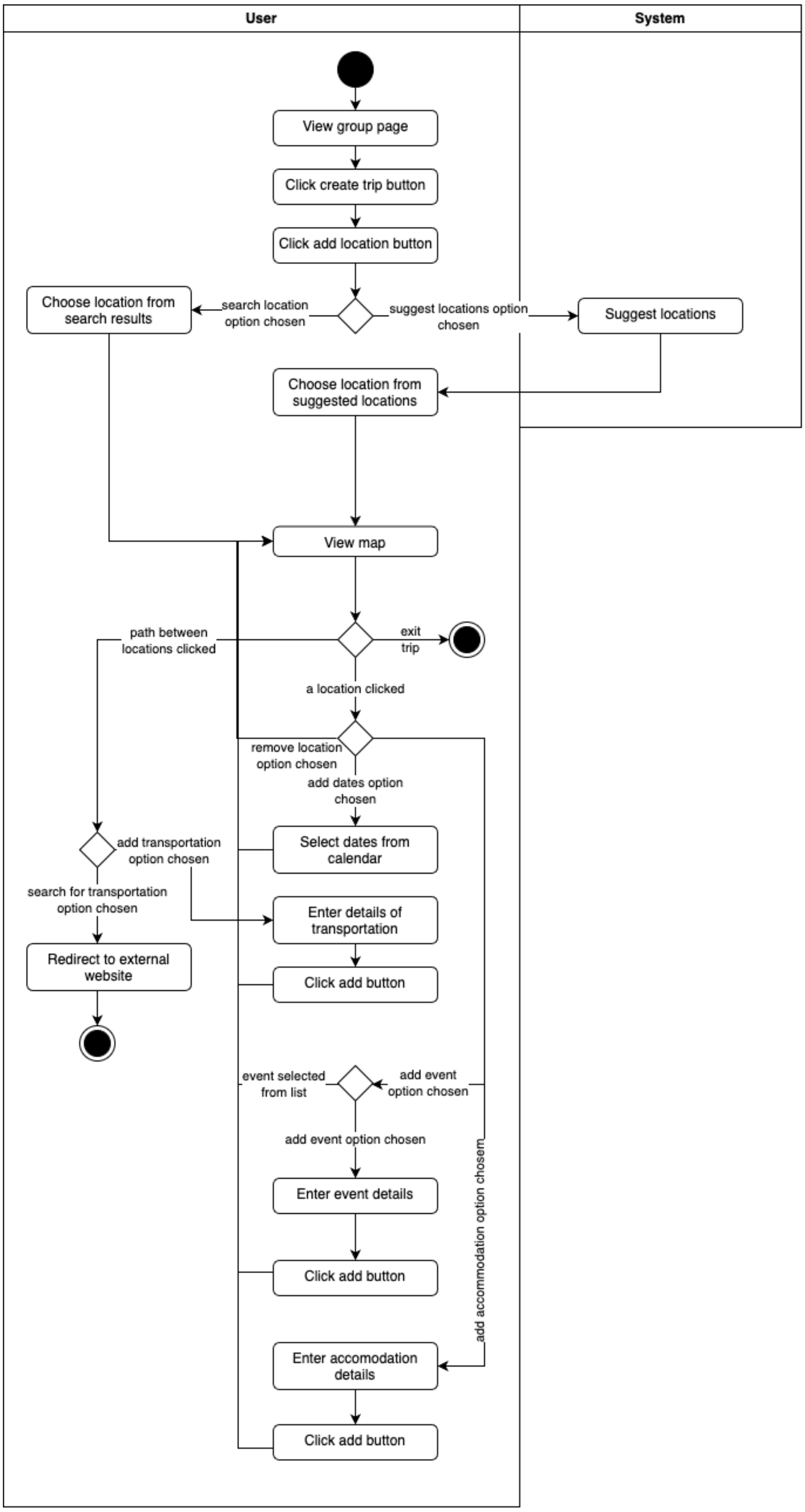
Add Available Dates



Create New Travel Group

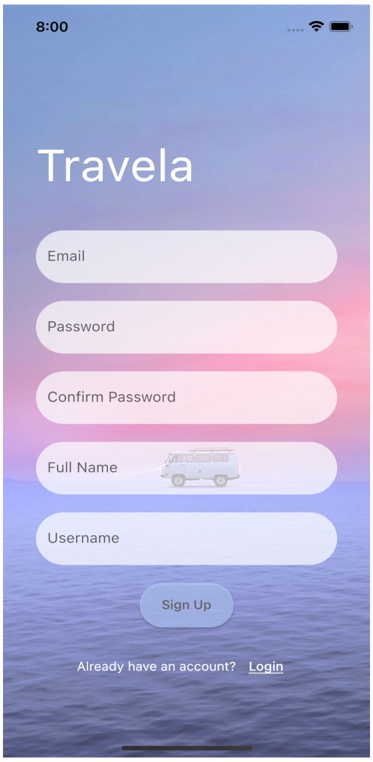


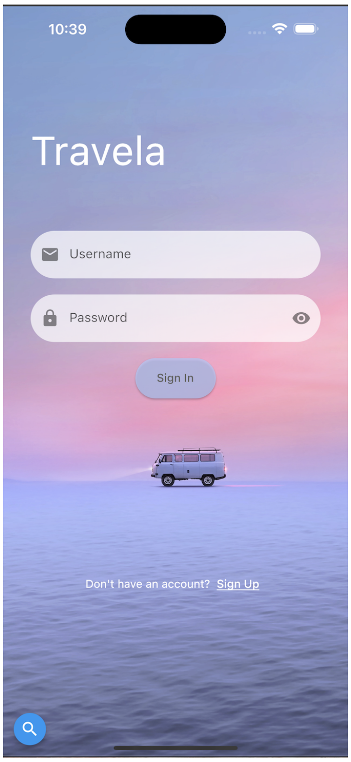
Create and Edit Trip


User
System
View group page
Click create trip button
Click add location button
Choose location from search results
search location option chosen
suggest locations option chosen
Suggest locations
Choose location from suggested locations
View map
path between locations clicked
exit trip
a location clicked
remove location option chosen
add dates option chosen
Select dates from calendar
add transportation option chosen
search for transportation option chosen
Enter details of transportation
Click add button
Redirect to external website
event selected from list
add event option chosen
add event option chosen
Enter event details
Click add button
add accommodation option chosem
Enter accomodation details
Click add button

