SW Engineering CSC648/848 Summer 2021
July 8, 2021

Globetrotter Travel Assistant

Milestone 2

Team 2

Role	Name	Email
Team Lead/Github Master	Taylor Artunian	tartunian@mail.sfsu.edu
Front-End Lead	Yi Liu	
Back-End Lead	Luis Alfaro	
Front End	Ruja Rajbhandari	
	Kajeme Cheneque	
	Jesus Correa	
	Robin Fernando	

Revision History

Version	Date Submitted	Date Revised
M2V2	07/22/21	
M2V1	07/08/21	07/22/21
M1V2	07/08/21	
M1V1	06/22/21	07/08/22

Table of Contents

Data Definitions

Entity Name		Entity Description					
	Column Name	**Column Description**	**Data Type**	**Length**	**Primary Key**	**Nullable**	**Unique**
activity		This represents something that a user does during a trip or is planning to do.					
	activity_id	The id of the activity.	varchar	36	true	false	true
	end_location	The id of the location at which the activity ends.	varchar	36	false	true	false
	end_time	The timestamp at which the activity ends.	timestamp	19	false	false	false
	start_location	The id of the location at which the activity starts.	varchar	36	false	false	false
	start_time	The timestamp at which the activity starts.	timestamp	19	false	false	false
	trip	The id of the trip that the activity is part of.	varchar	36	true	false	false
	trip_owner	The id of the trip's owner.	varchar	36	true	false	false
airline		A type of service_provider that sells flights.					
	airline_code	The IATA code of the airline.	varchar	7	true	false	true

	service_provider	The id of the service_provider that the airline represents.	varchar	36	true	false	true
airport		A type of location that belongs to an airport.					
	iata_code	The IATA code of the airport.	varchar	3	true	false	true
	location	The id of the location of the airport.	varchar	36	true	false	true
currency_code		A collection of codes for currencies.					
	code	The ISO 4217 code for the currency.	varchar	3	true	false	false
	name	The name of the currency.	varchar	255	false	false	false
file		This is a file uploaded by a registered_user to our web server.					
	extension	The extension of the file.	varchar	5	false	true	false
	file_id	The unique id of the file.	varchar	36	true	false	true
	file_name	The name of the file.	varchar	255	false	false	false
	folder_path	The folder path of the file.	varchar	255	false	false	false
	full_file_path	The complete path of the file.	varchar	255	false	false	true
	owner	The owner of the fil.	varchar	36	true	false	false
flight_activity		This is a type of activity that represents a flight in the user's trip.					
	activity	The id of the activity.	varchar	36	true	false	true

	flight_offer_json_data	The json data of the selected offer from the airline.	varchar	4095	false	false	false
	trip	The id of the trip.	varchar	36	true	false	false
	trip_owner	The id of the trip owner.	varchar	36	true	false	false
guest_to_trip_association		Associates a user to a trip as a guest.					
	guest	The id of the guest's user entity.	varchar	36	true	false	false
	trip	The id of the trip.	varchar	36	true	false	false
	trip_owner	The id of the trip owner.	varchar	36	true	false	false
home_location		Associates a user with a location as their home.					
	location	The id of the location which represents the registered_user's home.	varchar	36	true	false	false
	user	The id of registered_user which the home_location belongs to.	varchar	36	true	false	true
hotel_activity		This is a type of activity which holds information about a planned hotel stay.					
	activity	The id of the activity.	varchar	36	true	false	true
	hotel_offer_json_data	The json data of the selected offer from the hotel.	varchar	4095	false	false	false
	trip	The id of the trip.	varchar	36	true	false	false

	trip_owner	The id of the owner.	varchar	36	true	false	false
location		An address, a GPS coordinate or both.					
	city	The name of the city.	varchar	255	false	true	false
	country	The name of the country.	varchar	255	false	true	false
	latitude	The GPS latitude coordinate.	decimal	10.7	false	true	false
	location_id	The id of the location.	varchar	36	true	false	false
	location_name	The name of the location.	varchar	255	false	true	false
	longitude	The GPS longitude coordinate.	decimal	10.7	false	true	false
	postal_code	The postal code.	varchar	255	false	true	false
	state	The name of the state.	varchar	255	false	true	false
photo		This is a type of file which is also an image.					
	description	The description of the photo.	varchar	255	false	false	false
	file	The file which the photo represents.	varchar	36	true	false	false
	file_owner	The owner of the file.	varchar	36	true	false	false
	is_profile_photo	Tells whether the photo is a registered_user's profile photo.	bit	1	false	true	false
	photo_id	The unique id of the photo.	varchar	36	true	false	true

	title	The title of the photo.	varchar	255	false	false	false
photo_album		A photo album is created for each trip and is owned by a user. It is a collection of photos.					
	photo_album_id	The unique id of the photo.	int	10	true	false	false
	trip	The id of the trip which the photo_album belongs to.	varchar	36	true	false	true
	trip_owner	The id of the owner of the trip.	varchar	36	true	false	false
photo_to_photo_album_association		Associates a photo to a photo_album.					
	owner	The owner of the photo file.	varchar	36	true	false	false
	photo	The photo which is part of the album.	varchar	36	true	false	false
	photo_album	The album which the photo is part of.	varchar	36	true	false	false
registered_user		User that has registered. They may own trips, files, photos, photo albums, and reviews.					
	password_hashed	The bcrypt hash of the registered_user's password.	varchar	255	false	false	false
	preferred_currency	The registered_user's preferred currency.	varchar	3	false	true	false
	primary_phone_country_code	The registered_user's primary phone country code.	varchar	2	false	true	false

	primary_phone_number	The registered_user's primary phone number.	varchar	10	false	true	false
	secondary_phone_country_code	The registered_user's primary phone number.	varchar	2	false	true	false
	secondary_phone_number	The registered_user's secondary phone number.	varchar	10	false	true	false
	user	The id of the user entity.	varchar	36	true	false	true
	username	The username of the registered_user.	varchar	255	false	false	true
review		A review of a service_provider written by a registered_user.					
	content	The text content of the review.	varchar	255	false	false	false
	reviewer	The id of the registered_user who made the review.	varchar	36	true	false	false
	service_provider	The id of the service_provider that the review is about.	varchar	36	true	false	false
service_provider		A company that sells services to users such as an airline.					
	location	The id of the service_provider's location.	varchar	36	false	true	false
	service_provider_id	The id of the service provider.	varchar	36	true	false	false
	service_provider_name	The name of the service_provider.	varchar	255	false	false	false

service_record	This is a record of a service purchased from a service provider such as a flight or hotel reservation.						
	activity	The id of the activity.	varchar	36	true	false	false
	json_data	The json data provided by the service_provider.	varchar	510	false	true	false
	service_provider	The id of the service_provider.	varchar	36	true	false	false
	service_record_id	The id of the service_record.	varchar	36	true	false	true
	trip	The id of the trip.	varchar	36	true	false	false
	trip_owner	The id of the trip owner.	varchar	36	true	false	false
trip	A trip holds activities and is named.						
	owner	The id of the trip's owner.	varchar	36	true	false	false
	trip_id	The unique id of the trip.	varchar	36	true	false	true
	trip_name	The unique name of the trip.	varchar	255	false	false	true
user	Unregistered user. They may be guests of trips.						
	email	The unique email of the user.	varchar	255	false	false	true
	user_id	The unique id of the user.	varchar	36	true	false	true
usersessions	Holds session information for logged in users.						

	data	The data associated with the logged in user's session.	mediumtext	0	false	true	false
	expires	The timestamp of when the session expires.	int	10	false	false	false
	session_id	The id of the user_session.	varchar	128	true	false	false

Prioritized Functional Requirements

Priority 1

1. Registered users can edit their birthday.
2. Registered users can edit their email.
3. Registered users can edit their home city
4. Registered users can edit their home postal code.
5. Registered users can edit their home state
6. Registered users can edit their home street address.
7. Registered users can edit their password.
8. Registered users can edit their preferred currency.
9. Registered users can edit their primary phone number.
10. Registered users can edit their profile photo.
11. Registered users can edit their secondary phone number.
12. Registered users can edit their username.
13. Registered users can view the ‘Account Management’ page.
14. Unregistered users cannot access Account Management.
15. All registered users can view the cost of the trip when checking out.
16. All registered users can view the flights they are purchasing when checking out.
17. Unregistered users cannot access Checkout.
18. Registered users can login using the ‘Login’ page.
19. Users shall provide their password when logging in.
20. Users shall provide their username when logging in.
21. Registered users can reset their password from the ‘Login’ page by sending a reset link to the account’s email.
22. Unregistered users can view the ‘Registration’ page.
23. Login page will have a ‘forgot password?’ button.
24. The password field will be obscured on the Login page.
25. All users can contact the team.
26. All users can search for flights.
27. All users can view the “Meet the Team” page to find out more information about the team.
28. Registered users can delete photos they uploaded.
29. Registered users can upload pictures of their trip.
30. Unregistered users cannot access Photo Albums.
31. Unregistered users cannot access Previous Trips.
32. Unregistered Users shall provide their date of birth when creating an account
33. Unregistered users shall provide a password when creating an account.
34. Unregistered users shall provide a primary phone number when creating an account.
35. Unregistered users shall provide a username when creating an account.
36. Unregistered users shall provide their email when creating an account.

