Vacation Board App

Team Name: Team Printf(“Team\n”);

Working Application Title: FRí

Full Project Documentation

Spring 2022

James Tipton, Ankit Jain, Kent Studer, Kobby Mensah

Table of Contents

4. **Best Method Summary**

Design and UI/UX

Design subsections:

Development and Testing

Introduction and Foundations

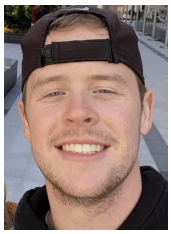
Main goal:

The main goal of the group is to streamline group/family vacation planning. One of the biggest downsides to planning a trip with multiple parties is communication on who wants to go where and for what price. In order to maximize trip planning efficiency the team must create an application that aggregates live data from various websites relevant to planning a trip. These include airlines, hotels, rentals, and general tourism information that are crucial in making a decision. By doing so a mechanism can be created by which everyone involved in the trip can view and vote on certain key aspects of the trip (destination, flights, and hotels).

Team information:

1. **Mission Statement:** Team Printf("Team\n")'s mission statement is to make planning a trip as relaxing as the trip itself.
2. **Team Members:**

a. Kent Studer: Backend Developer



i. Kent Studer is a senior computer science student here at Mizzou who is a research fellow for the Bioinformatics department. With research in developing python servers for web services, Kent will handle a lot of the backend components of this application.

b. James Tipton: Full Stack Developer

i. James Tipton is a senior computer science and psychology student at Mizzou, who is also a research fellow for the Bioinformatics department. With research and experience working with front-end web development and backend APIs, James will handle a lot of front end development for this application, and back end API logic.

c. Ankit Jain: Backend Developer/Database Developer

i. Ankit Jain is a senior computer science student at Mizzou who currently works at Shelter Insurance as a part-time software developer. With experience in Python Flask backend development , MySQL, and Azure Cosmos DB, Ankit will handle backend development as well as database design.

d. Kwabena (Kobby) Mensah: Frontend Developer/Database Developer

i. Kobby Mensah is a senior computer science major at the University of Missouri, who is also currently a member of the US army and works part time at the university hospital. Kobby over the years has acquired experience in frontend and backend technologies such as web development, API's, MySQL, JavaScript and MongoDB. He will be assisting with frontend development application.

3. **Mentor:** Jim "JimR" Ries is a faculty member here at Mizzou's Computer Science department. JimR teaches the first coding class most students take, CS 1050 as well as

the more advanced operating systems course (CS 4520). The reason the team members decided to go with JimR as mentor was because not only did he inspire and motivate every member to continue down the path of software development, but all members of this team are also in his operating systems course this semester. Having such a profound impact on the team in our younger years coupled with his considerable knowledge of C and love for all things code, made this an easy choice for this group. JimR also has extensive experience in the industry with team-oriented software development, and will be able to help us with valuable advice.

Introduction:

- **Team name: Team Printf**

Team Printf represents a passion for Computer Science in the very line of code that started it all, printf. With a diverse background of technical abilities Team Printf aims to combine incorporate backend development with dynamic and user friendly front-end development. Each member possesses a common goal of making an otherwise arduous task such as vacation planning, fun and cooperative for all members involved. In doing so Team Printf hopes to reshape trip planning to be just as relaxing or fun as the trip itself. Our application name is Frí (pronounced “free”) which means “vacation” in Icelandic. We chose to go with this name because it is simple, catchy and captures the background of our application.

- **Problem definition (statement):**

Trip planning amongst a group presents itself as a mess of differing opinions, and ideas. Deciding budgets, locations, activities, accomodations, etcetera for a vacation with multiple

people takes a lot of time and energy and is usually done through apps unrelated to travel such as Snapchat or GroupMe. Without a centralized store of relevant travel data, most of these vacation ideas either burn out or leave people unincluded in the plans.