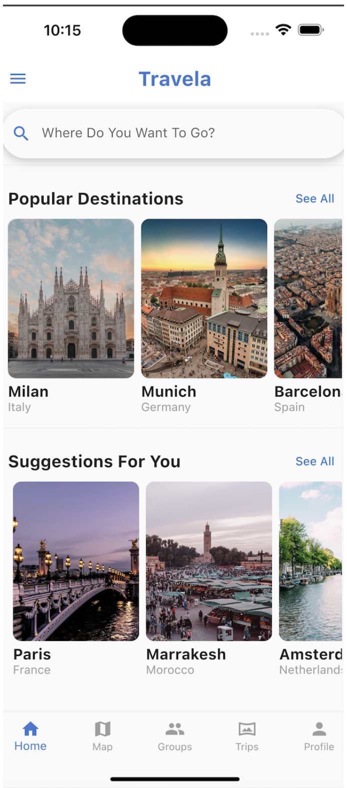
3.1.4 User Interface

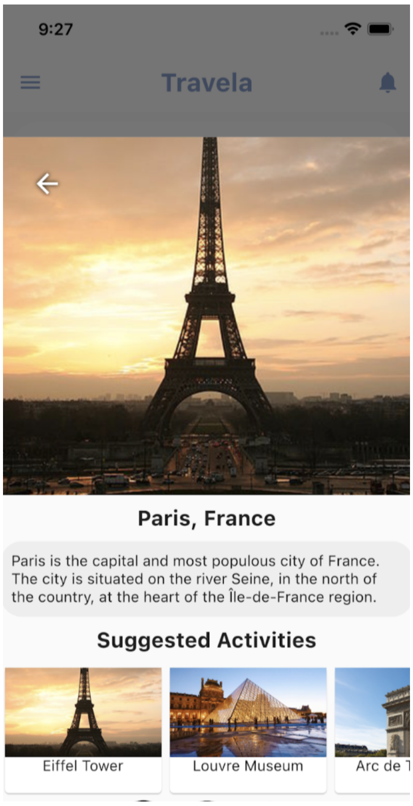
3.5.5.1 Sign Up - Login Page





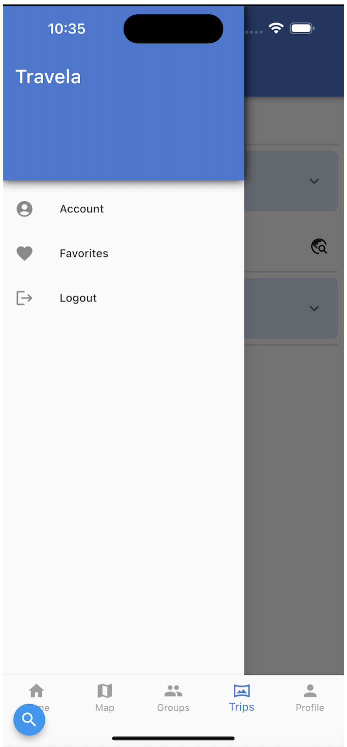
3.5.5.2 Home Page

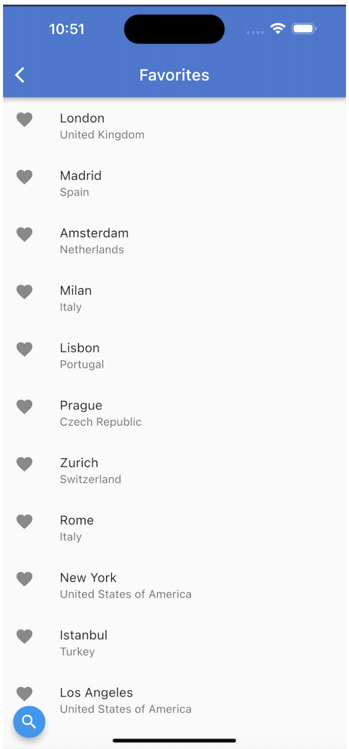




Paris is the capital and most populous city of France. The city is situated on the river Seine, in the north of the country, at the heart of the Île-de-France region.