37. Unregistered users shall provide their first name when creating an account.
38. Unregistered users shall provide their home city when creating an account.
39. Unregistered users shall provide their home postal code when creating an account.
40. Unregistered users shall provide their home state when creating an account.
41. Unregistered users shall provide their home street address when creating an account.
42. Unregistered users shall provide their last name when creating an account.
43. Unregistered users will be able to create an account.
44. All registered users can add at least one destination.
45. All registered users can add multiple Activities to a Trip.
46. All users can choose from the available flights.
47. All registered users can edit the destination of a trip.
48. All registered users can edit the starting location of a trip.
49. All users can edit the displayed currency.
50. All registered users can remove the destination of a trip.
51. All registered users can remove the starting location of a trip.
52. All registered users can remove Activities from a Trip.
53. All Registered Users can specify a return date.
54. All registered users shall specify the dates of departure for each flight.
55. All users will be able to view arrival times for the available flights.
56. Unregistered users can access the Route Planner.
57. Registered users can delete trips that they previously created.
58. Registered users can edit trips that they previously created.
59. Registered users can purchase their Saved Trips.
60. Unregistered users cannot access Saved Trips.
61. Registered users can view trips that they previously created.
62. The displayed currency will default to the preferred currency of a logged in user.
63. The website will be easy for all Users to maneuver through.

Priority 2

64. Unregistered users shall provide their gender when creating an account.
65. Registered guests may purchase trips on which they are guests.
66. Registered guests will receive a confirmation email after checkout.
67. All guests added to the trip will be sent an email containing a link to the trip.
68. All users will be able to click on individual team images to find out more information about the specific member.
69. Registered users can add a photo description when uploading a picture.
70. Registered users can edit their photo descriptions.
71. Registered users can edit titles of pictures they uploaded.
72. Registered users shall give a title when uploading a picture.
73. Users will be sent a confirmation email after completing registration.
74. Registered users can create a review of their flight.
75. Registered users can delete their reviews.
76. Registered users can edit their reviews.

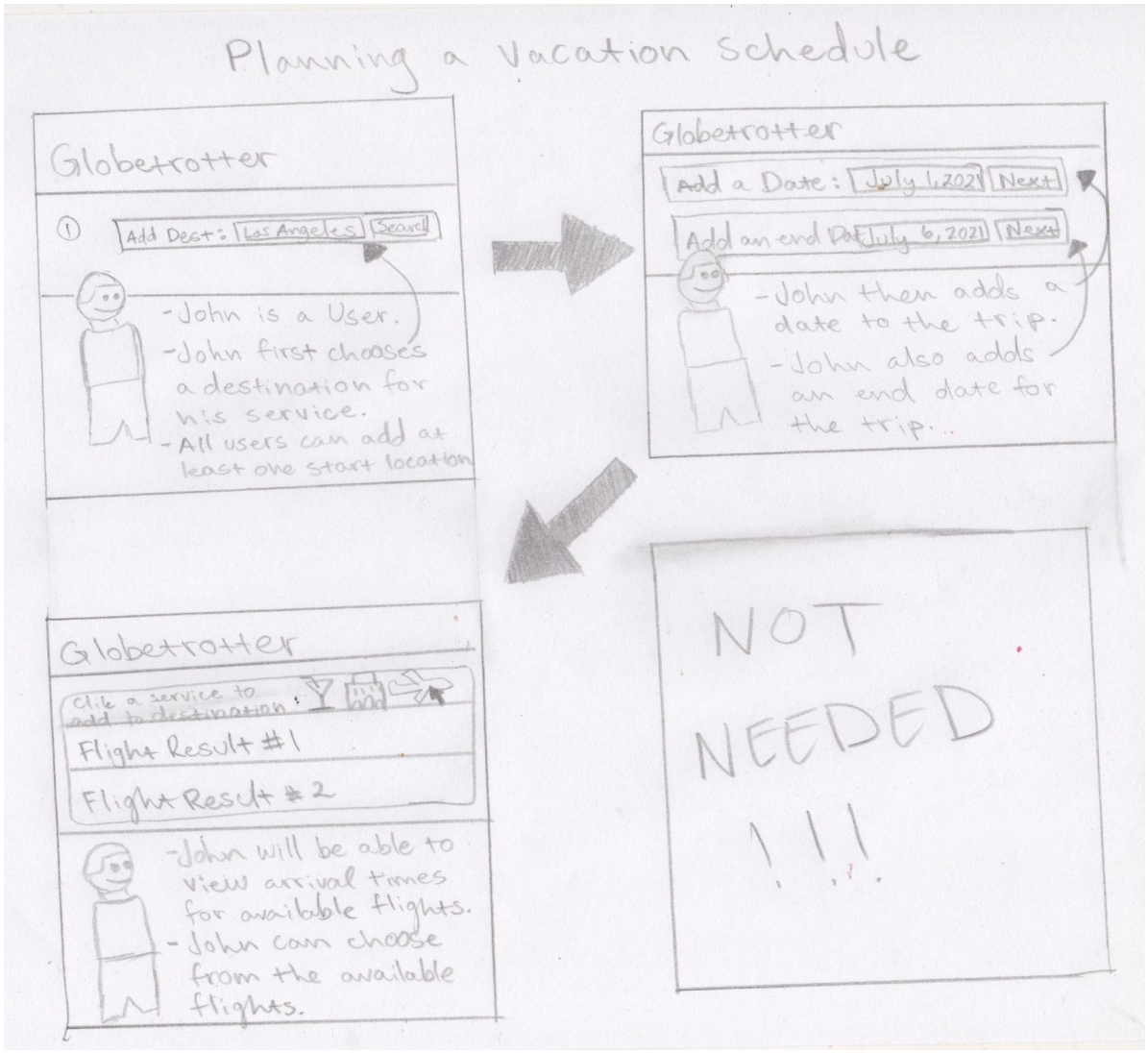
77. All users can look at reviews of Airlines.
78. All guests can view trips on which they are guests.
79. All users may add entertainment activities to a particular day of a trip.
80. All users may add hotel reservations to a particular day of a trip.
81. All users will be able to view weather information for each location.
82. Dates added to a trip may not overlap.
83. Registered users can edit trips on which they are guests.
84. Registered users may cancel flights which they purchased.
85. Registered users can be issued a refund for flights cancelled within 48 hours of departure.
86. Registered users can receive credit for flights cancelled within 48 hours of departure.
87. Users may export the schedule prepared by Globetrotter to a file.
88. Users may print the schedule prepared by Globetrotter.

Priority 3

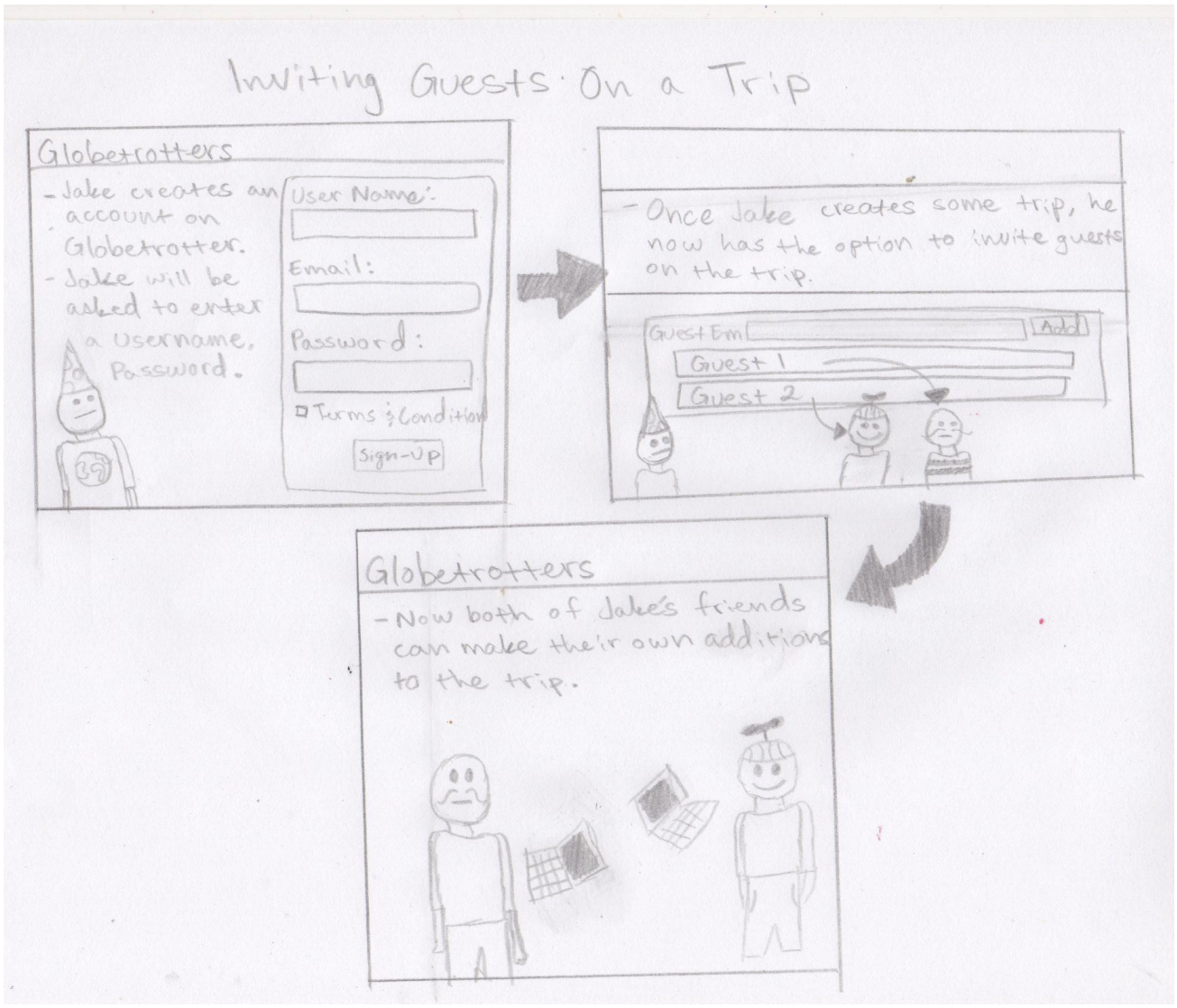
89. All registered guests can add guests to the trip.
90. All registered guests can add multiple flight stops in between the start and destination locations (waypoints).
91. All registered guests can add multiple flight stops in between the start and destination locations.
92. All registered guests can add other registered users to the trip.
93. All registered guests can add unregistered guests to the trip.
94. All registered guests can edit the destination of a trip.
95. All registered guests can edit the starting location of a trip.
96. All registered guests shall specify the dates of departure.
97. All registered guests can only add up to nine registered users to the trip.
98. All registered guests can only add up to one registered user at a time.
99. All registered guests can remove the destination of a trip.
100. All registered guests can remove the starting location of a trip.
101. All registered guests can remove waypoints.
102. Registered users will receive a confirmation email after checkout.
103. Unregistered guests may not purchase trips on which they are guests.
104. All registered guests can add at least one starting location.
105. All users shall specify a budget.
106. Registered users can reserve other registered guests to hotels.

