SW Engineering CSC648/848 Section 01

Project Name: globetrottr

Team Number: 03

| Name: | Role: |
| --- | --- |
| Jay Gupta | Team Lead |
| Devin Kern | GitHub Master |
| Essa Husary | Scrum Master |
| Justin Wang | Backend Lead |
| Kevin Liu | Frontend Lead |
| Brandon Khuu | Backend/Frontend Developer |

Milestone 2

Date: 3/10/2023

History Table (Revisions)

### 1. Data Definitions V2

| Data Name | Definition | Sub-data (fields/attributes) | Usage |
| --- | --- | --- | --- |
| Query | A form that we let the user fill in when they would like to have recommendations. | * Destination * Budget * Party size * Party * Start date * End date * Hashtags * Travel means * Find savings * Max schedules to generate | * The data filled by the user will be used to generate schedules. |
| Schedule | The app’s main output unit. | * Destination * scheduleId * Total cost * Average midpoint rating * number of midpoints * overall rating * Total travel time * (array of) Events and routes * Party: array of userIDs * Party size * user\_rating * sharedWith: array of {userID, permission} objects * comments | * Schedules generated by the query will be presented to the user. The user can view each of them and decide to accept (save) them or decline (ignore) them. Schedules that are not saved by the user will be discarded. * By default, every generated schedule would have an overall rating that indicates how well the schedule matches the user’s query. This would be used for sorting the generated results. * User can rate a schedule from 1 to 5 stars. |
| Midpoint | A place where an event happens. | * type * location * name * rating * hashtags * contact information * Quick Knowledge | * Acts as a part of an event in schedules * Presents detailed information to users when they search for it. |
| Quick Knowledge | A list of quick facts of a midpoint (dos and don’ts etc.) | * (an array of) strings | * For the users to quickly lookup the facts of this midpoint |
| Search | Unlike a query, search is a single keyword that the user types into the search bar on the search page to look for schedules or midpoints. | N/A | * For the user to look up schedules or midpoints. |
| Event | Contains a midpoint, start date and end date. | * eventId * midpoint * start date * end date | * Acts as a part of schedule in conjunction with routes |
| Route | A travel mean that connects two events | * routeId * travel mean * start date * end date | * Acts as a part of schedule in conjunction with events |
| Hashtag | A single word that describes the midpoint. For the simplicity of this project, we would use a set of pre-defined hashtags. Some example hashtag can be: #pet-friendly, #relaxing, #historical | N/A | * Facilitate faster search for midpoints * To better recommend midpoints to users * Allow users to filter midpoints |
| Party | A collection of any registered users that would go over the same schedule. | * An array of users | * In the query we can extract the party members’ favorite hashtags to better recommend midpoints. |
| User | Any individual that could be planning or participating in a schedule, or simply registered to our app. | * username * userID * password * email * (array of) search history * (array of) queries * (array of) user schedules * (array of) favorite hashtags * gender * Age * Preferred language | * information like username and password would be used for logging them in * userID is for uniquely identifying users in the DB * Any schedules that the user accepts would be saved in the planned schedule collection. Users can also choose to remove schedules from it. * Schedules can be moved between past/current/planned schedule collections. |
| Guest | Any individual that’s not registered but accessing our app. | * GuestID | * Guests have limited privileges compared to registered users in terms of searching, making queries etc. |
| Schedule Collection | A collection of schedule objects. | * An array of schedules | * Schedules with a similar purpose, grouped together. |