3.5.5.3 Drawer and Favorites Page





London
United Kingdom

Madrid
Spain

Amsterdam
Netherlands

Milan
Italy

Lisbon
Portugal

Prague
Czech Republic

Zurich
Switzerland

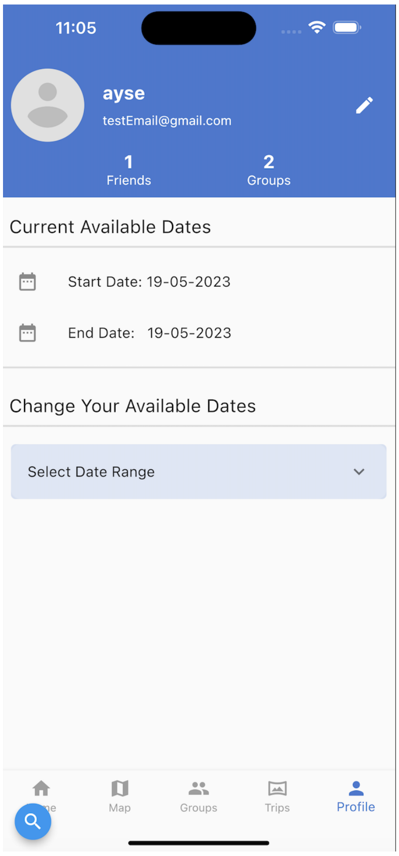
Rome
Italy

New York
United States of America

Istanbul
Turkey

Los Angeles
United States of America

3.5.5.4 Profile Page



ayse
testEmail@gmail.com

1 Friends
2 Groups

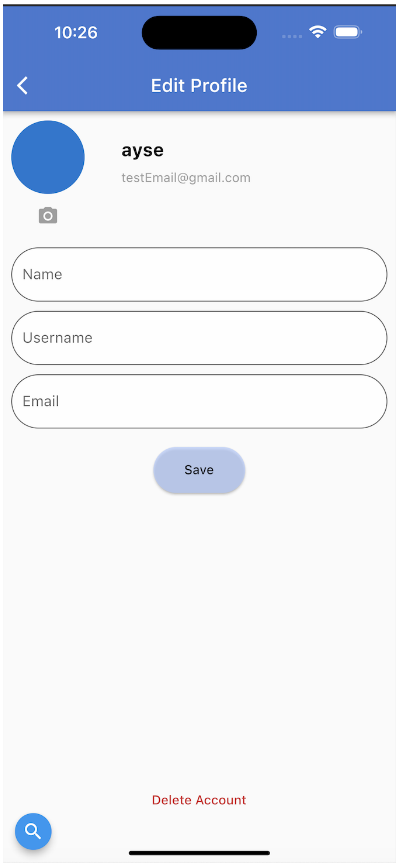
Current Available Dates

Start Date: 19-05-2023

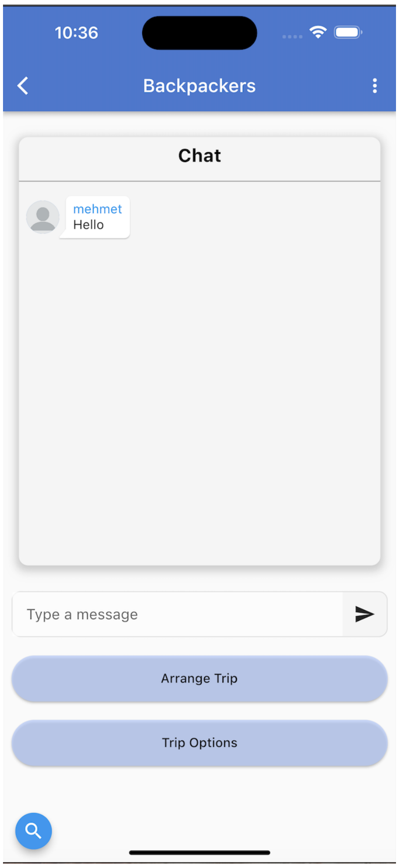
End Date: 19-05-2023

Change Your Available Dates

Select Date Range

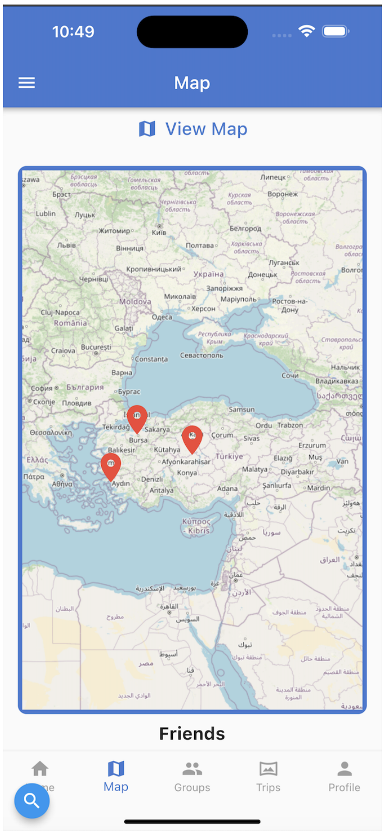


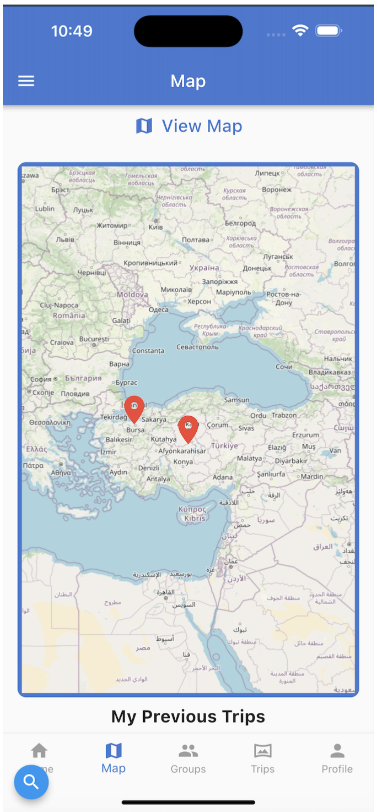
ayse
testEmail@gmail.com



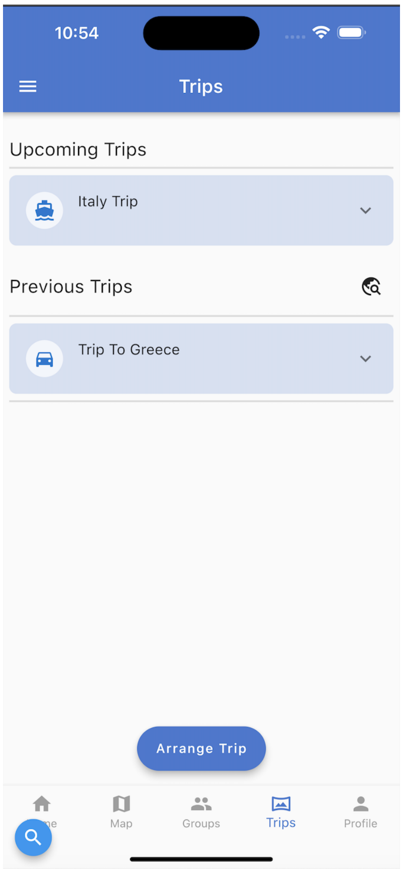
mehmet
Hello

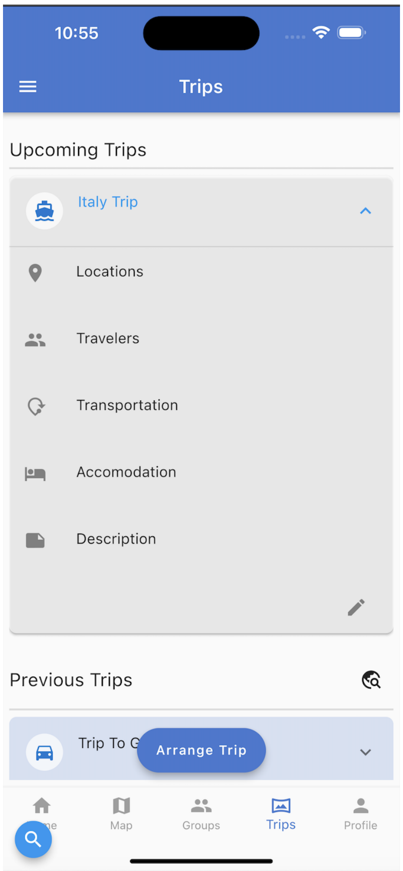
3.5.5.5 Maps Page

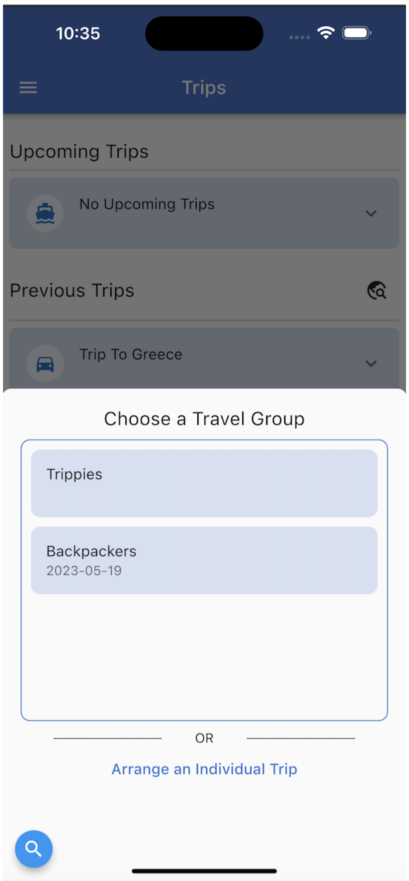




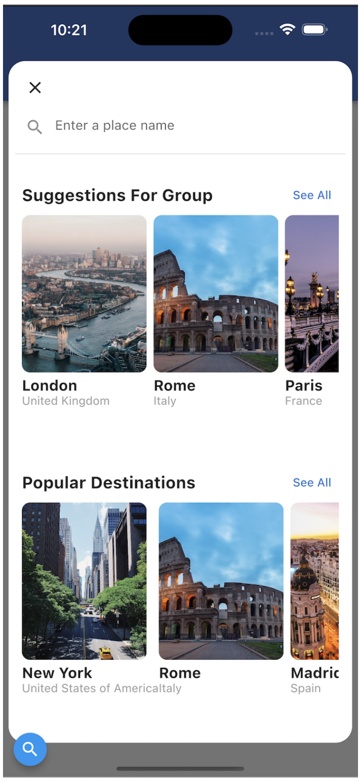
3.5.5.6 Trips Page

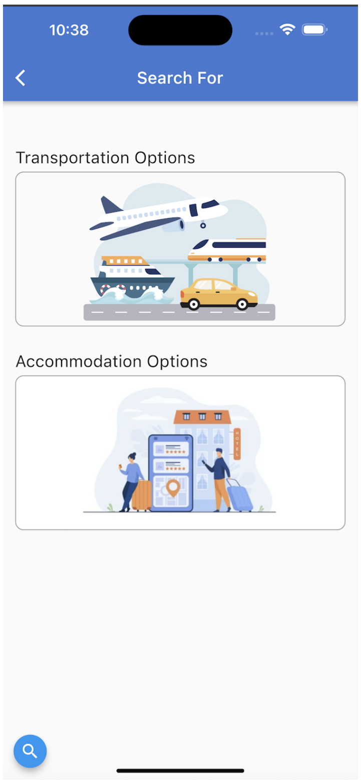




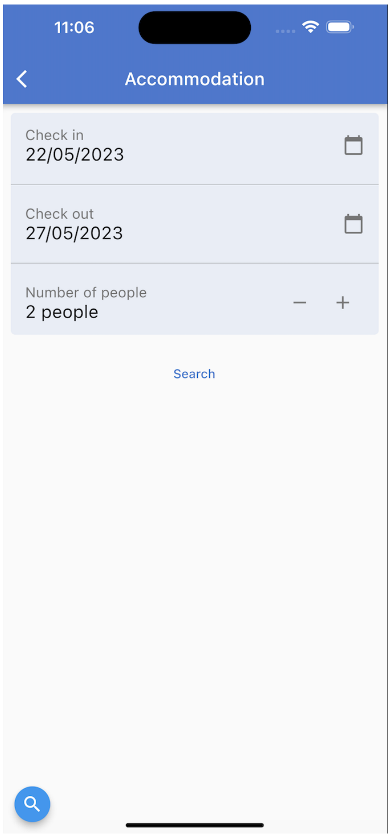


3.5.5.7 Adding City to trip and after adding city to trip





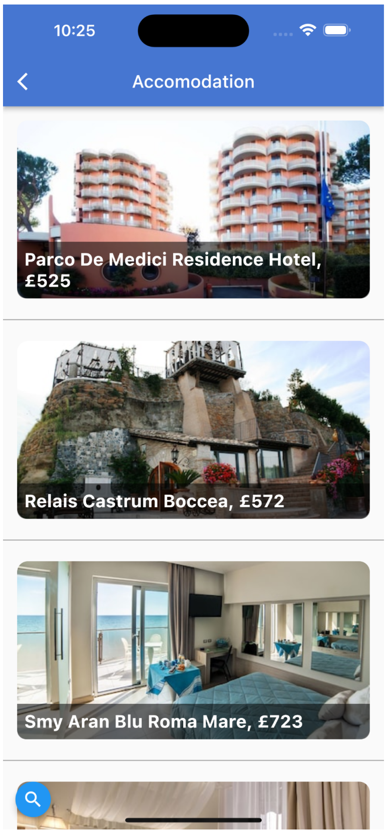
3.5.5.8 Accomodation Page



Check in
22/05/2023

Check out
27/05/2023

Number of people
2 people

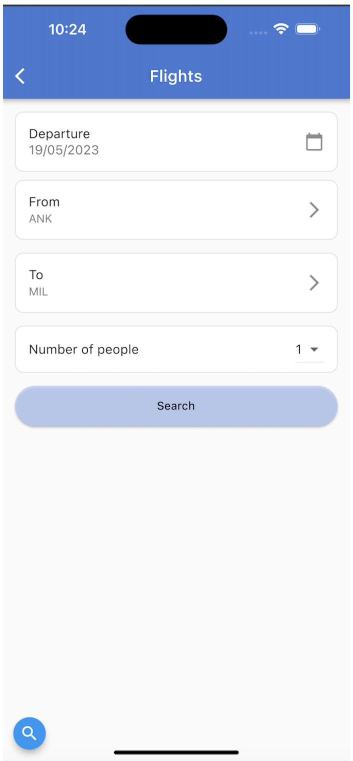


Parco De Medici Residence Hotel, £525

Relais Castrum Boccea, £572

Smy Aran Blu Roma Mare, £723

3.5.5.9 Transportation Options

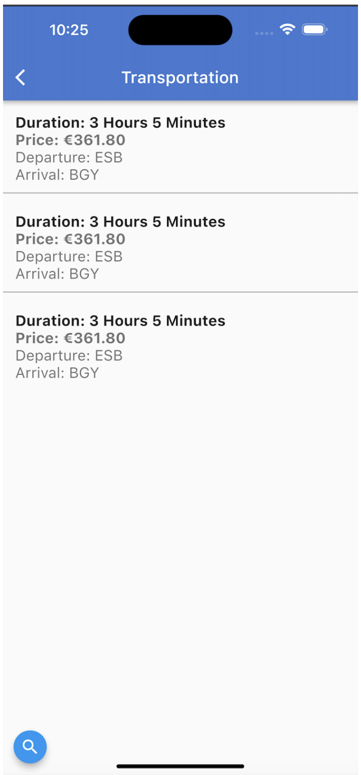


Departure
19/05/2023

From
ANK

To
MIL

Number of people 1

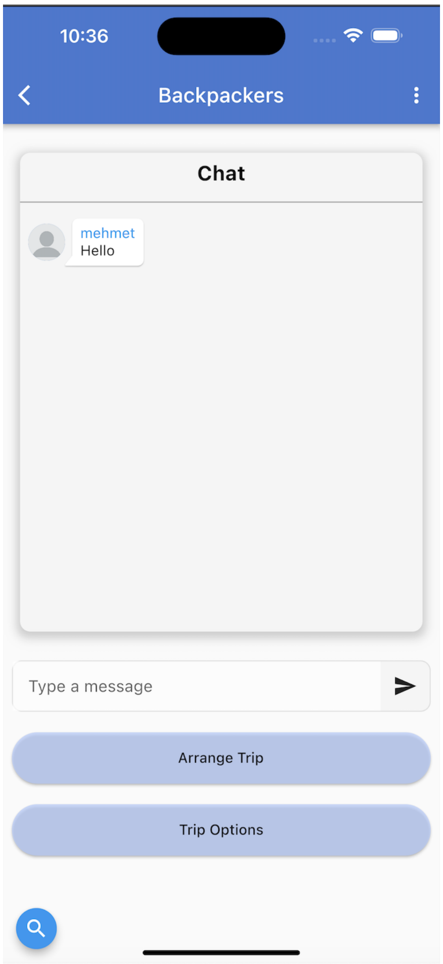


Duration: 3 Hours 5 Minutes
Price: €361.80
Departure: ESB
Arrival: BGY

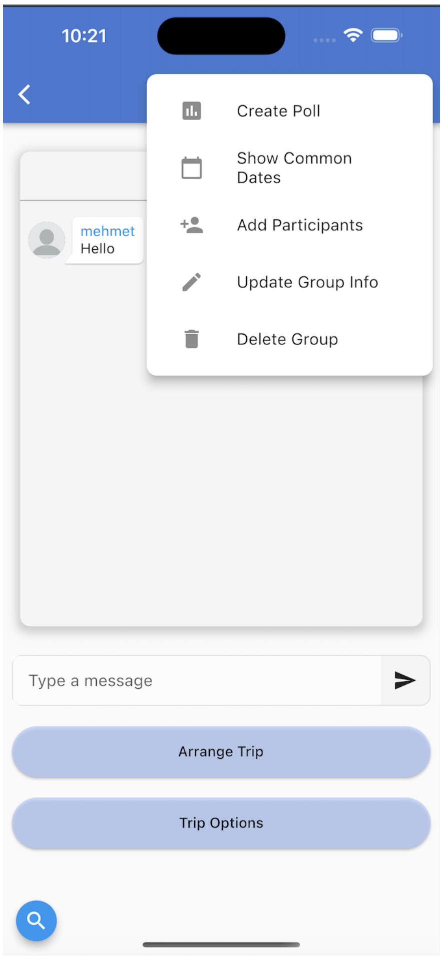
Duration: 3 Hours 5 Minutes
Price: €361.80
Departure: ESB
Arrival: BGY

Duration: 3 Hours 5 Minutes
Price: €361.80
Departure: ESB
Arrival: BGY

3.5.5.10 Groups Page



mehmet
Hello



mehmet
Hello

4 Development/Implementation Details

We have used Java with Spring Boot to build the backend of the application, which provided an efficient and secure structure. Also, the application uses a MySQL database on an RDS machine provided by AWS. Hibernate as ORM is used to reduce the number of commands we would need to pass to maintain the structure of the database. The application is deployed on an AWS EC2 instance, though it was initially planned to use Elasticbean as it provides more sources for backend deployments. However, we chose EC2 as it allowed ease of development and deployment. The application is deployed as a jar file built using Maven. As mentioned, hibernate and spring boot take care of database creations upon start, and we then write country and city information. MySQL Workbench and IntelliJ IDEA were used on local devices to develop the code and access the database.

5 Test Cases and Results

Test ID	TC#1
Test type/Category	Functional
Title	Check the response when invalid or missing information is given during sign-up
Procedure of testing steps	1. Fill the registration form with an already existing account information, invalid email, or do not fill in the necessary fields 2. Click the “Sign Up” button
Expected results	Sign Up/ Registration fails and an error message appears
Priority/Severity	Critical
Date tested and test result	Success - 19 May 2023

Test ID	TC#2
Test type/Category	Functional
Title	Check the database when the sign-up process succeeded
Procedure of testing steps	1. Fill the registration form with valid information and fill in all necessary fields. 2. Click the “Sign Up” button 3. Check the database to see whether the new user has been added with the given information.