- **Problem resolution**:

The plan to solve this issue is letting users vote for the features of a group travel trip on a website platform. This will work by having a host user create an initial trip by entering in the total number of people going on the trip and the trip dates. The host user can then invite people to the created trip via email or social media, where each invited user can create polls that other users can vote on to decide preferences. These preferences include individual budgets for flights, lodging, transportation, food, and attractions. More logistics can be voted on such as departure location. The application will then use an algorithm to choose from preexisting data the most optimal flights, lodging, etc based on users' votes.

- **Benefits of the solution**:

The solution will help those who want to go on a group trip eliminate the issues of conflicts when deciding on features of the trip. It will become easy to add people to a trip and have them drop if they are no longer interested. It will also phase out the timely process of having to manually look up things to do for the group during the trip, the voting functionalities will find and decide the best fits for everyone.

Requirements:

Hardware requirements:

- Computers / mobile phones with internet for testing and developing
    - Web application needs to work on Google Chrome/chromium-based browsers. Google Chrome minimum requirements:

Windows

Important: Chrome is extending support for Windows 7 through January 15, 2023 for critical security and stability updates.

To use Chrome browser on Windows, you'll need:

- Windows 7, Windows 8, Windows 8.1, Windows 10 or later
- An Intel Pentium 4 processor or later that's SSE3 capable

Note: Servers require Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2, or Windows Server 2016.

Mac

To use Chrome browser on Mac, you'll need:

- OS X El Capitan 10.11 or later

Linux

To use Chrome browser on Linux, you'll need:

- 64-bit Ubuntu 18.04+, Debian 10+, openSUSE 15.2+, or Fedora Linux 32+
- An Intel Pentium 4 processor or later that's SSE3 capable

Android

To use Chrome browser on Android, you'll need:

- Android Marshmallow 6.0 or later

- Cloud platform (Google Cloud) to host website

User/non-user or functional/non-functional requirements:

Notes:

- Red denotes a backend-only variable, other associated data is passed by the user.
- Admin is a subclass of member.

User requirements bulleted list:

1. Register and verify a user based on user input data, generates user ID.
    a. User: Member
    b. Activity: Register
    c. Associated Data:
        i. Username
        ii. UserID
        iii. Password
        iv. Email
        v. Phone Number
2. Create a travel group tied to the leader (admin)'s id generated by the app. Can call add member function upon initialization. Private means admin needs to approve of adding members, public means any member can add others.
    a. User: Admin (subclass of member)
    b. Activity: Create group
    c. Associated Data:

i. Group name

ii. UserID

iii. Group photo(optional)

iv. Private/Public status

3. Will send invites out to the associated username or email regardless of group settings.
    a. User: Admin
    b. Activity: add member(s)
    c. Associated data:
        i. Username/ email
        ii. UserID
4. Send out invites via email (people who don't have the app) or directly through the app. Only allowed if the group is public. Join the group or choose tentative status, generate an user ID. Otherwise decline.
    a. User: Member
    b. Activity: add member(s)
    c. Associated data:
        i. Username/ email
        ii. Private/Public status
        iii. UserID
5. Join the group or choose tentative status, generate an user ID. Otherwise decline.
    a. User: non-Member
    b. Activity: Join group
    c. Associated data:

        i. GroupID

6. Enter preferences such as budget and location preferences.
    a. User: non-Member
    b. Activity: Enter individual preferences
    c. Associated data:
        i. GroupID
7. Delete the group, in turn removing everyone from it.
    a. User: Admin
    b. Activity: Delete group
    c. Associated data:
        i. GroupID
8. A user can remove themselves from the group
    a. User: Member
    b. Activity: Leave group
    c. Associated data:
        i. GroupID
        ii. UserID
9. Any member can create a poll to be voted on. This includes the poll settings, what the poll is about, and the voting options.
    a. User: Member
    b. Activity: Create Poll
    c. Associated data:
        i. Start Time

ii. End Time

iii. Options[]