### 2. Functional Requirements V2

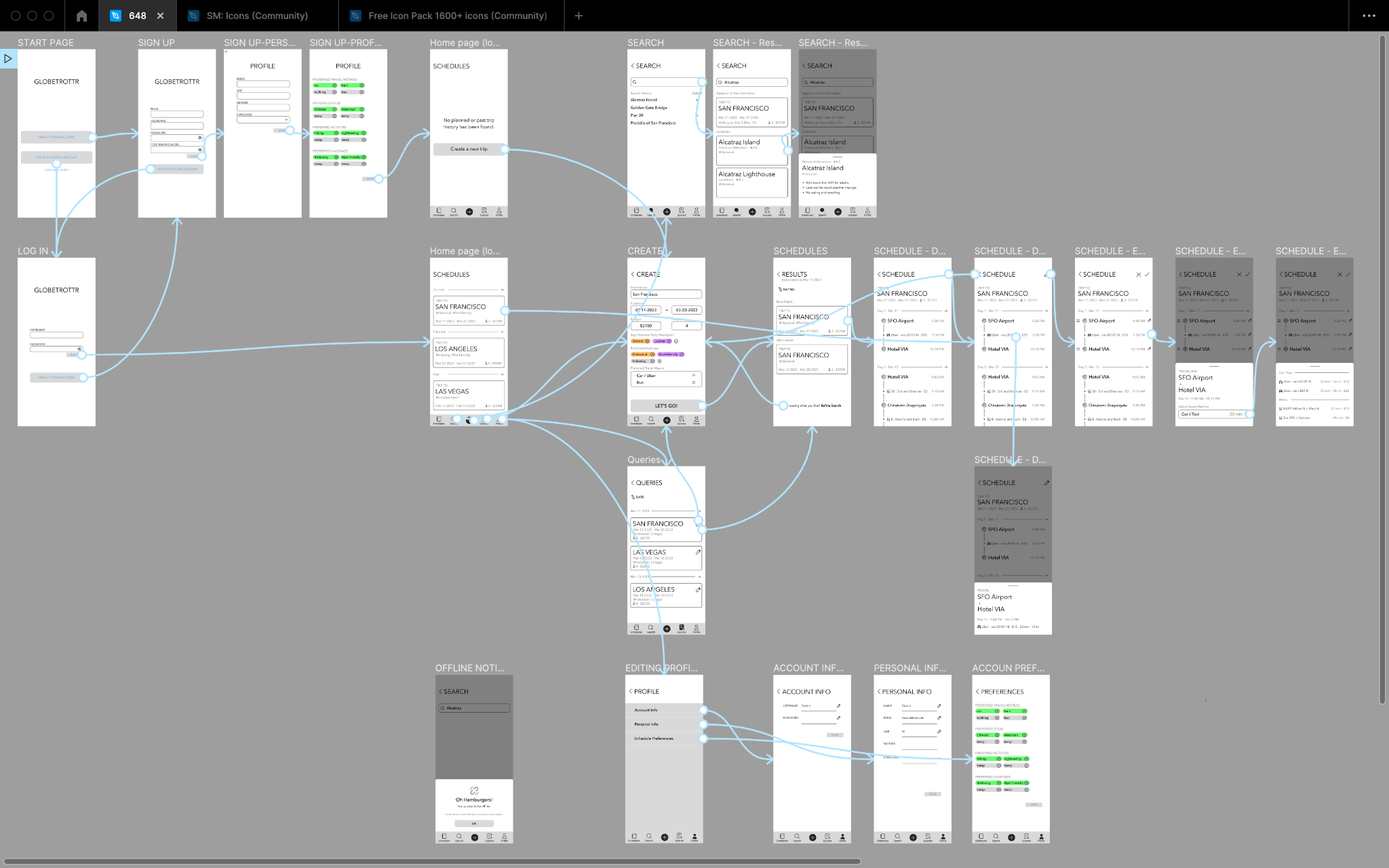
| Must-Have | Desired | Opportunistic |
| --- | --- | --- |

|  | ID | Functional Requirement Description | Details (As Needed) |
| --- | --- | --- | --- |
|  | 1 | Users are able to register for a Globetrottr account. | 1.1) User can create a new account  1.2) User can sign into an existing account  1.3) The password will have certain requirements such as length and character types that must be present, and will need to be entered twice to verify. |
|  | 2 | Guest users won’t be able to input as many variables in search as registered users will be able to. | 2.1) Registered users will be allowed to input place, budget, preferred routes, hashtags, and party size, among other variables, and will see schedules shared by other users.  2.2) Guest users will only be limited to inputting place and budget, and will not see schedules shared by other users.  2.3) Guest users will be limited to 5 searches per day. |
|  | 3 | Schedules will be sortable by a range of factors. | 3.1) Every schedule will have a variety of attributes, from overall rating to average restaurant rating.  3.2) Registered users will be able to sort schedules based on all possible attributes, while guest users will only be able to sort based on overall rating and total cost. Both will be shown the best recommended schedule, which will have the best fit with the user’s input values. |
|  | 10 | Important additional information regarding the chosen schedule will be shown to the user. | 10.1) This important information includes no-smoking, no-pets, dos and don’ts, and other policies and which midpoint/route they apply to.  10.2) Hotel & airplane bookings, restaurant reservations, and other arrangements will be conducted on 3rd party sites or instructions on how to do as such will be presented to the user. |
|  | 4 | Users can set preferences in their profile to facilitate a quicker search experience. | 4.1) Preference setting can be done during the sign-up process or by going to the user’s profile.  4.2) Preferences will include mode of travel (Uber, walk, train, etc.), favorite activities/type of food, and preferred hashtags.  4.3) These preferences will be input by default in the schedule search area; they can be overwritten if needed.  4.4) Suggested schedules from places around the world will be shown to the user based on their preferences on the recommendations page. |
|  | 17 | Search history (for queries and search terms) will be available to each user. | 17.1) The user can select whether or not they want their search history to be stored for each.  17.2) The search history will include the previous searches made, including variable values for queries, and search terms in the search area.  17.3) For queries, selecting a previous search query will input all of the values from that query into the current search fields. |
|  | 5 | Users can edit their schedules dynamically, and can request potential schedules to be created for the rest of the trip. | 5.1) Editing includes being able to replace a route/midpoint with another route/midpoint, or by extending the duration of an existing route/midpoint.  5.2) As the schedule is being edited, the attributes of the schedule, such as total cost and overall rating will be updated and shown to the user.  5.3) Users will be able to revert/redo changes and add comments to the schedule. |
|  | 7 | Users can edit their account profile or delete their account. | 7.1) The user will be able to update their name, location, description, etc.  7.2) Passwords can be reset/changed. A reset password will require an email to be sent to the user, from which they can perform the reset.  7.3) Account deletion will require the user to confirm via a checkbox on the website and by clicking a link in their email to confirm. |
|  | 8 | Schedules can be edited, viewed, and commented on by others upon permission being granted. | 8.1) A user can share their schedule with another user of the website by entering either the other person’s username or email.  8.2) Editors can edit the schedule, viewers can only view the schedule, and commenters can comment on the schedule but not make any concrete changes.  8.3) A revision history of those who edited the schedule can be viewed. A user with editing access can reset the schedule to a previous version in history. |
|  | 9 | Users can rate a schedule that they have been on. | 9.1) The rating will be from 1 to 5 stars.  9.2) This rating will be used to improve recommendations for that user. |
|  | 11 | Tutorials, tips & tooltips, and accommodations will be present throughout the app. | 11.1) A beginning tutorial on what the application can do for the user will be shown upon first visiting the site.  11.2) Disabled users will have appropriate accommodations on the site, and will be recommended activities/events that serve their needs best. Their disabilities will be part of their user profile. |
|  | 14 | Users can import their calendars from other third party applications. | 14.1) Google Calendar, iCalendar, and other apps can be imported from.  14.2) The scheduling algorithm will recommend schedules that fit around the preexisting events on the person’s calendar. |
|  | 6 | A machine learning algorithm will be used to recommend schedules to users. | 6.1) Based on the user’s choice of schedules to go on, their preferences, and similar users’ choices, this algorithm will be used to understand how likely they are to use a certain schedule, and make recommendations using it.  6.2) The user can decide to include machine learning results in their searches. |
|  | 12 | Users will be able to find other users in their area, or in a place that they are visiting. | 12.1) This feature will be for users that allow their location to be shown.  12.2) A user can send a message of less than 150 characters as part of a friend request; if this request is accepted, the two users can chat.  12.3) Users will be notified via email and on the webapp if they are being contacted. They can choose to accept, ignore, or block the other user. Notifications will also be sent if a friend request is accepted. |
|  | 13 | Advanced features of chat & friend groups. | 13.1) A user can invite multiple friends to have a group chat room.  13.2) When creating a schedule, a user can invite their friends/friend group as a party.  13.3) Messages can be unsent, hidden, reported, and replied to directly (in a group discussion).  13.4) Users can create multiple chat rooms for different friends/friend groups. |
|  | 15 | A virtual tour of a schedule will be available to the user. | 15.1) This virtual tour will show the user the street view of the places they will see, as well as the routes that they will take to get from place to place.  15.2) The virtual tour will be viewable in contemporary VR technologies such as Oculus.  15.3) Videos & pictures of midpoints submitted by other users or on the web will be shown as part of this virtual tour. |
|  | 16 | The website will be available in a variety of languages. | 16.1) The user can select their preferred language in settings.  16.2) Videos and text will be displayed in the user’s preferred language. |