Expected results	A new entry should be seen in the database with the given information
Priority/Severity	Critical
Date tested and test result	Success - 19 May 2023

Test ID	TC#3
Test type/Category	Functional
Title	Check the response when Google sign-up functionality is used
Procedure of testing steps	1. Click “Sign up with Google” button 2. Choose the Google account 3. Agree with the terms and conditions and authorize
Expected results	A new entry should be seen in the database with the given sign up information and Login succeeds
Priority/Severity	Low
Date tested and test result	Unsuccessful - 19 May 2023

Test ID	TC#4
Test type/Category	Functional
Title	Check the response when Google sign-in functionality is used
Procedure of testing steps	1. Click “Sign in with Google” button 2. Choose the Google account
Expected results	Login should succeed
Priority/Severity	Low
Date tested and test result	Unsuccessful - 19 May 2023

Test ID	TC#5
Test type/Category	Functional

Title	Check the response when an invalid password or email is given during login
Procedure of testing steps	1. Enter the email 2. Enter the password 3. Click the “Sign In” button
Expected results	Login fails and an error message appears.
Priority/Severity	Critical
Date tested and test result	Successful - 19 May 2023

Test ID	TC#6
Test type/Category	Functional
Title	Check the response when one of the necessary fields is empty during login.
Procedure of testing steps	1. Leave empty one of the necessary fields 2. Click the “Sign In” button
Expected results	Login fails and an error message appears.
Priority/Severity	Critical
Date tested and test result	Successful - 19 May 2023

Test ID	TC#7
Test type/Category	Functional
Title	Check the response during profile edit.
Procedure of testing steps	1. Go to the profile page. 2. Click “Edit ” icon in the top right corner. 3. Fill the fields that you want to change. 4. Click the “Save” button.
Expected results	Profile information should be updated and a success message appears
Priority/Severity	Major

Date tested and test result	Successful - 19 May 2023

Test ID	TC#8
Test type/Category	Functional
Title	Check the response during profile edit when all fields are empty
Procedure of testing steps	1. Go to the profile page. 2. Click “Edit ” icon in the top right corner. 3. Leave all the fields empty. 4. Click the “Save” button
Expected results	Profile information should not be updated and an error message appears
Priority/Severity	Major
Date tested and test result	Successful - 19 May 2023

Test ID	TC#9
Test type/Category	Functional
Title	Check the database when a travel group is formed
Procedure of testing steps	1. Go to the groups page 2. Click the “Form Group” button 3. Select friends to send an invitation and press the “Send Invitation” button 4. Click “Close” button when he/she is done sending invitations
Expected results	A new row should be seen in the database with the given information
Priority/Severity	Critical
Date tested and test result	Successful - 19 May 2023

Test ID	TC#10

Test type/Category	Functional
Title	Check the response after adding a friend to a travel group
Procedure of testing steps	1. Go to the groups page 2. Select the travel group that you want to add a friend 3. Click “+” button and choose the friend to be added and send invitation or send a friend request.
Expected results	The chosen friend should be added to the travel group, database should be updated and a travel group notification should be sent to the chosen friend
Priority/Severity	Critical
Date tested and test result	Successful - 19 May 2023