UI Mockups & Storyboards

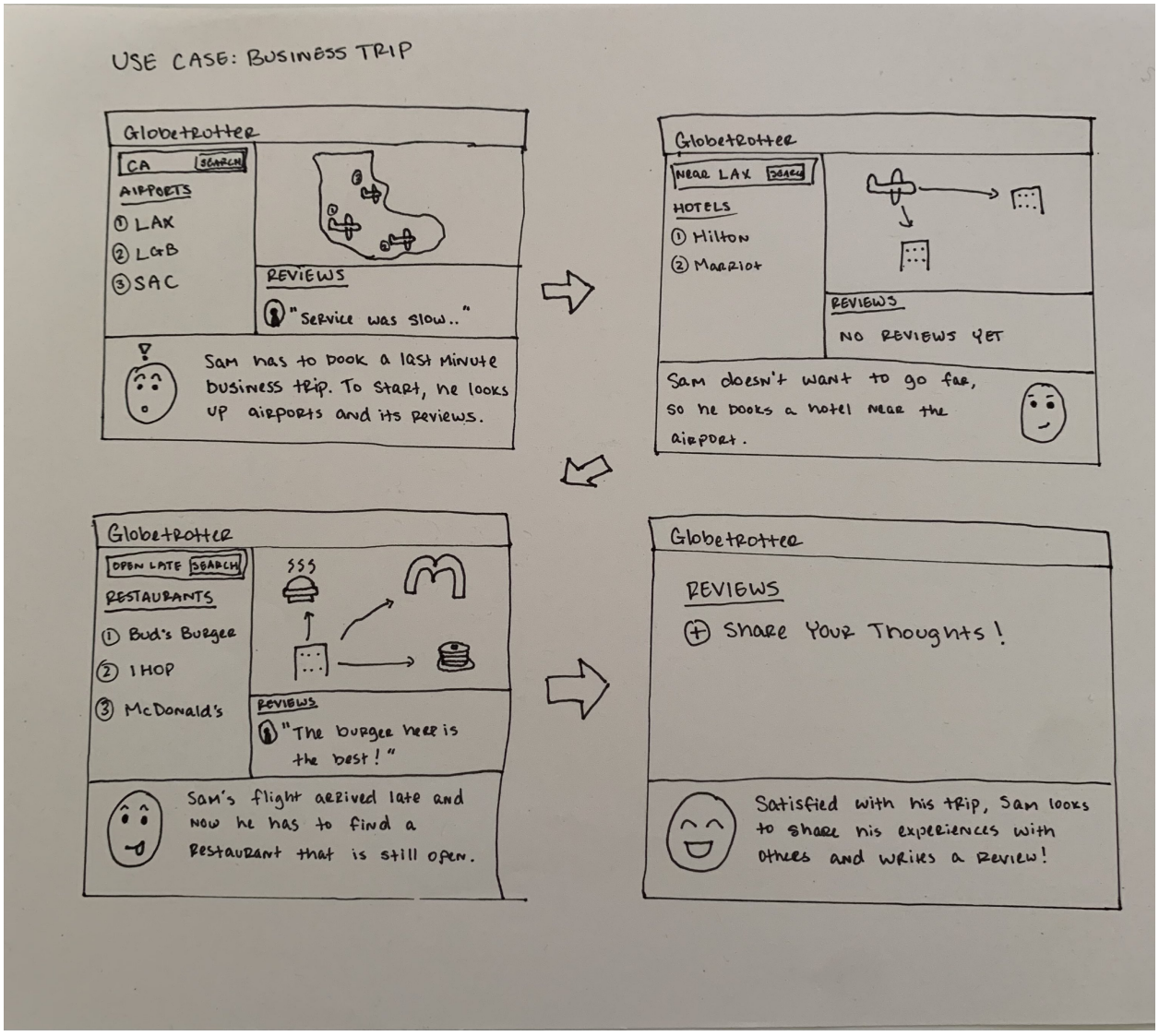
Planning a Vacation Schedule



Inviting Guests on a Trip

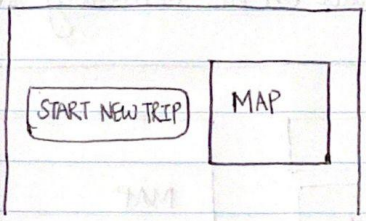


Business Trip

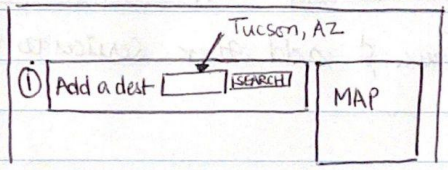


Activity Itinerary (Focused on booking/adding flights to itinerary)

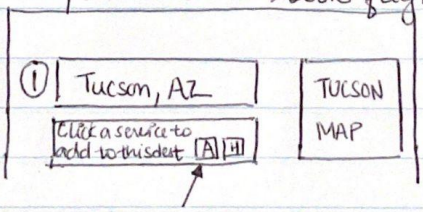
Bill creates an account & starts creating a new trip.



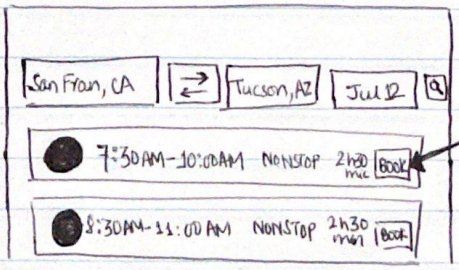
Bill is taken to a new trip screen, where he will enter Tucson, AZ as the destination



After adding Tucson as destination, Bill will be shown the map of Tucson & he will get options to click on services he wants to add for that destination. He clicks the airplane icon to ~~book~~ for flights to the destination (Tucson)

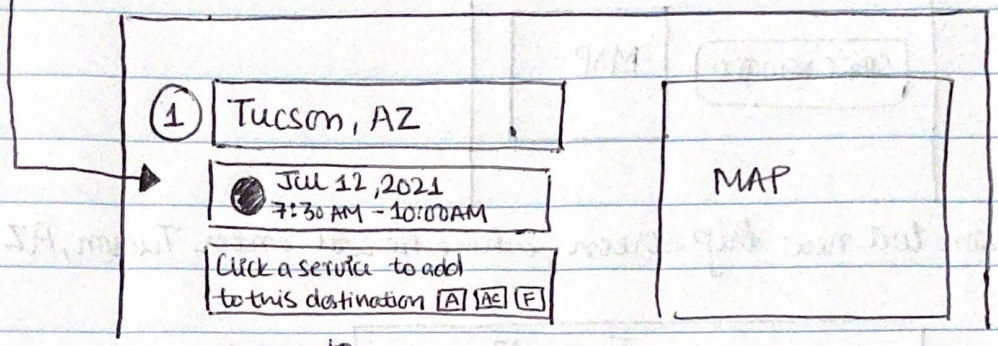


He is taken to add a flight screen, where he will get to enter the arrival & departure destination & dates. He will be able to see all the available flights & he can choose to book flights from the ~~results~~ displayed.



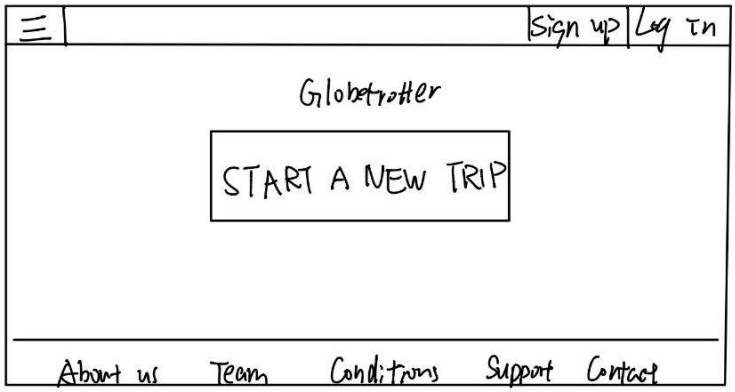
He books the 7:30AM flight.