### 3. UI Mockups and UX Flows

The Full Figma Project:

<https://www.figma.com/file/R616QJzr2ckFANrEUiEGxT/648?node-id=0%3A1&t=gXgWPaW84yrxBKmt-1>



### 4. High Level Architecture, Database Organization

Schedules:

{

destination: string,

scheduleId: int,

totalCost: float,

averageMidpointRating: float,

numberOfMidpoints: int,

overallRating: float,

totalTravelTime: int,

eventsAndRoutes: [array of event and route IDs],

party: [array of userids],

partySize: int,

userRating: float,

sharedWith: [

{

userId: string,

permission: string

},

{

userId: string.

permission: string

},

...

]

}

Operations permitted on Schedules:

add/search/deleted/displayed: Schedules are displayed to the User, the User adds a Schedule to the collection, and Schedules can be searched

Users:

{

userId: int,

username: string,

password: string,

email: string,

location: string

searchHistory: [array of strings],

queryHistory: [

{

destination: string,

budget: int,

...

},

{

destination: string,

budget: int,

...

},

...

],  
userDescription: {

age: int,

gender: character,

preferredLanguage: string,

favoriteHashtags: [array of strings],

aboutMe: string

},

schedules: [array of scheduleIds]

}

Operations permitted on Users:

add/search/deleted/displayed: Users added when they are registered, deleted when they delete their account. Users can search for other users (other userIds to add them to a Schedule party) and display the result of their search.

Events:

{

eventId: string,

midpoint: {

type: string,

location: string,

rating: float,

hashtags: [array of strings],

contactInformation: string,

quickKnowledge: [array of strings]

},

startDate: Date,

endDate: Date

}

operations permitted on Events:

add/search/displayed: Events can be searched, displayed, and added to Schedules by Users.

Routes:

{

routeId: string,

travelMean: string,

startDate: Date,

endDate: Date

}

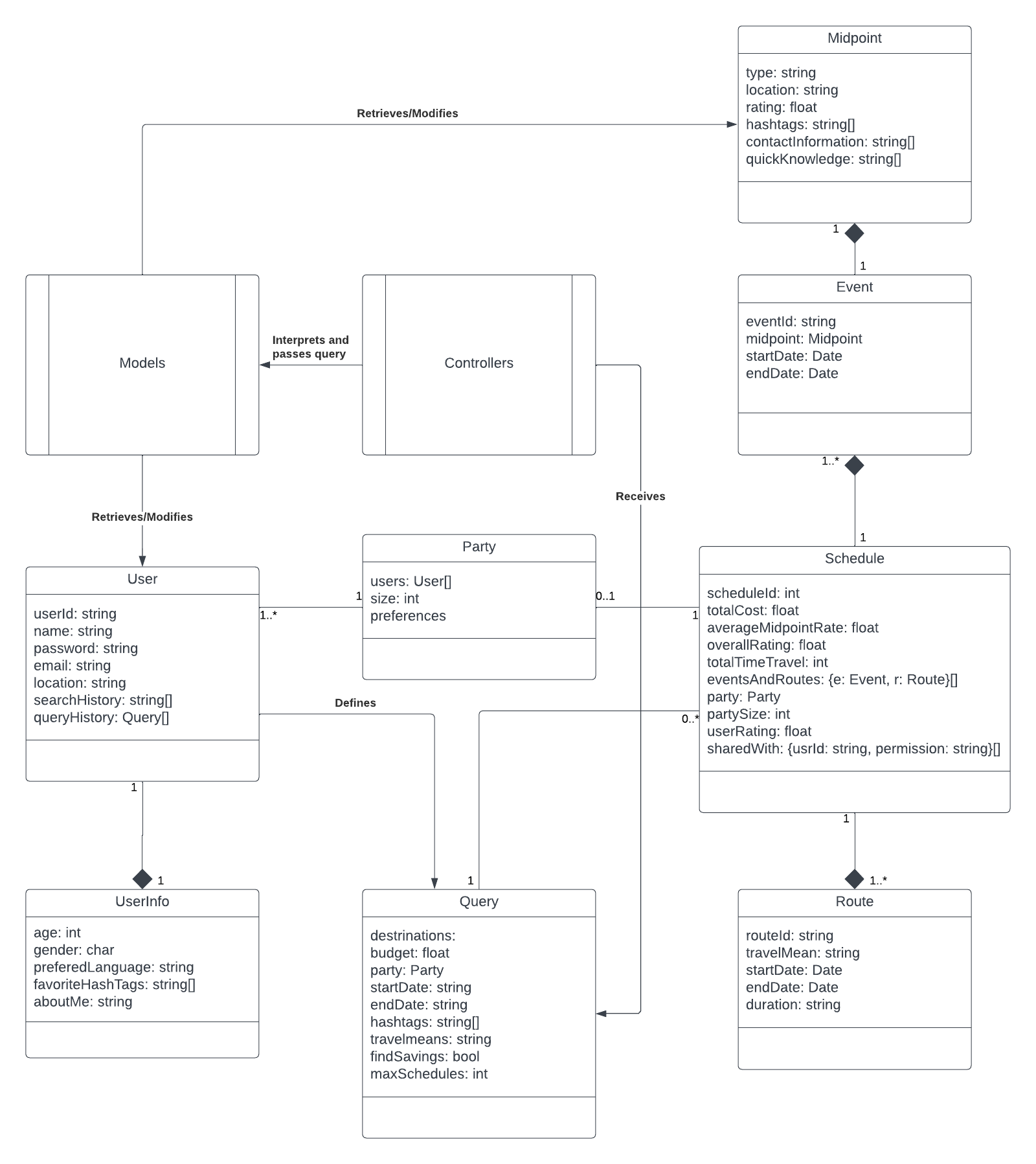
Operations permitted on Routes:

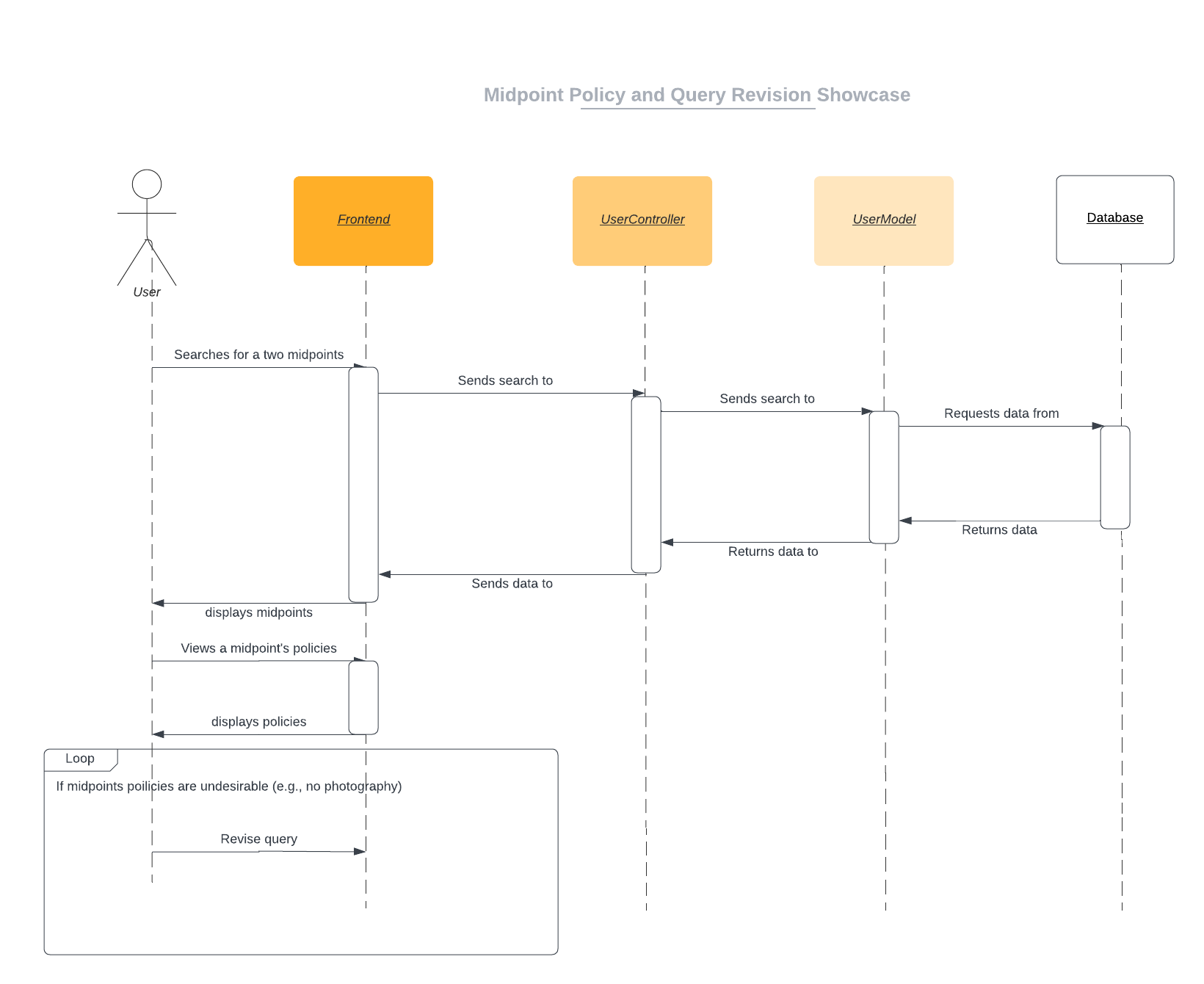
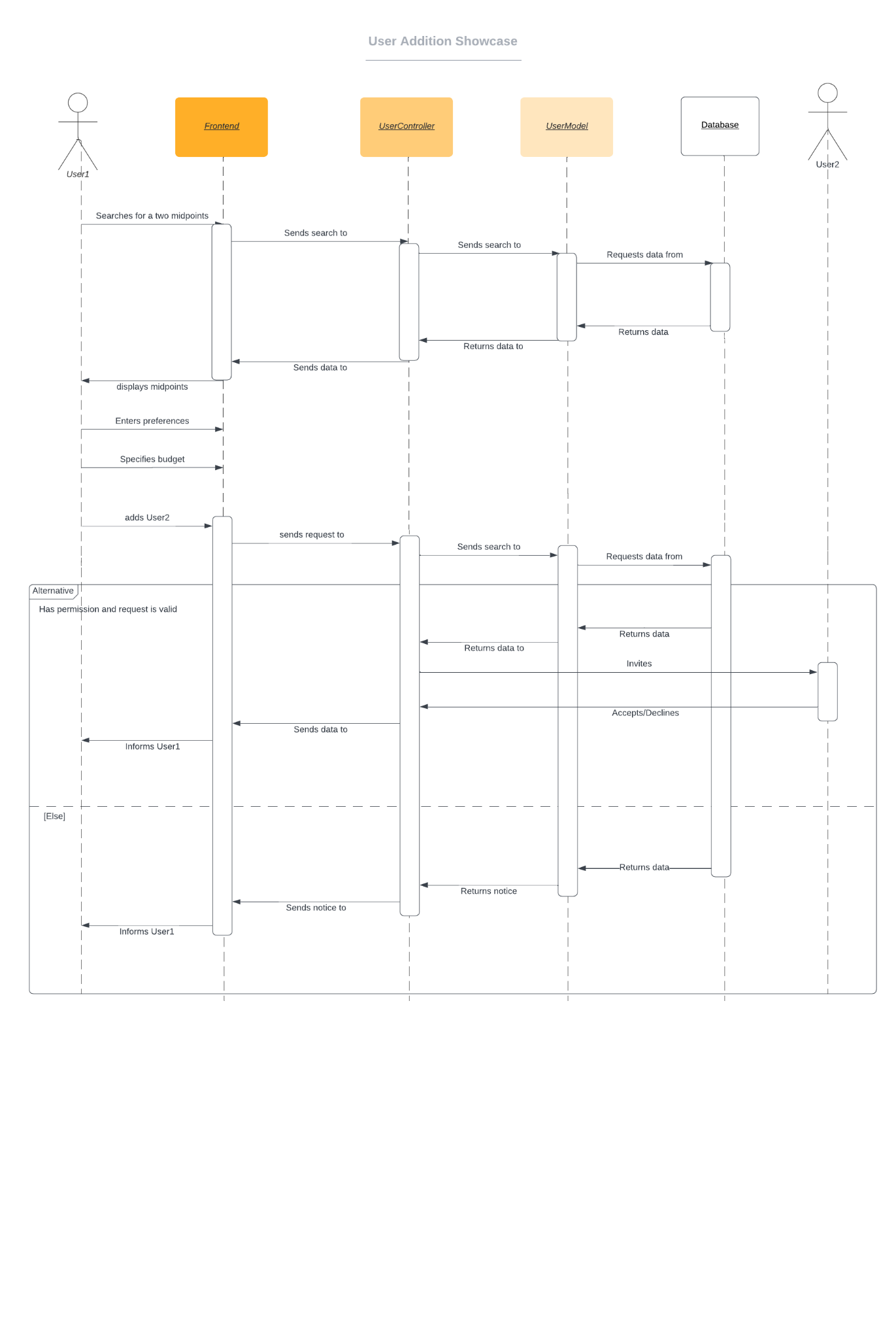
add/search/displayed: Routes can be searched, displayed, and added to Schedules by Users.