Test ID	TC#11
Test type/Category	Functional
Title	Check the response when a friend request is sent
Procedure of testing steps	1. Go to the groups page 2. Click the “Friends” icon at the upper right corner 3. Click the “Plus” button at the bottom of the page 4. Search for a user name to which the request will be sent. 5. Click the “Send” button
Expected results	The request should be seen in the same section (mentioned above) of the user so that the request is sent.
Priority/Severity	Critical
Date tested and test result	Successful - 19 May 2023

Test ID	TC#12
Test type/Category	Functional
Title	Check the response when a friend request is sent to the user itself or to an already friend.
Procedure of testing steps	1. Go to the map page 2. Click the “Friends” icon in the upper right corner

	3. Click the “Plus” button at the bottom of the page 4. Search for the username of a friend or yourself 5. Click the “Send” button
Expected results	Sending friend request fails and an error message appears.
Priority/Severity	Major
Date tested and test result	Successful - 19 May 2023

Test ID	TC#13
Test type/Category	Functional
Title	Check the response after arranging a trip.
Procedure of testing steps	1. Go to the groups page. 2. Select the travel group that you want to have a trip with. 3. Select among available dates of group members. 4. Select a place among ML suggestions that are given by the system or according to the poll results that are created by the other users.
Expected results	The user can view the drafted trip.
Priority/Severity	Major
Date tested and test result	Successful - 19 May 2023

Test ID	TC#14
Test type/Category	Functional
Title	Check the order of trip suggestions, suggested by the recommendation system, and check the response on Amadeus API calls
Procedure of testing steps	1. Go to the groups page. 2. Select the travel group that you want to have a trip with. 3. Select among available dates of group members. 4. ML suggestions should be listed
Expected results	The ML suggestions should be formed by requesting the Amadeus API. First, common locations that are visited by the

	travel group should be used as a parameter to the recommendation algorithm, then the rest of the cities. The API should return suggestions, and it should be displayed to the user by their order of relevance.
Priority/Severity	Major
Date tested and test result	Successful - 19 May 2023

Test ID	TC#15
Test type/Category	Functional
Title	Check the database after arranging a trip.
Procedure of testing steps	1. Go to the groups page 2. Select the travel group that you want to have a trip with. 3. Select friends to send an invitation and press the plus icon button. 4. Select a place among ML suggestions that are given by the system or according to the poll results that are created by the other users. 5. Vote for suitable hotels and transportation opportunities for the given available dates and places.
Expected results	A new trip should be seen in the database with the given information.
Priority/Severity	Critical
Date tested and test result	Successful - 19 May 2023

Test ID	TC#16
Test type/Category	Functional
Title	Check the response when a photo is posted for a previous trip.
Procedure of testing steps	1. Go to the maps page 2. Go to the “My Previous Trips” page by sliding the pages to the right. 3. Click the “View Map” button in the top of the page. 4. Click the trip location on the map that you want to add a photo.

	5. Click the “Add Photo” button in the top of the page . 6. Click the “Allow” button for access permission to your gallery. 7. Select the photos that you want to post from your gallery. 8. Click the “Upload” button.
Expected results	The photo should appear on the “My Previous Trips” page on the specific trip.
Priority/Severity	Major
Date tested and test result	Successful - 19 May 2023

Test ID	TC#17
Test type/Category	Functional
Title	Check the response when a previous trip is rated.
Procedure of testing steps	1. Go to the trips page 2. Go to the “Previous Trips” part . 3. Click the “Rate” dropdown button and select from 1 to 5.
Expected results	The overall rate of that place should be recalculated according to the given rate and displayed on the location itself.
Priority/Severity	Critical
Date tested and test result	Successful - 19 May 2023

Test ID	TC#18
Test type/Category	Functional
Title	Check the database when the account is deleted.
Procedure of testing steps	1. Go to the profile page. 2. Click the “Edit ” icon in the top right corner. 3. Click the “Delete Account ” button on the bottom of the page.
Expected results	Account should be deleted from database, along with all the personal data of the user (previous trips etc.)
Priority/Severity	Critical

Date tested and test result	Successful - 19 May 2023

Test ID	TC#19
Test type/Category	Functional
Title	Check the response when a city is added to favorites.
Procedure of testing steps	1. Go to the home page or search the wanted city from the home page. 2. Click one of the locations. 3. Click the "Heart" icon on the top right corner of the page.
Expected results	The city should be seen in the favorites page that can be reached from the drawer.
Priority/Severity	Minor
Date tested and test result	Successful - 19 May 2023

Test ID	TC#20
Test type/Category	Functional
Title	Check the response when a city is selected.
Procedure of testing steps	1. Go to the home page or search the wanted city from the Home Page. 2. Click one of the locations.
Expected results	The suggested activities should be seen on the bottom sheet.
Priority/Severity	Major
Date tested and test result	Successful - 19 May 2023

Test ID	TC#21
Test type/Category	Functional
Title	Check the response when available dates are entered.
Procedure of testing	1. Go to profile page.

steps	2. Select the available dates of travel from the calendar.
Expected results	Calendar is updated according to the given dates.
Priority/Severity	Critical
Date tested and test result	Successful - 19 May 2023

Test ID	TC#22
Test type/Category	Functional
Title	Check the response when the “Show Common Dates” button is clicked.
Procedure of testing steps	1. Go to groups page. 2. Select the travel group that you want to have a trip with. 3. Click the “Show Common Dates” button on the upper right corner menu.
Expected results	User should only see available common dates of all group members in the calendar.
Priority/Severity	Critical
Date tested and test result	Successful - 19 May 2023

Test ID	TC#23
Test type/Category	Functional
Title	Check the response when a message is sent in group chat.
Procedure of testing steps	1. Go to groups page. 2. Select the travel group that you want to have a trip with. 3. Click to edit icon. 4. Write the message you want to send.
Expected results	The message should be seen in the chat box of the group.
Priority/Severity	Critical
Date tested and test result	Successful - 19 May 2023

Test ID	TC#24
Test type/Category	Functional
Title	Check the response when a poll is created.
Procedure of testing steps	1. Go to groups page. 2. Select the travel group that you want to have a trip with. 3. Click the “Create Poll” button. 4. Add the questions and the choices for forming the poll. 5. Click the “Create” button.
Expected results	The poll should be seen in the chat box and can be reached by all members.
Priority/Severity	Critical
Date tested and test result	Successful - 19 May 2023

Test ID	TC#25
Test type/Category	Functional
Title	Check the response when a location is voted more than once while arranging a trip.
Procedure of testing steps	1. Go to the groups page. 2. Select the travel group that you want to have a trip with. 3. Select among available dates of group members. 4. Select a place among ML suggestions that are given by the system or according to the poll results that are created by the other users. 5. Vote for suitable hotels and transportation opportunities for the given available dates and places by selecting more than one opportunity.
Expected results	Voting fails and an error message appears.
Priority/Severity	Critical
Date tested and test result	Successful - 19 May 2023

Test ID	TC#26

Test type/Category	Functional
Title	Check whether search bar for places to visit returns relevant results
Procedure of testing steps	1. Enters a place name he/she wishes to search in the TextField at the top of the Home page. 2. Press the “Search” button.
Expected results	If the input is valid, it should show a number of places that are relevant to the input. If not, it should display an error message.
Priority/Severity	Critical
Date tested and test result	Successful - 19 May 2023

Test ID	TC#27
Test type/Category	Functional
Title	Check whether the info (including review and rating) of selected destination can be viewed
Procedure of testing steps	1. Searches and selects a place he/wishes to view in the main menu. 2. View Preferred season information of a destination. 3. Scrolls down and views the review and rating section.
Expected results	For every destination, reviews and its rating should be up to date and synchronously retrieved from the database.
Priority/Severity	Critical
Date tested and test result	Unsuccessful - 19 May 2023

Test ID	TC#28
Test type/Category	Functional
Title	Check the home page for user destination suggestions
Procedure of testing steps	1. Switches to home page.