After he books his desired flight, he will be able to see the flight with details on his itinerary screen.



Now Bill can choose to continue & add other services to his trip.

Login/Sign-Up Process



Main Page

Sign up Page

Creating Account

First name
Last name
Address :
State :
Zip :
County:

Account number:
password:
verify password:

email address :

☐ show password

☐ I agree with terms and conditions..

Create account

Log in Page

Log in

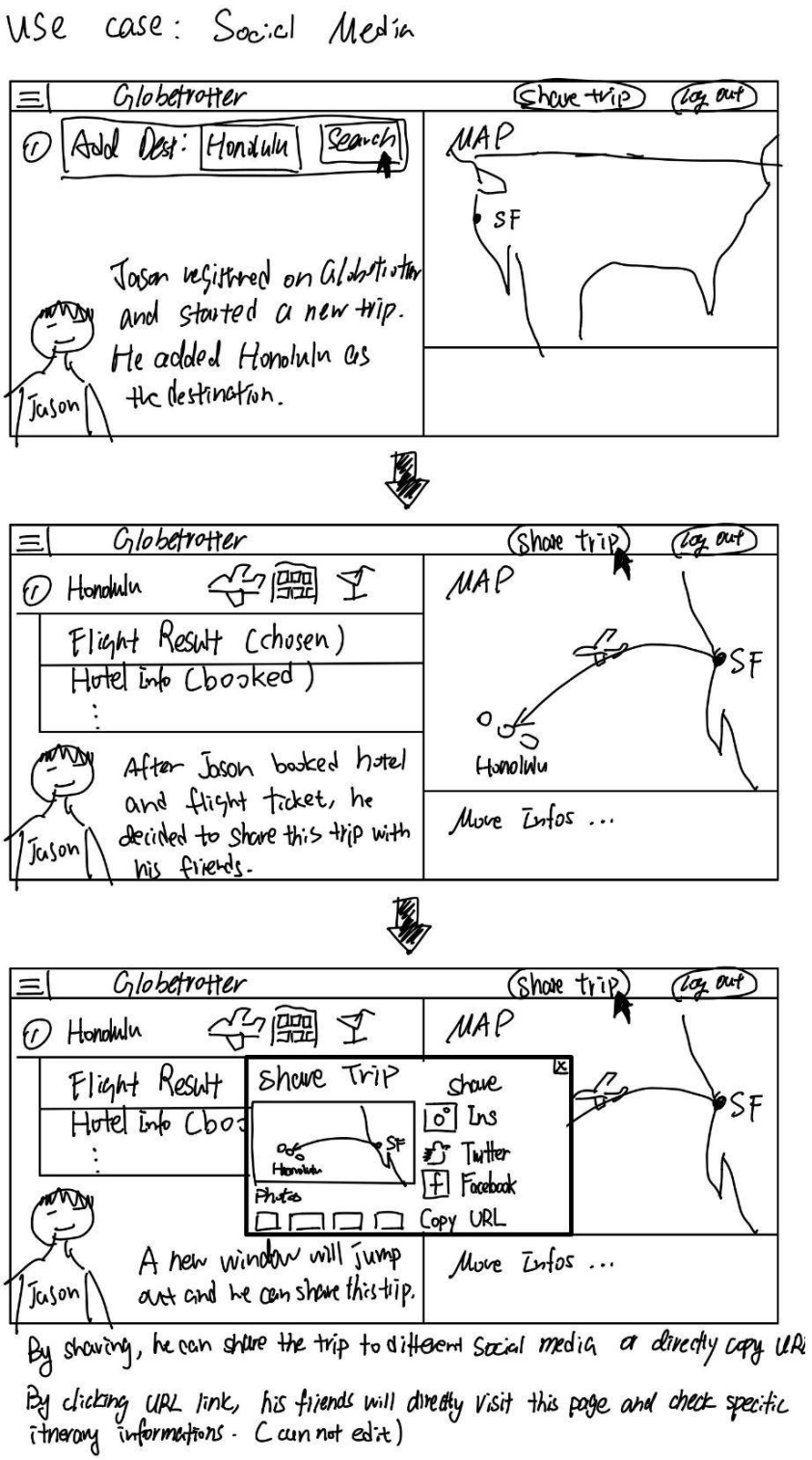
Account num :
Password :

Log in

forgot password ?

Don't have an account? create

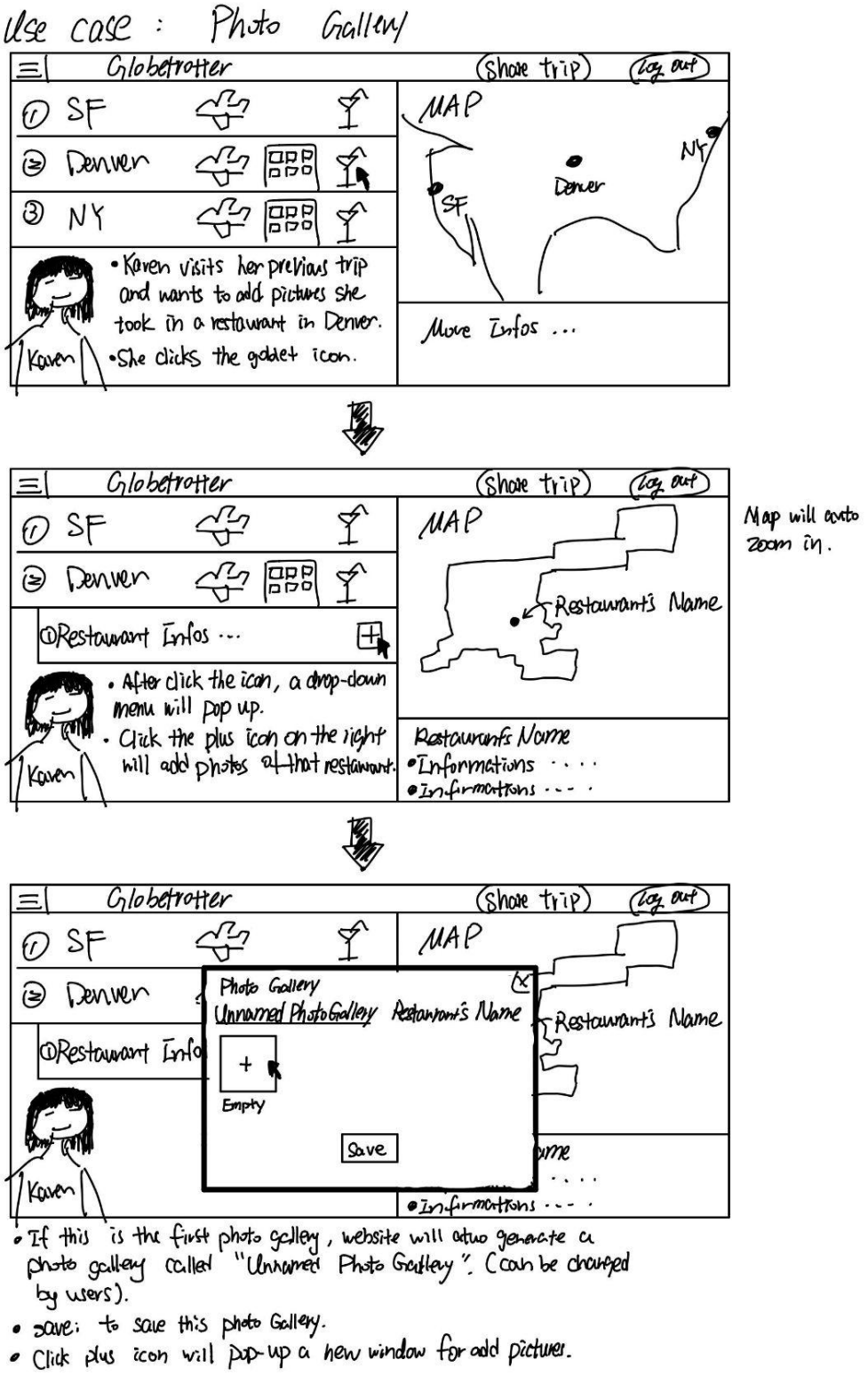
Social Media

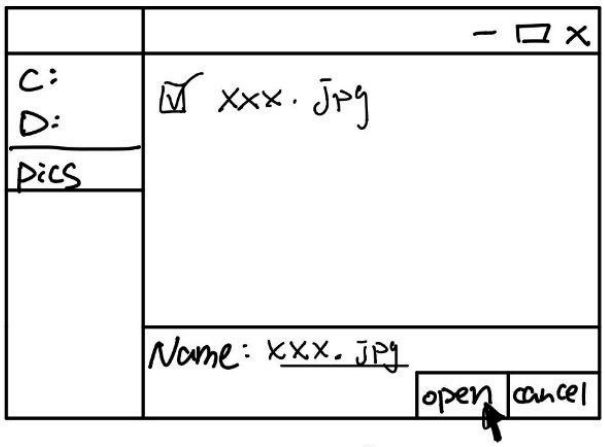


By sharing, he can share the trip to different social media or directly copy URL

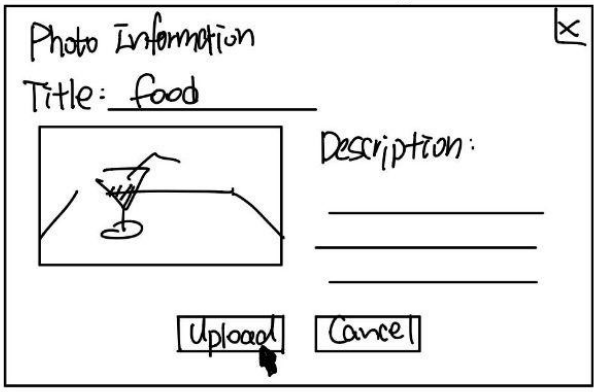
By clicking URL link, his friends will directly visit this page and check specific itinerary informations. (Can not edit)