iv. Information[]

v. PollID

vi. UserID

10. The member who created the poll can edit and change any of the poll settings and options.
    a. User: Member
    b. Activity: Edit poll
    c. Associated Data:
        i. PollID
        ii. UserID
11. The member who created the poll can delete the poll.
    a. User: Member
    b. Activity: Delete poll
    c. Associated Data:
        i. PollID
        ii. UserID
12. Vote on preference of options (flight, location, restaurant, dates, accommodations, etc)
    a. User: Member
    b. Activity: Vote
    c. Associated Data:
        i. UserID

ii. Options[]

13. A user can change their vote on any given poll.
    a. User: Member
    b. Activity: Change vote
    c. Associated Data:
        i. UserID
        ii. PollID
14. Alter group settings (name, photo, private/public status etc)
    a. User: Admin
    b. Activity: Change group settings
    c. Associated Data:
        i. GroupID
        ii. Group name
        iii. Group photo
        iv. private/public status
15. Change status to tentative (unsure if joining), accept (joining), or decline (not joining). Gives member privileges upon joining, gives viewing privileges upon tentative.
    a. User: Non-member
    b. Activity: Change invite status
    c. Associated Data:
        i. UserID
16. Change status of whether the group can see your location or not.
    a. User: Member

   b. Activity: Change tracking status
   c. Associated Data:
      i. User ID
      ii. Tracking status

17. View map to see locations of other members.
   a. User: Member
   b. Activity: View other user's locations
   c. Associated Data:
      i. User ID
      ii. Tracking status
      iii. Group map

18. Search any location in the world, the function returns a list of places to stay at the given location. The StayResults list contains a list of locationIDs that make places uniquely identifiable
   a. User: Member
   b. Activity: Search location
   c. Associated Data:
      i. Search entry
      ii. StayResults[]

19. User can favorite a location they like and store that information in a favorites list
   a. User: Member
   b. Activity: Favorite place
   c. Associated Data:

i. LocationID

ii. UserID

20. Removes a locationID from the given UserIDs favorite list

a. User: Member

b. Activity: Remove favorite

c. Associated Data:

i. UserID

ii. LocationID

Project Related Research

Project Related Research: Team Printf

Introduction: Team Printf's goal is to effectively accomplish the features the app will require to centralize the planning and decision making of a group trip, including the items needed prior to the trip (flights, hotels) and during the trip (attractions, ground transportation). Implementation methods will comprise of API's, languages, algorithms, and frameworks that are found necessary to accomplish our application's features. The team is to build a web application and/or mobile application with a user interface that utilizes APIs to input data needed to form a group trip (e.g. flight prices, hotel locations) and algorithms that will match user preferences with certain items to create end decisions.

Problem Statement: To find the implementation methods that will effectively accomplish the features of the application that the team wishes to create. The team will consolidate multiple

choices and compare the values and disadvantages of each, and choose the best option or options for the project moving forward:

Application of Research:

Legend:				
Framework	Server	Algorithm	Data consolidation API	Map/Location API
Language	Flight API	Veto	Maybe	Chosen

- **Algorithms necessary for user-based features**
  - To implement the user voting and decision-making functions the team will use the stable marriage algorithm[3], with the boyer-moore majority voting algorithm or a sequence alignment algorithm as possible backups[4].
  - The team considered stable marriage, Boyer-Moore, and Sequence alignment. The stable marriage was chosen because it will work well in choosing a single choice for the group as a whole based on individual preferences. We can use a modified version of this to handle polls and preferences where the matches are not binary.
  - The other two algorithms may be used for minor implementations, for instance sequence alignment can be used to align preferences among subgroups within the larger travel group.