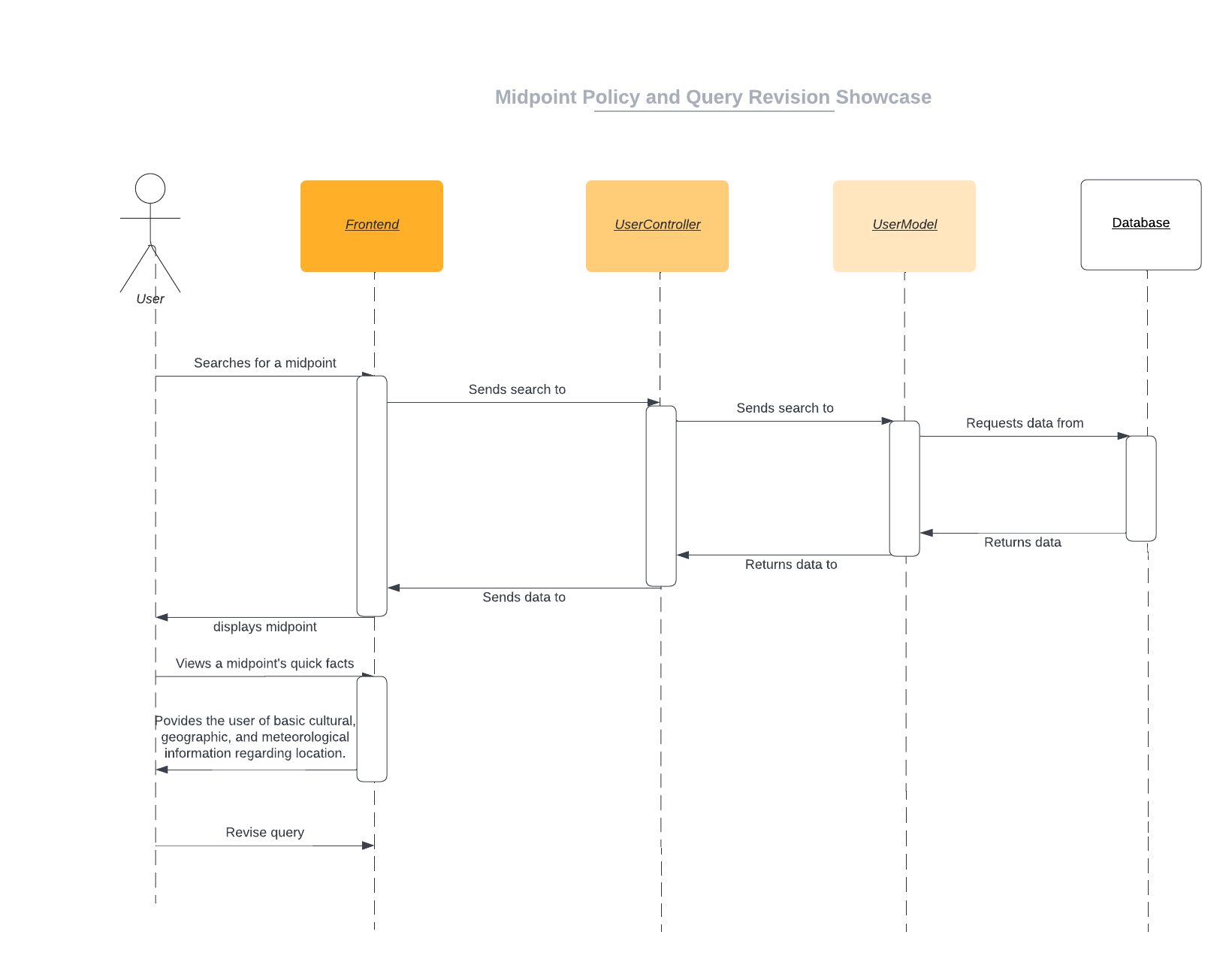
APIs:

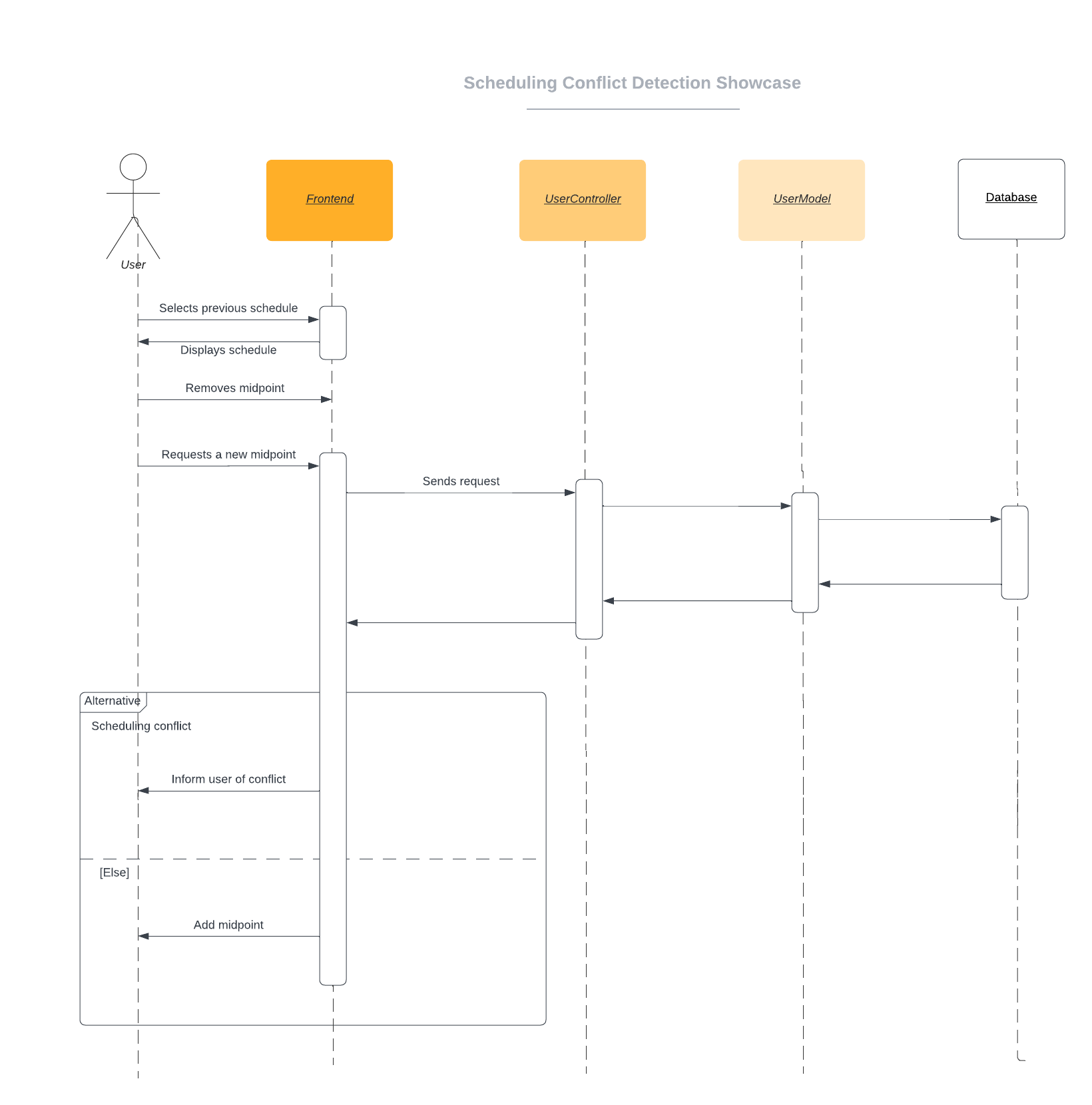
* Our own APIs:
  + to sign up users:
    - request: post request with username, email, and password of new user along with other user data
    - response: new user in database confirmation, and new user’s profile page
    - route function: creates the user in the database, sends back their userID
  + to log in users:
    - request: post request with username and password
    - response: confirmation if their credentials succeeded, error if it didn’t
    - route function: checks the credentials in the database, sends boolean of true if successful
  + delete user:
    - request: delete request with userID
    - response: confirmation if delete succeeded, error if it didn’t
    - route function: deletes the user from the database, sends boolean of true if successful
  + update user profile:
    - request: patch request with updated fields
    - response: confirmation if update succeeded
    - route function: updates the user profile using the new values, sends boolean of true if successful
  + generate schedules:
    - request: post request with query fields and values
    - response: list of schedules generated
    - route function: uses request values to contact external 3rd party APIs and other functions in the program to come up with schedules that the user might like; sends back these schedules if successful
  + get search history:
    - request: get request with user id
    - response: user’s search history (for either search terms or previous queries)
    - route function: performs a get request to get the user’s entry in the database; returns the user’s search history from within that entry
  + edit schedule:
    - request: put request with updated schedule events, routes, and attributes
    - response: confirmation if put succeeded
    - route function: replaces the schedule’s events and routes array in the database with the new one; recalculates and replaces attributes in the schedule object
  + post comment on schedule:
    - request: post request with comment and schedule ID
    - response: confirmation if post succeeded
    - route function: finds the schedule in the database; adds comment to its comments attribute
  + add to party:
    - request: post request with schedule ID of schedule that contains the party, user ID of user being added to party
    - response: confirmation if post succeeded
    - route function: gets the schedule with that schedule ID; adds user to the schedule’s party
* 3rd Party APIs:
  + “TripAdvisor” API:
    - Will use in our travel guide app to help user attain information such as reviews, policies, flight information, etc. about restaurants, hotels, airports, and attractions.
  + “Travel Guide” API:
    - We’ll use this API for the same purposes as the TripAdvisor API, but with additional functionalities such as having a map that the user can use to navigate to view popular restaurants, venues, etc..
  + “Booking.com” API:
    - Will use in our app to help the user understand information about hotels that they want to book into such as policies, reviews, descriptions, room listings, tips, and photos. The API also features the ability for the user to filter searches and have autocomplete searches as well as the ability to book vehicle rentals which may come in handy.
  + “Route and directions” API:
    - Will help the user navigate from one point to another in the most efficient way possible, and includes the modes of transportation as well as the predicted times it’ll take to travel via those modes of transportation. Similar functionalities to Google’s “Google Maps” and Apple’s “Maps”.
  + “TrueWay Matrix” API:
    - Acts the same way as the “Route and directions” API but allows filters such as avoiding freeways, tolls, and ferries.