Photo Gallery

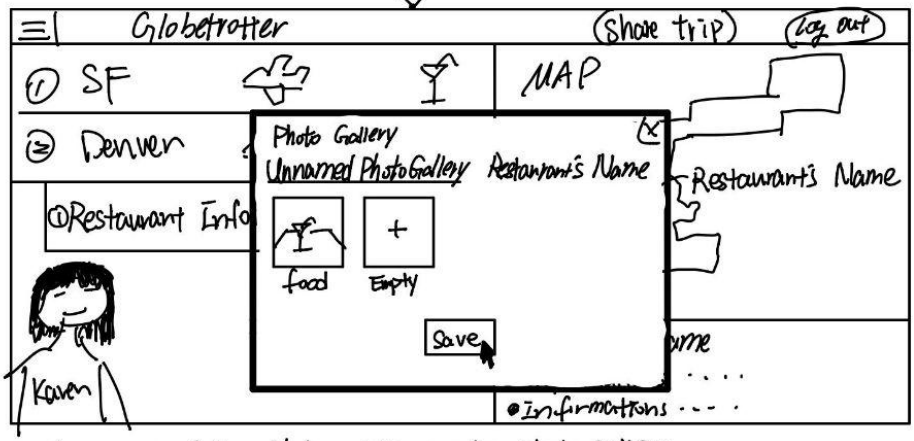




- Then Karen can browse her PC folders and choose picture to upload.



- If file opened successfully, this window will show up on her website.
- Click upload will upload this picture and information to the photo gallery.



- Karen successfully added a picture in her photo gallery.
- Click save icon will save the whole photo gallery

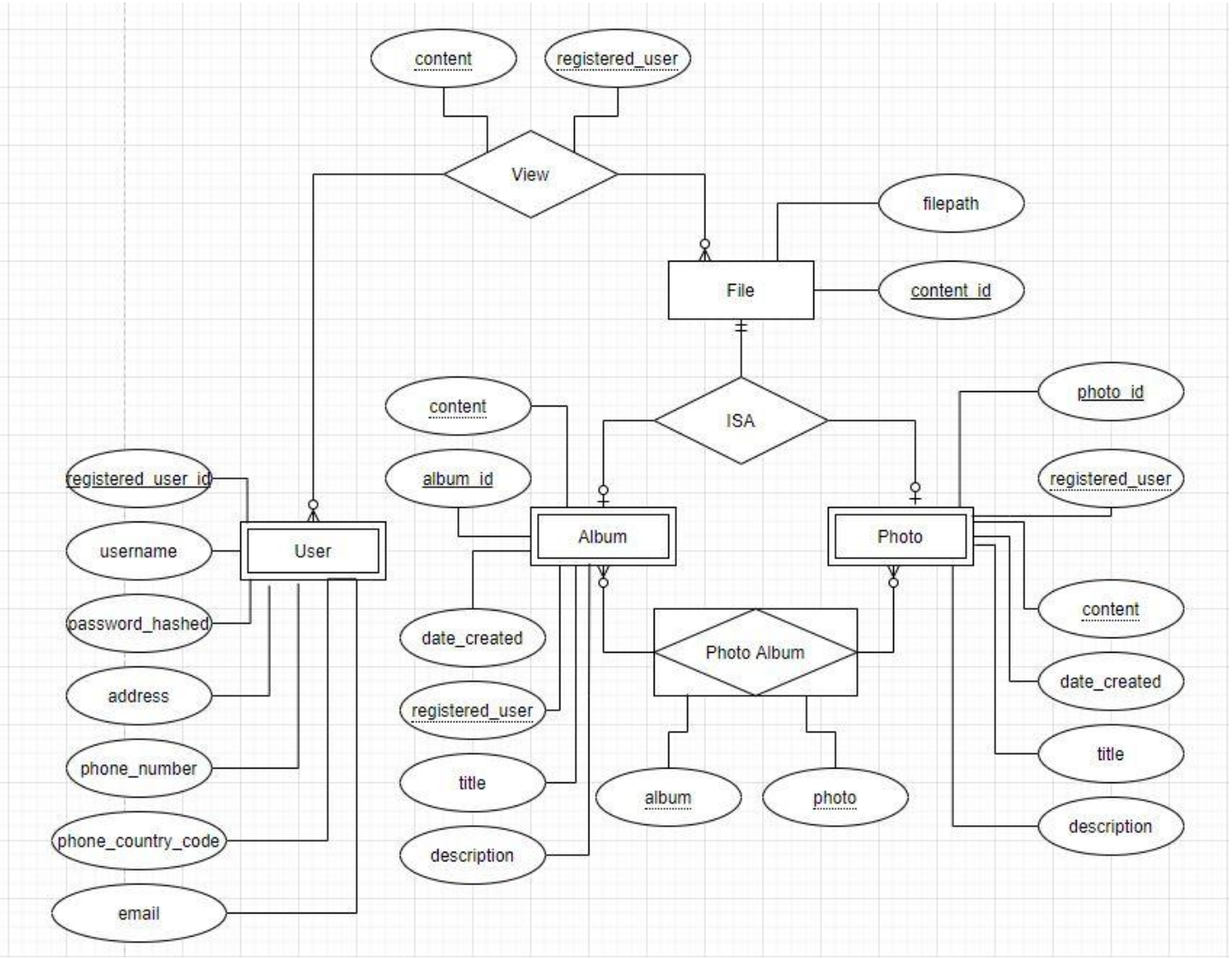
High Level Database Architecture & Organization

Business Rules for Photo Albums

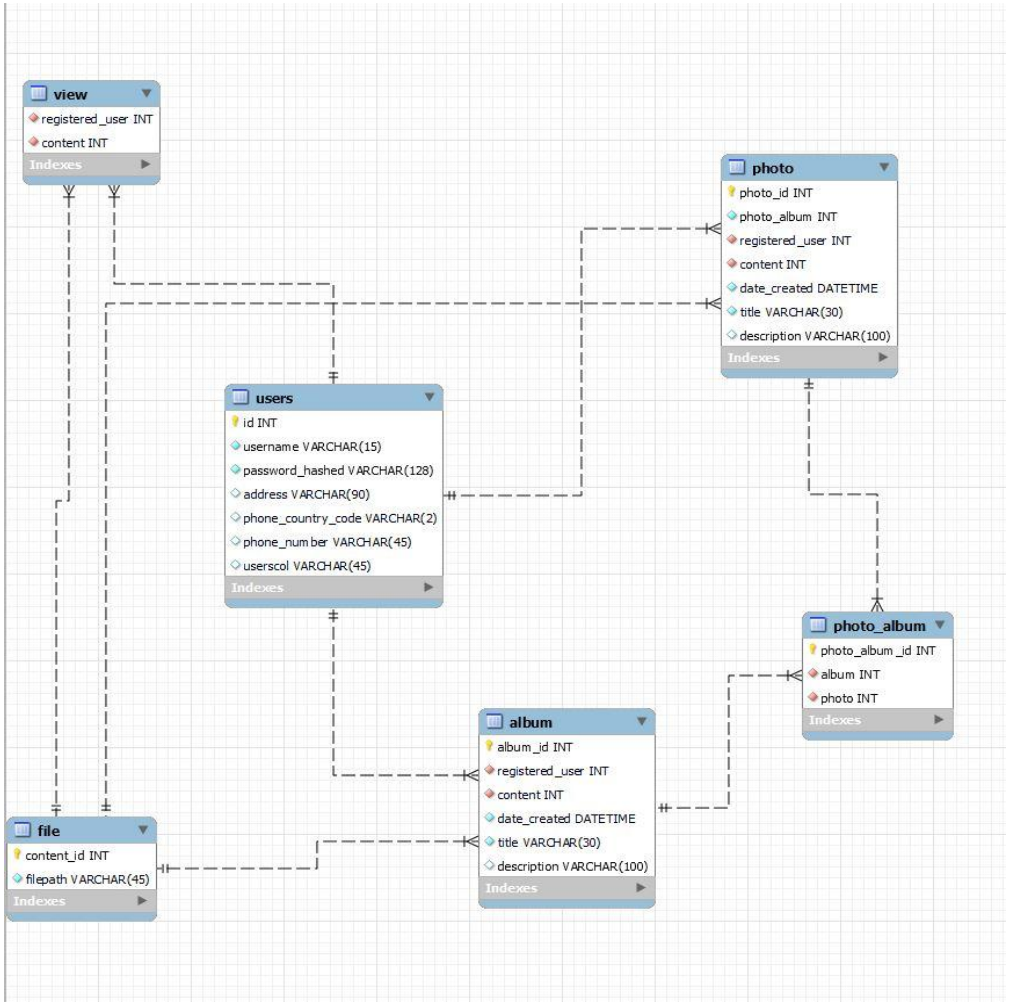
Entities, Attribute, Relation, Quantitative

1. User
    a. A User shall be able to view many user_content
2. File
    a. A File shall either be a Photo or an Album.
    b. A File shall be viewed by many Registered Users
3. Album
    a. An Album shall Photo_Album many Photos
4. Photo Album
    a. A Photo Album shall have many photos.
    b. A Photo Album shall be viewed by many users.
5. Photo
    a. A Photo shall be Photo_Albumed by many albums.
    b. A Photo shall have one name.
    c. A Photo shall have one description.