Expected results	The most suitable destination options specifically recommended to the user should be seen in the Home page.
Priority/Severity	Major
Date tested and test result	Successful - 19 May 2023

Test ID	TC#29
Test type/Category	Functional
Title	Check whether there are activity suggestions for a destination
Procedure of testing steps	1. Switch to home page. 2. Select a destination. 3. Views the destination page.
Expected results	The popular activities that can be done in that destination should be seen.
Priority/Severity	Minor
Date tested and test result	Successful - 19 May 2023

Test ID	TC#30
Test type/Category	Functional
Title	Check if upcoming trips is visible
Procedure of testing steps	1. Switch to the trips page, after creating a trip on the app.
Expected results	Upcoming trips that are planned by their travel groups should be visible on the screen.
Priority/Severity	Critical
Date tested and test result	Successful - 19 May 2023

Test ID	TC#31
Test type/Category	Functional

Title	Check whether the previous trips of yourself is visible or not
Procedure of testing steps	1. Switch to the map page.
Expected results	A world map with the previous trips of the user should be seen directly.
Priority/Severity	Major
Date tested and test result	Successful - 19 May 2023

Test ID	TC#32
Test type/Category	Functional
Title	Check whether the previous trips of your friends is visible or not in the World map
Procedure of testing steps	1. Switch to the map page. 2. View the friend list that is found in the Maps page. 3. Click to the friend's name that has their own map. 4. Sees the related info (photo,review) of their friends.
Expected results	A world map with the previous trips of the user's friends should be seen.
Priority/Severity	Major
Date tested and test result	Successful - 19 May 2023

Test ID	TC#33
Test type/Category	Functional
Title	Get a bearer token for authorization
Procedure of testing steps	1. Make a post request to "/authentication/token" endpoint with the body {username, password} 2. Try accessing the list of trips of your own. 3. Try accessing the list of trips of another user.
Expected results	Get a valid bearer token that allows the user to access user's own data, but not others.

Priority/Severity	Critical
Date tested and test result	Successful - 19 May 2023

Test ID	TC#34
Test type/Category	Functional
Title	Check response to wrong authentication requests
Procedure of testing steps	1. Make a post request to “/authentication/token” endpoint with wrong information the body {username, password}
Expected results	The endpoint should not give a bearer token for wrong credentials. There should be an error.
Priority/Severity	Critical
Date tested and test result	Successful - 19 May 2023

Test ID	TC#35
Test type/Category	Functional
Title	Check response when a poll receives a vote
Procedure of testing steps	1. Navigate to a group chat. 2. Click on a poll. 3. Vote one of the options in the poll by clicking on it. 4. View the results of the poll.
Expected results	The poll results should update according to the new vote received.
Priority/Severity	Critical
Date tested and test result	Successful - 19 May 2023

Test ID	TC#36
Test type/Category	Functional
Title	Check response when a location is added to a trip

Procedure of testing steps	1. Navigate to a trip. 2. Click on edit button. 3. Search for a new location by name. 4. Choose the location to add to the trip.
Expected results	The location can be seen among the list of locations in the trip and is marked on the map.
Priority/Severity	Critical
Date tested and test result	Successful - 19 May 2023

Test ID	TC#37
Test type/Category	Functional
Title	Check whether password is encrypted when a user signs up.
Procedure of testing steps	1. Navigate to sign up page. 2. Choose the sign up with email option. 3. Fill in the account information. 4. Click sign up button
Expected results	The newly registered user's password should be encrypted in the database.
Priority/Severity	Critical
Date tested and test result	Successful - 19 May 2023

Test ID	TC#38
Test type/Category	Functional
Title	Check if the user gets a notification when a new message is sent to a group chat.
Procedure of testing steps	1. The user has the Travela app installed and is logged in. 2. A new message is sent to a group chat that the user is a member of.
Expected results	A push notification is sent to the user's device.
Priority/Severity	Critical

Date tested and test result	Unsuccessful - 19 May 2023

Test ID	TC#39
Test type/Category	Functional
Title	Check the response when a new transportation method is added to a trip.
Procedure of testing steps	1. Go to the groups page. 2. Select group. 3. Arrange a trip by clicking the button at the bottom of the page. 4. Select the origin location. 5. Select the destination location. 6. Select number of passengers. 7. Select dates. 8. Click on the add transportation button. 9. Select type of transport. 10. Select the transportation method from available options.
Expected results	The transportation method between selected locations should be visible among trip information and should be added to the map between the two locations.
Priority/Severity	Major
Date tested and test result	Successful - 19 May 2023

Test ID	TC#40
Test type/Category	Functional
Title	Check the response when a new accommodation method is added to a trip location.
Procedure of testing steps	1. Go to the groups page. 2. Select group. 3. Arrange a trip by clicking the button at the bottom of the page. 4. Select the location. 5. Select number of adults. 6. Select dates.

	7. Click on the add accommodation button. 8. Select the accommodation option from available options.
Expected results	The accommodation method in a location should be visible among trip information.
Priority/Severity	Major
Date tested and test result	Successful - 19 May 2023

Test ID	TC#41
Test type/Category	Functional
Title	Check whether old messages are still available after a user leaves a group.
Procedure of testing steps	1. Navigate to a group. 2. Click on the leave group button.
Expected results	The past messages from the group should still be available, but the messages sent to the group after they leave should not be sent to the user.
Priority/Severity	Critical
Date tested and test result	Successful - 19 May 2023

Test ID	TC#42
Test type/Category	Functional
Title	Check the total cost of the trip.
Procedure of testing steps	1. Navigate to a trip. 2. Click on the trip to view more information. 3. View the total cost of the trip.
Expected results	The cost of the trip should be equal to the sum of all the costs of accommodation and transportation.
Priority/Severity	Minor
Date tested and test result	Successful - 19 May 2023

Test ID	TC#43
Test type/Category	Functional
Title	Check whether previous trips are updated correctly.
Procedure of testing steps	1. Navigate to the trips page.
Expected results	If the trip's date has passed, the completed trip should be removed from the current trips page and be added to the previous trips page.
Priority/Severity	Critical
Date tested and test result	Successful - 19 May 2023

Test ID	TC#44
Test type/Category	Functional
Title	Check response to password change
Procedure of testing steps	1. Go to the Profile Page. 2. Click change password. 3. Enter current and new password. 4. Confirm password change. 5. Log out. 6. Try to login with the old password. 7. Try to login with the new password.
Expected results	New password should be encrypted in the database and replace the old password. When the old password is used, login should be unsuccessful. When the new password is used, the user should be able to login successfully.
Priority/Severity	Critical
Date tested and test result	Successful - 19 May 2023

Test ID	TC#45
Test type/Category	Functional

Title	Check response to trip deletion
Procedure of testing steps	1. Navigate to a trip. 2. Click delete trip button. 3. Refresh trips page.
Expected results	Trip should be removed from the group and/or the user's profile, along with all the locations, transportation methods and accommodation options. Finally, the trip itself should be removed from the database.
Priority/Severity	Critical
Date tested and test result	Successful - 19 May 2023

Test ID	TC#46
Test type/Category	Functional
Title	Check response to group deletion
Procedure of testing steps	1. Navigate to the groups page. 2. Click the delete group button.
Expected results	All members of the group should be removed from the group. All trips associated with that group should be deleted. All messages in the group chat should be deleted. Finally, the group itself should be removed from the database.
Priority/Severity	Critical
Date tested and test result	Successful - 19 May 2023

Test ID	TC#47
Test type/Category	Functional
Title	Check the response of correct login credentials
Procedure of testing steps	1. Enter the email 2. Enter the password 3. Click the "Sign In" button
Expected results	If the login credentials are correct, the user should be navigated to the home page.

Priority/Severity	Critical
Date tested and test result	Successful - 19 May 2023

Test ID	TC#48
Test type/Category	Non-functional
Title	Check the response of the navigation bar.
Procedure of testing steps	1. Go to the home page. 2. Click one of the icons on the navigation bar.
Expected results	User should be navigated to the related page.
Priority/Severity	Critical
Date tested and test result	Successful - 19 May 2023

Test ID	TC#49
Test type/Category	Non-functional
Title	Check the positioning of the drawer widget.
Procedure of testing steps	1. Go to the home page or map page. 2. Click the “Drawer” icon in the upper left corner.
Expected results	The navigation bar should not be seen with the drawer.
Priority/Severity	Major
Date tested and test result	Successful - 19 May 2023

Test ID	TC#50
Test type/Category	Non-functional
Title	Check the map in trip view
Procedure of testing steps	1. Navigate to a trip. 2. View the trip in map format.
Expected results	The trip should appear as a map, where the locations and

	transportations between them are shown and further information about a location can be viewed by clicking on it.
Priority/Severity	Major
Date tested and test result	Unsuccessful - 19 May 2023

Test ID	TC#51
Test type/Category	Non-functional
Title	Check that the messages in group chats are in correct order.
Procedure of testing steps	1. Navigate to a group chat. 2. Send several messages to the group chat. 3. Receive several messages from other group members.
Expected results	The messages should appear sorted based on time, with the most recent message appearing at the bottom.
Priority/Severity	Critical
Date tested and test result	Successful - 19 May 2023