### 5. High Level UML Diagrams









### 6. Identify Actual Key Risks for your Project at this Time

* skills risk and mitigation plan:
  + If a certain technology is difficult to understand, or its use in the current workspace is unclear, it could delay the expected completion time of the current milestone.
    - to resolve:
      * plan and research before choosing a technology
      * start early
      * use resources to make oneself unstuck; don’t linger on the codebase for too long
  + If a significant blocker is encountered in the codebase, the feature may not be delivered, and it could delay the expected completion time of the current milestone.
    - to resolve:
      * understand the codebase to a reasonable extent before trying to work on it or extend its features
      * use debuggers and other tools designed for working with bugs
      * use pair programming to get another set of eyes on the code
      * research for solutions in documentation or Stack Overflow (verify these solutions in the documentation, and understand them thoroughly before utilizing them)
  + If certain team members don’t learn the way the study plan advises, it could cause a discrepancy in the knowledge of each team member about the technology.
    - to resolve:
      * ensure that the study plan has a variety of learning methods, such as visual, textual, and hands-on
      * encourage each member to complete the study guide the way it is advised, or to suggest changes for the whole group to follow
      * compile a quick reference guide while going through the study plan that each member can go to in order to get the gist of the technology
* schedule risks:
  + If a member doesn’t communicate their progress on a task with the rest of the team, it could erode team culture and cause significant delays.
    - to resolve:
      * have every member speak during stand-up
      * check in with members throughout the week; be open to answering questions from other members
  + If a member makes a change outside of their responsibilities in the codebase/architecture without informing the rest of the group, it could cause the team confusion later on in the development stage.
    - to resolve:
      * make member responsibilities clear in the beginning
      * if a member wants to change something/propose a change for a feature outside of their control, they should bring it up with the person in charge of that feature, or bring it up during the meeting
  + If a project management tool isn’t used, the roadmap to completing a milestone may be unclear to a member.
    - to resolve:
      * use a tool like Trello to keep track of member tasks and sprint goals
      * the team lead should make a plan for the milestone and next sprints in the team meetings and gain approval from the group; this plan should be accessible to anyone in the group at any time
* teamwork risks:
  + If some team members have more experience in a technology than others, it may cause an imbalance where the more experienced member is doing a lot more of the work
    - to resolve:
      * before anything, understand where everyone is at in their understanding of technologies and general knowledge
      * compile “crash-course” resources that the less experienced member(s) may use to get to a state where they can contribute effectively
      * plan such that each member will do their fair share of the work of the overall project
  + If a team member consistently misses team meetings, they may not know what their tasks are, or they will require other team members to constantly assist them in doing their part of the milestone.
    - to resolve:
      * ensure that meetings are at time that works for everyone
      * have the Scrum master take detailed notes about what occurred during each meeting
  + If a member on the front-end of the code doesn’t know how code works on the back-end, and vice-versa, it could be an issue for each to build their systems to be compatible with the other side’s.
    - to resolve:
      * have the front-end team and back-end team working closely together on each deliverable, where appropriate
      * keep documented standards and informational guides, such that anyone on the team can know how any part of the system works in the codebase/architecture
      * ask questions and be curious during meetings, even if your work may not seem directly related
* legal/content risks:
  + If content/SW is not obtained legally/used with consent from the original creator, it could be seen as plagiarism or as a copyright violation.
    - to resolve:
      * before using work that isn’t ours, ensure that we have the proper rights to use it
      * if acknowledgement is needed, reference the original creator and where the code/material was taken from
  + If the user lives in a region where their data is protected and their privacy is not respected, it could get our application blacklisted from that region.
    - to resolve:
      * have a privacy policy in place
      * ensure consent is gotten from users before using their data for a specific purpose
      * make sure user data is safe from malicious actors by using encryption methods and following industry standards

### 7. Project Management

Our team consists of a team lead, 2 front-end developers, and 3 back-end developers. The front-end developers worked on the UI mockups and UI flows, while the back-end developers and the team lead worked on the rest of the milestone. Each section apart from the vertical software prototype had either the team lead working on it or 2 back-end developers; this was done such that the burden of a section was not on one person. We first decided to get the document part of the milestone done, as well as ensure that the architecture and general requirements for the project were reviewed by the team. Then, once we got a draft down, we wanted to work on the vertical software prototype as a team, with the front-end developers working on the www page, and the backend developers working on the server and database aspects.

During each scrum meeting, the team lead specifies the agenda, ensuring that every person has a chance to talk about their section and any potential blockers that they are facing. Each member is given the chance to ask questions, raise concerns, and give new ideas. The screen is almost always shared, such that there is a visual for the team to see. These meetings were held during the weekly Monday time, as well as the Friday time set by the team. Having these days far apart made it possible to assign tasks on Monday and review the team’s output on Friday. For this milestone, the team lead assigned sections to groups of people, using their role and experience as gauges of where they fit best.

To manage each member’s tasks, we used Trello. Trello allows us to have an Agile card-like format, with sections for backlog, to-do, in-progress, review, QA, ready for deployment, and done. As we complete tasks, we move these cards from section to section throughout the sprint. This makes it easy for each team member to track their individual tasks, as well as the tasks of the team. Having this format mitigates the issues of members not knowing what to do or doing something someone else is already doing; at a glance, one can understand the team’s progress in the sprint. We use Discord as our main communication platform: it is used for sharing resources, asking questions, and conducting our weekly meetings.