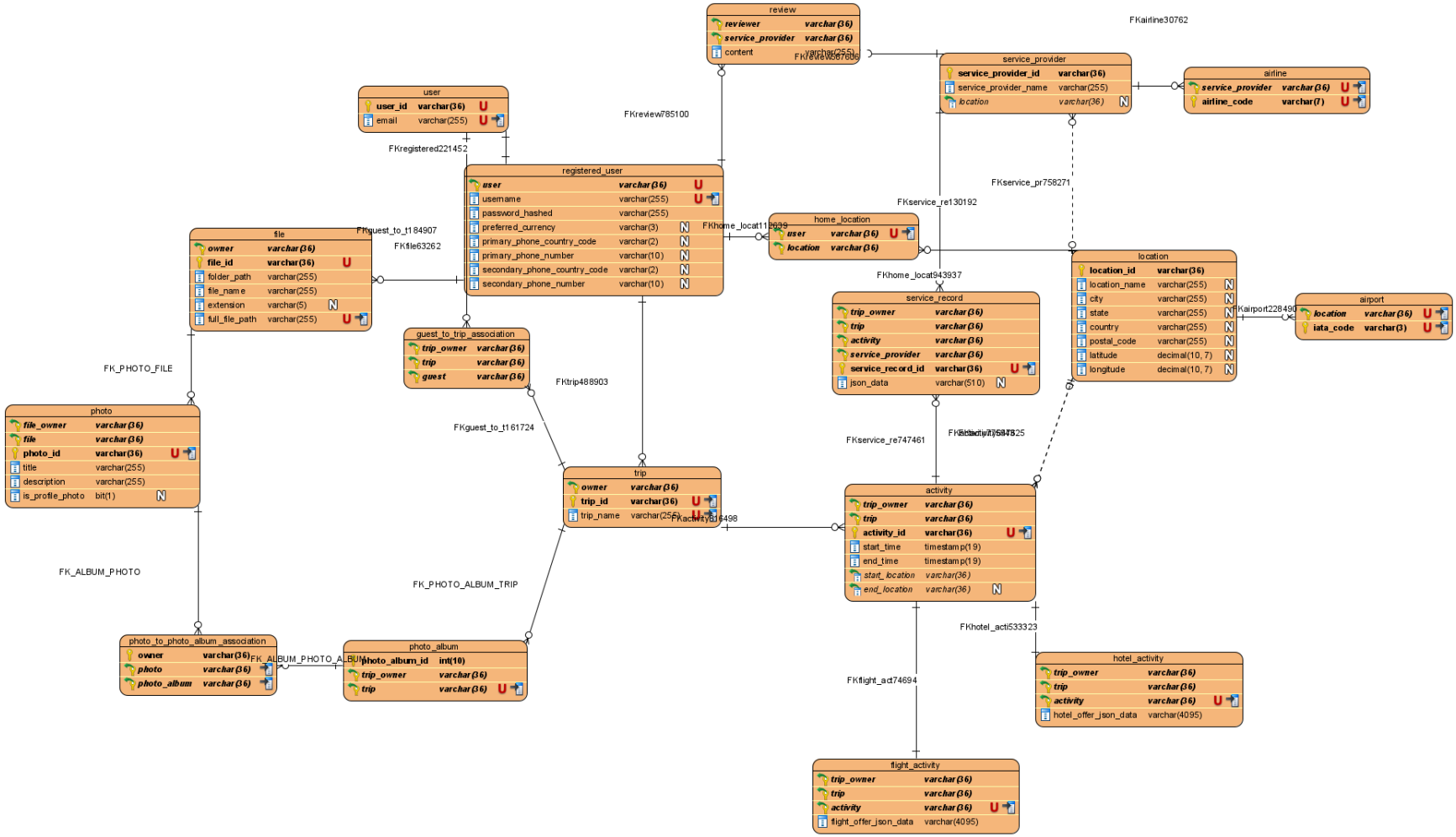
Entity Relationship Diagram (Photo Albums)



Database Model (Photo Albums)



Database Model (Full)



Search Implementation

- We will be using MySQL as our database management system because some of our team members already have some experience with MySQL.
- Photos will be kept in a file system and not stored as blobs.
- The user will be able to search for users by Username or Email.
    - Users will have a dropdown choice os Username and Email to choose from and confirm their choice witht the “next” button.
    - After they have confirmed what they are going to be inputting, the user will input their text in the textbox and press submit to perform the search.
    - SELECT * FROM GlobetrotterV1.users WHERE username LIKE '%${searchString}% will be used to search for users by username where “searchString” is the input of the user.
    - `SELECT * FROM GlobetrotterV1.users WHERE email LIKE '%${searchString}%'` will be used to search for users by email where “searchString” is the input of the user.

High Level APIs & Main Algorithms

Our API's

Auth

- Registration:
    - POST Request**:** The unregistered user will be asked to enter a user name and password for their account. The password will be hashed and the username, along with the hashed password, will be stored in the database.
- Login:
    - POST Request**:** All users will be required to enter a username and a password. If the username exists in the database and the hash of the supplied password matches the 'password_hashed' field of that user, the user will be given a token that will serve as proof that they have been authenticated. A user session will be created for the user in the database.

Users

- Search Users
    - GET Request: The Registered User will supply either a username or email and will be returned a list of Registered Users with matching values.
- Get Me
    - GET Request: The Registered User will supply their access token which the server will use to identify them. After the user has been identified, they will be returned their user object.

Trips

- My Trips
    - GET Request: The Registered User will supply their Access Token which the server will use to identify them. After the user has been identified, all Trips belonging to the User will be returned.
- Get Guests
    - GET Request: The Registered User will supply their Access Token as well as a Trip ID. After the Trip is found in the database, all User objects for Guests of that Trip will be returned.
- Add Guest
    - POST Request: The Registered User will supply their Access Token, a Trip ID, and a User ID. This will add the User with the supplied User ID as a Guest to the Trip with the provided Trip ID.

Reviews

- My Reviews
    - GET Request: The Registered User will supply their Access Token, and be returned a list of Reviews which they have made.
- Create Review
    - POST Request: The Registered User will supply their Access Token, a ServiceProvider ID, and a string. This will add a new Review object in the database which will be associated with the Service Provider.
- Delete Review:
    - DELETE Request: The Registered user will be able to delete a review. They will first have to get the request that they wish to delete. The database will return the review that they searched for and then they will have the option to delete their review.

Photo Albums

- Create Photo Album for Trip
- Get Photo Album for Trip
- Create Photo in Photo Album
- Get Photo from Album
- Delete Photo from Album

Flights

- Search
    - GET Request**:** All users will be able to search for flights. The users will be asked to enter some information, such as: Origin Location Code, Destination Location Code, departure date, number of passengers, currency code, and the number of results desired. After the information is acquired, it will be sent to the Amadeus “Flight Offer Search” API which will return a list of flights based on the criteria given by the user.

Third Party API’s

The APIs that follow are used to obtain information about hotel rooms and flights, as well as to perform booking.

Flights

We decided to go with the Amadeus Air APIs because these APIs returned a lot of information that would be good for the user to know. It also took care of certain things that we would not have to implement on our own. For example, if a return date is specified, the API would return round trips instead of us having to filter out the trips we want to display. The results are already filtered when being returned by the API.

Amadeus AIR APIs

- Flight Offer Search
    - GET Request**:** Information entered by the User, when they do a GET request, will be sent to the Amadeus Flight Offer API. This API will return a list of flights based on the criteria given by the user.
- Flight Offers Price
    - GET Request**:** Once the user searches for flights using the Flight Offer Search endpoint, a GET request will be made to the Flight Offers Price endpoint automatically. This endpoint will return the price for any flights that were returned by the Flight Offers Search endpoint. Once the prices have been retrieved, the flights, along with their prices may be displayed.

Hotels

We chose the Amadeus Hotel APIs because they are the most complete hotel searching APIs that we found. Using these APIs we are able to get detailed information about room availability, ratings, and prices, for a variety of hotels.

Amadeus Hotel APIs

- Hotel Search
    - GET Request: When calling this API the following parameters are supplied: either a city code or GPS coordinate, a search radius, check-in and check-out dates, room quantity, a currency code, as well as the number of adults. After the request is made, the API returns a list of **Hotel Offers**, each containing a list of room offers from a particular hotel, as well as information about the hotel.
- Hotel Booking
    - POST Request: When calling this API, the request body should contain the **Hotel Offer** ID, a list of guests, and payment information. The API will return an array for each successful booking, each containing a 'providerConfirmationId' and an array of 'associatedRecords'.

Amadeus SDK

To simplify the process of using the Amadeus APIs, we intend to use the Amadeus Node SDK. This SDK is updated by Amadeus and hosted on Github at https://github.com/amadeus4dev/amadeus-node

Main Algorithms

- The app will mainly use sorting and filtering algorithms to present the data. These will appear when:
    - Viewing flights in order of price
    - Filtering flight ticket results by date and time

-When the user starts a search, it will either be a valid or invalid search.

-If User searches for a non-existent Flight, Restaurant, Activity, Trip, then the backend will send a message saying that the search is not valid to the fronted UI.