Test ID	TC#52
Test type/Category	Performance
Title	Check whether the login operation takes less than 2 seconds
Procedure of testing steps	1. Enter the email 2. Enter the password 3. Click the “Sign In” button
Expected results	The login operation which involves backend and database communication should take less than 2 seconds, and output the correct response.
Priority/Severity	Minor
Date tested and test result	Successful - 19 May 2023

Test ID	TC#53

Test type/Category	Performance
Title	Check whether the suggest trip functionality takes less than 8 seconds
Procedure of testing steps	1. Go to the groups page. 2. Select the travel group that you want to have a trip with. 3. Select among available dates of group members. 4. ML suggestions should be listed
Expected results	Recommending trip locations which involve backend and database communication as well as external API calls should take less than 8 seconds, and output the correct response.
Priority/Severity	Minor
Date tested and test result	Successful - 19 May 2023

Test ID	TC#54
Test type/Category	Performance
Title	Check the response during fetching data
Procedure of testing steps	1. Go to the Maps page.
Expected results	See the loading icon on the page until the data is fetched from the server, which should not take more than 5 seconds.
Priority/Severity	Minor
Date tested and test result	Successful - 19 May 2023

6 Maintenance Plan and Results

Currently, we are hosting the application backend on Amazon Web Services' EC2 cloud service and we plan to continue using AWS's resources to host our project. By leveraging AWS, we can benefit from a range of tools specifically designed for logging and monitoring our application. This way, we will have valuable insight into how our app is performing and being used.

We also have a lot of features on mind that we want to implement in the future. However, we are aware that it takes a lot of responsibility to add new features as there's a risk that existing features are broken. We need to be careful not to disrupt the production version of our app and release new features with great attention.

Additionally, we will make use of Amadeus API's App Analytics service. This service provides us with comprehensive information about the number and frequency of calls made to the Amadeus API through our application. This approach allows us to assess the effectiveness of Amadeus API integration and optimize the usage of the API.

By combining the robust logging and monitoring capabilities offered by AWS, the careful implementation of new features, and the usage of Amadeus API's App Analytics service, we believe that our maintenance plan will be effective and we will be able to minimize potential issues that may arise in the future.

7 Other Project Elements

7.1 Consideration of Various Factors in Engineering Design

7.1.1 Aesthetics

Aesthetics of a product is one of the most important things as before looking at the specifications of a product, the first thing people notice is the way the app looks. To ensure a good aesthetic feel in the first look, we will try to follow the design trends in the industry and create a modern interface.

7.1.2 Maintenance

To develop a sustainable and continuous product, responsive design is a crucial part as Travela will be available for many devices and platforms with different sizes. Maintaining different designs for all different platforms is not a good idea for effectiveness so our goal will be to create limited different designs when needed and responsive for all devices otherwise.

7.1.3 Social Factors

Any person from any gender, age, race, etc can use Travela both individually or in groups. Thus, we can say that there are no social factors directly affecting our app.

7.2 Ethics and Professional Responsibilities

The first and foremost responsibility we had was the responsibility we had for each other. From the moment we decided to form a project group, we have the responsibilities of completing our share of the work and contributing to the reports. We also have the responsibility of cultivating an environment where everyone's ideas are respected and we can communicate in healthy ways. We always made sure to report to the rest of the group whenever we worked on something new or experienced any issues. We split tasks based on everyone's technical skills and experience and helped each other frequently.

We have responsibilities not just towards each other but also towards the users of our application. Whenever an user decides to sign up for Travela, we are responsible for keeping their information safe and secure. For this purpose, we used security tools such as Spring security because existing frameworks would be much easier to implement than trying to do it by ourselves, not to mention significantly more secure. We use bearer tokens to confirm that an user is properly authenticated and cannot access other users' data. We also encrypt users' passwords when storing them in our database so that the passwords cannot be viewed by anyone.

7.3 Teamwork Details

7.3.1 Ensuring Proper Teamwork

Teamwork is one of the most important factors that lead to success in such projects. Thus, it is important to utilize as much teamwork as possible, for the best outcomes. There are several ways of ensuring this, some examples can be seen below.

- Dividing work equally that matches the interests and specialities of the team members, in face to face or online meetings
- Scheduling and having meetings weekly
- Setting weekly/monthly goals about the project, foreseeing how much work has to be done by each individual and planning forward accordingly
- Using Google docs for enabling simultaneous work on reports
- Using Git and GitHub for the version control and enabling simultaneous work on the code
- Using a kanban board (maybe Jira) to keep track of the weekly goals and sprints
- Using communication channels such as Whatsapp efficiently, to daily update the work that has been done, or the progresses that have been made, or which tasks need more manpower to finish
- Communicating regularly

7.3.2 Contributing and functioning effectively on the team

Efe Ertürk contributed to the backend part of the project, as well as documentation. He also worked on ML solutions for the recommendation problem.

Efe Şaman worked on the backend of the project, mainly focusing on the authentication and authorization features. Took part in designing software architecture as well.

Çağla Ataoğlu is on the backend team as well. She mostly focused on part of the controller, service and entity code. She contributed to the UML diagrams during the design process.

Yağmur Eryılmaz is on the frontend team. She contributed to the UI design and implementation. She worked on the connection between backend and frontend. Also contributed to the documentation and UML.

Cenk Duran contributed to the frontend part of the project. He mainly focused on the implementation of the UI and also the API integrations of the project.

7.3.3 Helping create a collaborative and inclusive environment

To help create a collaborative and inclusive environment, Efe Ertürk used communication channels like whatsapp regularly and efficiently to keep up with the overall condition of other teammates and also to be involved in the project decisions. He also helps schedule weekly meetings.

Efe Şaman played an active role on Whatsapp and other communication channels to create an inclusive and friendly environment. Worked on building a casual, collaborative relation with his peers. Focused on the relation with his peers as much as the code.

To help create an environment that values everyone’s opinions and preferences, Çağla Ataoğlu took part in communicating with the group through their Whatsapp group, online meetings and in-person meetings. She listened and valued others’ opinions and helped distribute workload and tasks.

Yağmur Eryılmaz had an active role on Whatsapp and other communication channels and also in person meetings to create an inclusive environment. Also she expressed her opinions clearly and contributed to discussions and helped creating a collaborative environment via focusing on the relation with her teammates.

Cenk Duran played an active role on our communication channels, in the group meetings both online and face to face. He described how he wanted to implement directly with the team and worked according to the group’s decision all the time.

7.3.4 Taking the lead role and sharing leadership on the team

As a team, every individual had different roles for different tasks, and the lead role was shared amongst the team members and switched frequently.

Efe Ertürk took the leading role at backend artifact of the project, as well as the ML part of the project, due to his prior experiences in these domains.

Efe Şaman is responsible for the authorization and authentication part of the project, and took a lead role in building it on the backend. Moreover, at times he took a leading role in designing the backend.

Çağla Ataoğlu assumed a leadership role in the backend team in some instances. On some occasions, she led meetings and other collaborative work, such as writing documentation.

Yağmur Eryılmaz took a leading role in UI design and determination of features. She also communicated with the supervisor and informed him about the advancements.

Cenk Duran took the leading role at implementing the frontend artifact because of his previous experiences with flutter. He used his experience in both mobile and web platforms.

7.3.5 Meeting objectives

We set out to achieve a primary objective: providing personalized trip suggestions to a diverse group of users with varying interests, budgets, and time constraints. We are proud to state that we have successfully met this core objective, delivering a valuable service to our travel enthusiasts. By implementing a robust algorithm and leveraging user preferences, we have been able to offer tailored recommendations that cater to individual needs, ensuring an enhanced travel experience for all.

Moreover, we have effectively addressed several sub-objectives within our software. Users can now explore popular destinations, allowing them to stay informed about trending travel hotspots and make informed decisions. The feature to favorite cities enables users to create personalized travel wish lists, saving their preferred locations for future reference. Additionally, we have integrated social networking functionalities, empowering users to connect with friends, form travel groups, and engage in seamless communication through chat features. To streamline decision-making processes within a group, we have incorporated poll functionalities, enabling users to collectively decide on various aspects of their trip.

In our efforts to enhance user experience, we have integrated a map view that displays trips and visited locations, providing a visual representation of travel history and facilitating a sense of accomplishment. Furthermore, our software encompasses numerous other functionalities, each designed to enrich the travel planning and exploration journey. These include features such as itinerary creation, hotel and restaurant recommendations, transportation options, and even local event suggestions.