-If the User searches for a Flight from a valid airport and to a valid airport(destination), then the backend will send a list of airlines displaying the following departures of flights.

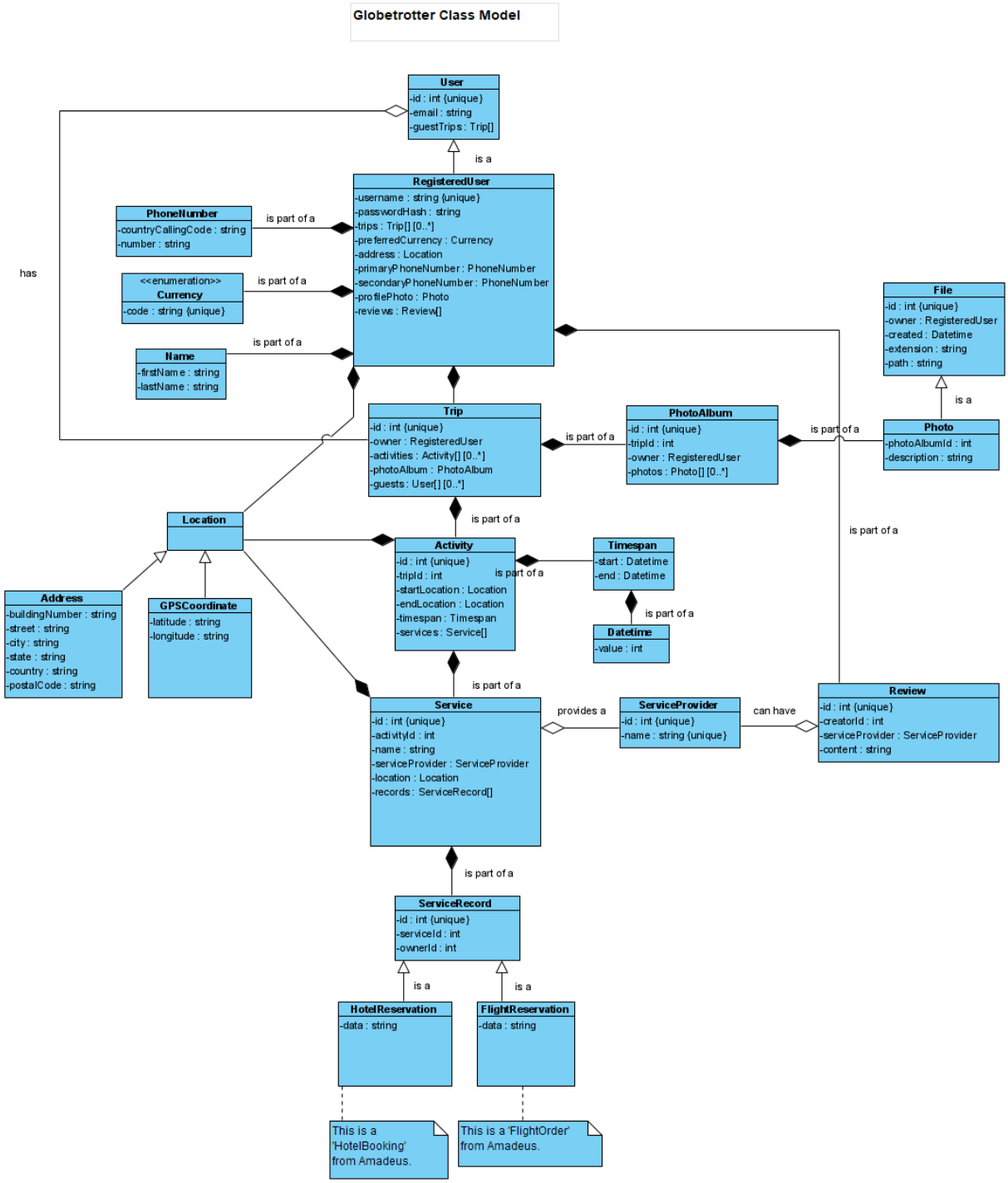
-If the User searches for a Trip that was created previously, the backend will show a Trip that matches the searched input given by the User.

-If User searches for activities around a certain region, the backend will display a list of things available for the User to view on the frontend UI.

-If the User searches for a service provider such as a restaurant, the backend will send a list of restaurants available in the region of interest to the fronted UI for the User to view.