By successfully meeting these sub-objectives, we have created a comprehensive and versatile software solution that not only caters to the core objective of personalized trip suggestions but also offers a wide array of supplementary features to elevate the overall user experience. Our commitment to meeting these objectives has resulted in a software product that is intuitive, user-friendly, and capable of meeting the diverse needs and preferences of our user base.

7.4 New Knowledge Acquired and Applied

For the frontend artifact, we learned new technologies such as Dart and Flutter. With that, we learned mobile and also web development. We learned to connect a flutter mobile application and connect it to a backend server. Also, we learned to deploy a website using Amazon AWS and deploy the mobile app to Google Play Store. Hope the app is accepted by Google Play Store as soon as possible, so that our mobile users can enjoy the app as well.

For the backend artifact, we learned new technologies such as Java, SpringBoot and Maven to generate a REST-CRUD API. We learned to connect the Java backend to a MySQL server. Moreover, we learned to work with other API's. Also, we learned to deploy the whole backend server along with the MySQL database to the cloud, Amazon AWS. For testing purposes, we learned to use Postman and MySQL workbench.

Last, but not least, we learned to work as a team. Without teamwork, this project would have been much harder. Thanks to the teamwork, the whole process that consists of requirements elicitation, design, implementation, quality assurance and testing stages was a lot of fun.

8 Conclusion and Future Work

In the end, we learned a lot during the entire process of building Travela. We are proud of our work and we believe that it's an important step in solving the problem of arranging trips. As people who have dealt with the experience of trying to plan a trip with groups of people, we as developers have awareness of the scope of the problem we tackled. During the development of Travela, we didn't have difficulty putting ourselves in our users' shoes and consider which features we would want and why.

While we're content with what we have built, there are still goals we haven't met and features we would like to implement in the future. During our meetings, we discussed way more features than we could implement, which means that even though the period for working on this project has ended, we are still filled with ideas and potential improvements that we could work on in the future. For example, we want to go further with the social media aspect of the application and enhance trip sharing, reviewing and in-chat features of Travela. Also, we plan to implement Google authentication in the near future.

One thing that we are definitely looking forward to is expanding our database and having more users that we can gather data from. To gather more data, we are currently using Amadeus API, but our own suggestion algorithm works based on previous trips that the users have been to, and the trip that a group plans together will be stored in the database and will improve the recommendations of other completely unrelated users, which is exciting to think about. Right now, we are only working with the Amadeus's data and test data we manually enter, so our trip recommendation system doesn't have a lot to work with but as the application grows, we will be better able to see what Travela is capable of.

And lastly, we plan to release our application on both AppStore and Google Play Store. Hope the app is accepted by these platforms as soon as possible, so that our mobile users can enjoy the app as well.

9 Glossary

AWS: Amazon Web Services. A service Amazon offers providing on-demand cloud computing platforms.

EC2: Elastic Compute Cloud that Amazon offers through AWS. It provides computational resources in the cloud.

RDS: Relational Database Service that Amazon offers through AWS.

SQL: Structured Query Language, a standard language for managing relational databases.

MySQL: MySQL is an open-source relational database management system [5].

Hibernate: A Java framework to help reduce database interactions by automating some database related work

.

ORM: Object–relational mapping (ORM, O/RM, and O/R mapping tool) in computer science is a programming technique for converting data between a relational database and the heap of an object-oriented programming language. This creates, in effect, a virtual object database that can be used from within the programming language [6].

Spring Boot: Java framework used to build Spring based web applications.

Maven: A build automation tool used primarily for Java projects [7].

IATA code: The International Air Transport Association's (IATA) Location Identifier is a unique 3-letter code (also commonly known as IATA code) used in aviation and also in logistics to identify an airport [8].

IP: IP stands for "Internet Protocol," which is the set of rules governing the format of data sent via the internet or local network [9].

VPC: A virtual private cloud is a secure, isolated private cloud hosted within a public cloud [10].

API: Application Programming Interface, a set of defined rules that enable different applications to communicate with each other [11].

URL: A Uniform Resource Locator, a reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it [12].

JSON: JavaScript Object Notation, an open standard file format and data interchange format that uses human-readable text to store and transmit data objects [13].

DTO: Data Transfer Object, an object that carries data between processes [14].

Unsplash: Unsplash is a website dedicated to proprietary stock photography [15].

POST: The HTTP POST method sends data to the server [16].

GET: The HTTP GET method requests a representation of the specified resource [17].

PUT: The HTTP PUT request method creates a new resource or replaces a representation of the target resource with the request payload [18].

PNG: Portable Network Graphics, a raster-graphics file format that supports lossless data compression [19].

JAR: Java ARchive, a package file format typically used to aggregate many Java class files and associated metadata and resources into one file for distribution [20].

IntelliJ IDEA: IntelliJ IDEA is an Integrated Development Environment (IDE) for JVM languages designed to maximize developer productivity [21].

MySQL Workbench: MySQL Workbench is a unified visual tool for database architects, developers, and DBAs. MySQL Workbench provides data modeling, SQL development, and comprehensive administration tools for server configuration, user administration, backup, and much more [22].

10 References

[1] Knight, Rob. “Average person spends 10 hours planning their holiday, survey claims”. Independent UK. https://www.independent.co.uk/travel/holiday-booking-planning-travel-survey-tourist-a8801211.html. (Accessed: 15 Oct 2022)

[2] A. Inc., “Human interface guidelines,” Human Interface Guidelines - Human Interface Guidelines - Design - Apple Developer. [Online]. Available: https://developer.apple.com/design/human-interface-guidelines/guidelines/overview/. (Accessed: 17 Oct 2022).

[3] “Material design for Android : android developers,” Android Developers. [Online] Available:https://developer.android.com/develop/ui/views/theming/look-and-feel. (Accessed: 17 Oct 2022).

[4] “What is token-based authentication?,” Okta. [Online]. Available: https://www.okta.com/identity-101/what-is-token-based-authentication/. (Accessed: 17 Oct 2022).

[5] “MySQL,” Wikipedia, https://en.wikipedia.org/wiki/MySQL (accessed May 19, 2023).

[6] “Object–relational mapping,” Wikipedia, https://en.wikipedia.org/wiki/Object%E2%80%93relational_mapping (accessed May 19, 2023).

[7] “Apache maven,” Wikipedia, https://en.wikipedia.org/wiki/Apache_Maven (accessed May 19, 2023).

[8] K. K.- Nationsonline.org, “International Airport Codes - IATA 3-letter code for Airports - Nations Online Project,” International Airport Codes IATA 3-letter code for airports - Nations Online Project, https://www.nationsonline.org/oneworld/IATA_Codes/airport_code_list.htm#:~:text=The%20International%20Air%20Transport%20Association’s,logistics%20to%20identify%20an%20airport (accessed May 19, 2023).

[9] Kaspersky, “What is an IP address – definition and explanation,” www.kaspersky.com, https://www.kaspersky.com/resource-center/definitions/what-is-an-ip-address (accessed May 19, 2023).

[10] What is a virtual private cloud (VPC)? - cloudflare, https://www.cloudflare.com/learning/cloud/what-is-a-virtual-private-cloud/ (accessed May 19, 2023).

[11] “What is an application programming interface (API)?,” IBM, https://www.ibm.com/topics/api (accessed May 19, 2023).

[12] “URL,” Wikipedia, https://en.wikipedia.org/wiki/URL (accessed May 19, 2023).

[13] “JSON,” Wikipedia, https://en.wikipedia.org/wiki/JSON (accessed May 19, 2023).

[14] “Data Transfer Object,” Wikipedia, https://en.wikipedia.org/wiki/Data_transfer_object (accessed May 19, 2023).

[15] “Unsplash,” Wikipedia, https://en.wikipedia.org/wiki/Unsplash (accessed May 19, 2023).

[16] MozDevNet, “Post - http: MDN,” HTTP | MDN, https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST (accessed May 19, 2023).

[17] MozDevNet, “Get - http: MDN,” HTTP | MDN, https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/GET (accessed May 19, 2023).

[18] MozDevNet, “Put - http: MDN,” HTTP | MDN, https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PUT (accessed May 19, 2023).

[19] “PNG,” Wikipedia, https://en.wikipedia.org/wiki/PNG (accessed May 19, 2023).

[20] “Jar (file format),” Wikipedia, https://en.wikipedia.org/wiki/JAR_(file_format) (accessed May 19, 2023).

[21] “IntelliJ IDEA overview,” IntelliJ IDEA Help, https://www.jetbrains.com/help/idea/discover-intellij-idea.html (accessed May 19, 2023).

[22] “MySQL Workbench,” MySQL, https://www.mysql.com/products/workbench/ (accessed May 19, 2023).