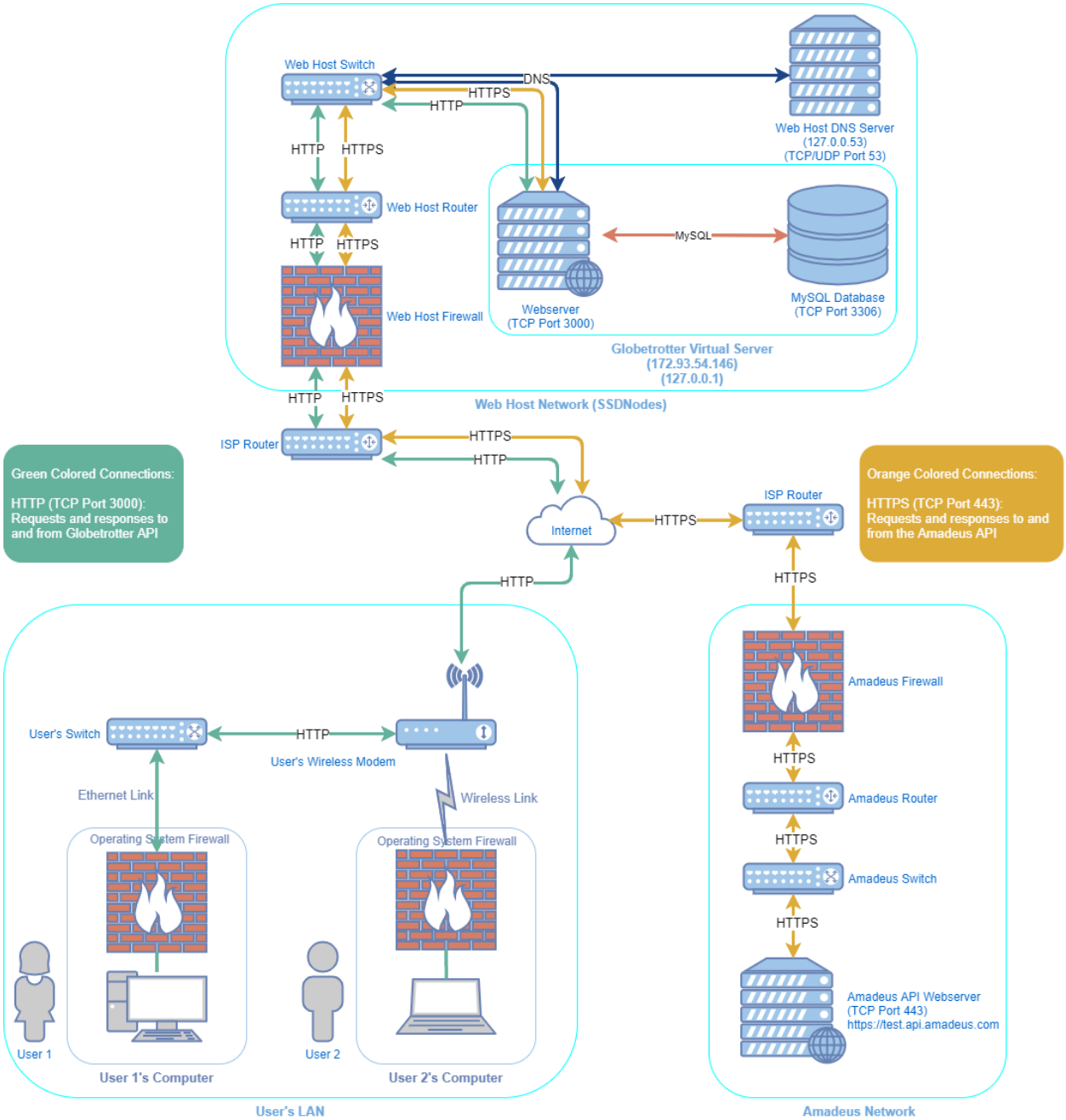
High Level UML Diagrams

Data Model

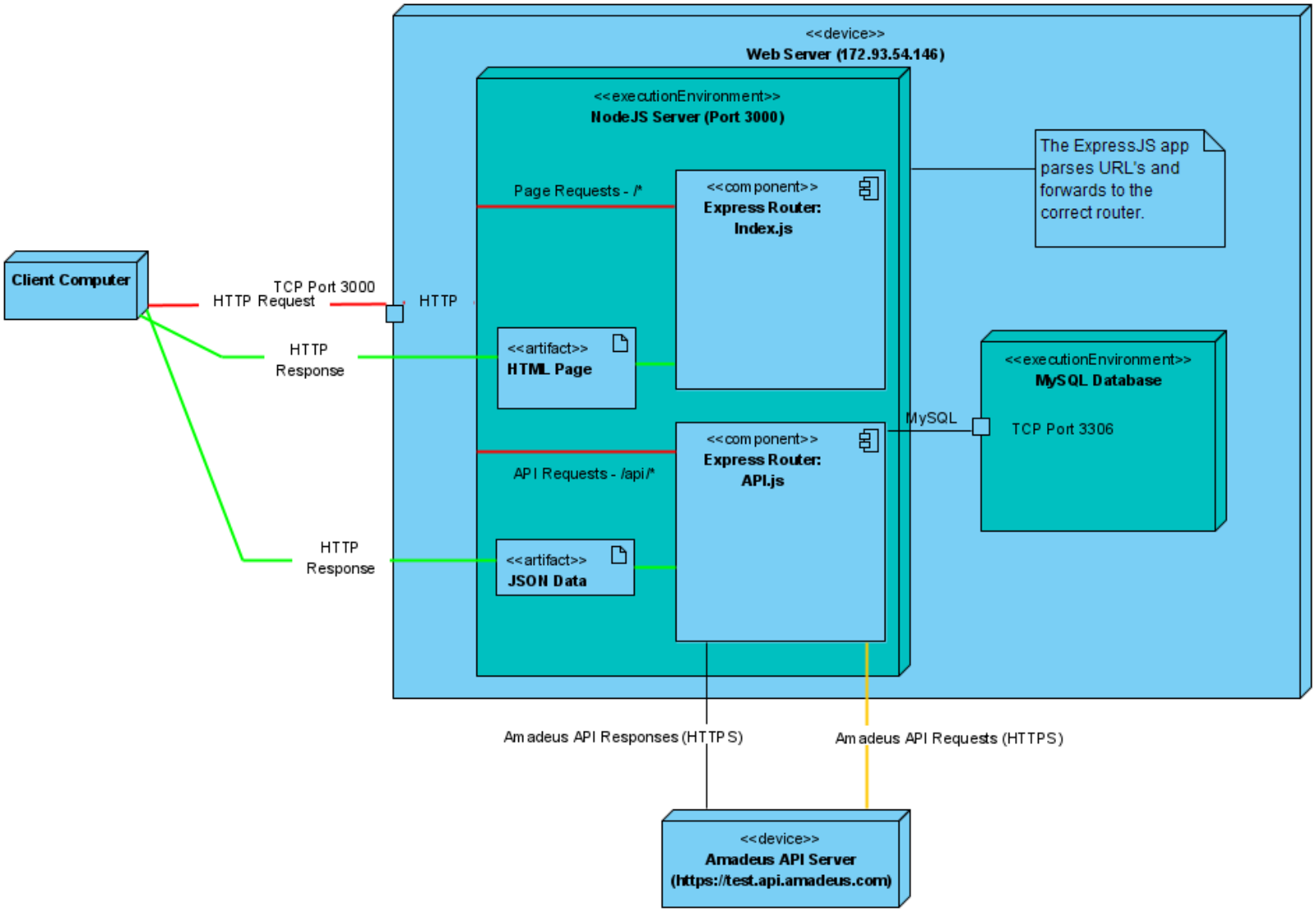


High Level Application Network & Deployment Diagrams

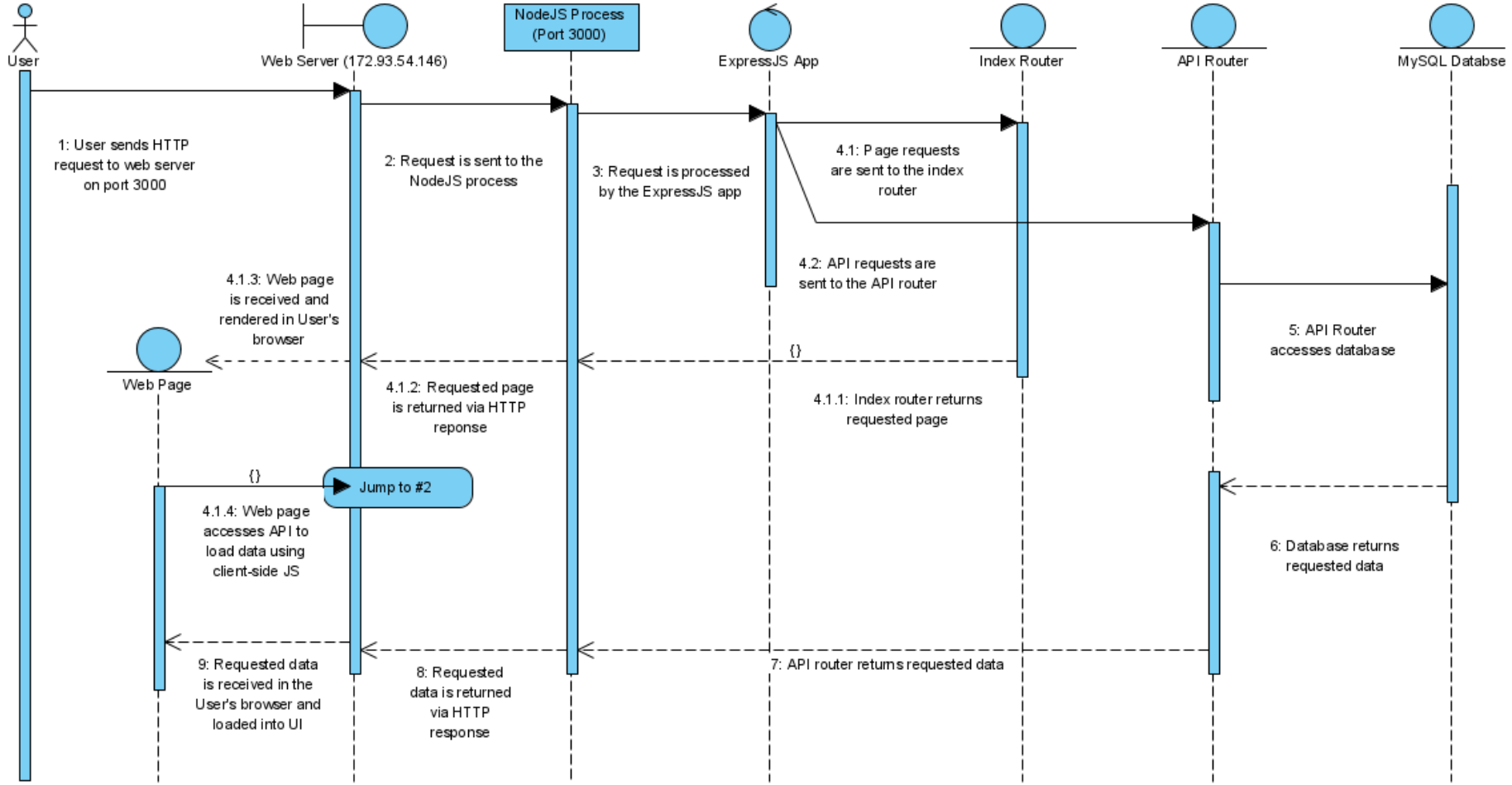
Network Diagram



Deployment Diagram



Request Processing Sequence



Key Project Risks

Skill Risks

- Not everyone is familiar with the software stack.
    - Solutions:
        - Provide an overview of the software stack.
        - Direct team members to online tutorials.
        - Recommend books that focus around the software stack.

Schedule Risks

- Implementing a checkout feature may take more time than the other parts of the system.
    - Solutions:
        - Focus more on other features while still trying to figure out a way to implement a checkout feature. Making one feature, such as flight planning and booking, very strong will make it easier to add other features in the future.
- Integrating hotels as a usable Service may take more time than we anticipated
    - Solutions:
        - Same as the checkout feature. If we focus on making Flights work well as a Service available to users, it may be easier to integrate hotel reservations in the future

Teamwork Risk

- Uneven distribution of front/back end team members
    - Solutions:
        - Attempt to distribute tasks as evenly as possible.
        - Offer assistance on tasks that team members might be behind on.
        - Reach out for help if tasks assigned are running behind schedule.

Technical Risks

- Need to develop access control for the API endpoints. Some of the endpoints should only be accessed by an Admin level user.
    - Solutions:
        - We can implement a Role column on each user in the database with an option of either Admin or User. The Role of a user can be checked in each API call.
- Cannot yet deserialize JSON objects returned from the database into typed-objects with methods.

    - Solutions:
        - Research JavaScript classes and prototypes. Creating prototypes or classes from the JSON data will allow us to have better control over the data.
- Overall, keeping track of all the data and parameters that go into and come out of the external API's is tricky.
    - Solution:
        - Same as above. We need to figure out how to create our UML data model in JavaScript using classes or prototypes. We will focus on making our data model more robust and learn how to model it in code.
- We are in need of a mapping and geocoding API for the front end.
    - Solution:
        - Team lead will assign members of the front end to experiment with mapping API's.

Legal Risks

- None at this time.

Project Management Strategy

Present

- We had more frequent meetings.
- We had separate and more detailed meetings for the front and back end teams.
- We began using Trello to divide and assign tasks to team members.
- Deadlines were set on tasks within Trello.
- We reviewed individual tasks and provided constructive feedback as a team.

Future

- Team lead will assign tasks with clearer objectives and more specific due dates.

Team Contributions

Jesus

Created storyboards for ‘Planning a Vacation Schedule’ and ‘Inviting Guests on a Trip’, Created front end of Vertical Prototype, Contributed to Data Definitions

Ruja

Developed interactive UI Prototype, Created storyboard for ‘Activity Itinerary’

Yi

Created storyboards for ‘Login/Sign-Up Process’, ‘Social Media’, and ‘Photo Gallery’

Robin

Created storyboard for ‘Business Trip’

Luis

Created the business rules for Photo Gallery, ERD diagram, Coded parts of the back end for Vertical Prototype, Contributed to Data Definitions

KJ

Added to API Endpoints table and Algorithms section

Taylor

API Endpoints table, Network and Deployment Diagrams, UML Diagram, contributed to Data Definitions