Status	Implementation Method	Pros	Cons	Source
	Stable Marriage Algorithm	-Relatively fair and optimized voting algorithm -Easy	-Stable Marriage takes O(n^2) time.	https://en.wikipedia.org/wiki/Stable_marriage_problem#:~:text=The%20stable%20marriage%20problem%20has,other%20than%20their%20current%20partners.
	Boyer-Moore Majority Voting Algorithm	-Simple algorithm that uses majority voting on an item to make decisions. -Fast. Time complexity of O(n).	-Should not be used in a small group of people because not enough variety to cast a justifiable vote.	https://www.geeksforgeeks.org/boyer-moore-majority-voting-algorithm/
	Sequence Alignment Algorithm	Another possible option for matching algorithms. (For voting and preferences)	More complex than stable marriage	*Algorithm Design* by Jon Kleinberg and Éva Tardos

- **API's to be used for backend features and data consolidation (maps, geodata, flight information, hotel/residence accommodation, etc.)**

- The application will utilize Amadeus for the flight, hotel, and point of interest APIs. For maps and location services, the group will use Amazon Location Services[2].
- The group decided on Amadeus for most of our API needs, as it is free, and includes hotel and point of interest API's.
- Google flights API is a backup option but it does have a cost involved.
- Amazon Location services was chosen because of free geocoding and location data capabilities.
- Google and Mapbox are backup options if issues arise.

Status	Implementation Method	Pros	Cons	Source
	Google Flights API	Its Google. Very dependable and reliable API	Cost per month involved (traffic based)	https://rapidapi.com/blog/google-flights-api-incorporate-travel-data-into-your-app/
	Skyscanner API		Does not give access to students	https://www.partners.skyscanner.net/affiliates/travel-apis
	Amadeus Flight API	Free tier (with request quota cap), REST API compatible with python	Transaction fee for production environment	https://developers.amadeus.com/pricing

Status	Implementation Method	Pros	Cons	Source
	Amazon Location Services	Provides maps, geocoding, and points of interest. A free tier exists	Not as well known as Google in terms of location data	https://aws.amazon.com/pm/location/?trk=eddb8f59-63e8-4fbc-a519-e4fd1c6e8466&sc_channel=ps&sc_campaign=acquisition&sc_medium=PAC-PaaS-P|PS-GO|Non-Brand|Desktop|SU|Location%20Services|Solution|US|EN|Text|PMO22-13403&s_kwcid=AL!4422!3!542733214710!p!!g!!maps%20api&ef_id=Cj0KCQjwxtSSBhDYARIsAEn0thT5KxoZLTwOCyFV7j68Kf0I6VWhgOE-pDqY-QNrguZM8SJECkWmQcsaAtpmEALw_wcB:G:s&s_kwcid=AL!4422!3!542733214710!p!!g!!maps%20api#Learn_More_About_Amazon_Location_Service
	Google Maps API	Well known API with all of the location data we would need for the app. Extensive documentation for the API. Javascript API available	Cost per month involved (traffic based), deprecated as of 2018	https://developers.google.com/maps
	Mapbox API	Free tier available. Provides maps, navigation SDK, geocoding, etc	Not as well known as Google and Amazon (documentation may be more sparse), lot of features we don't need	https://docs.mapbox.com/api/overview/
	Amadeus Hotel API	Free tier (with request quota cap) REST API compatible with python	Transaction fee for production environment	https://developers.amadeus.com/pricing
	Amadeus Point of Interest API	Free tier (with request quota cap) REST API compatible with python	Transaction fee for production environment, does not look very extensive	https://developers.amadeus.com/pricing
	Airbnb API	Airbnb is more commonly used nowadays and more accessible to interesting and/or less populated locations	Not accepting API access requests currently	https://www.airbnb.com/partner

- **Programming languages that will be best suited for this type of application**
    - The method that would allow for the team to reach their goal most effectively would be creating a web application using Flask written in Python[1].
    - The backend of the application will be made using mySQL to handle the persistent data[5].
    - The team decided on Python Flask over Angular/Typescript due to the lesser learning curve involved for the members during development. This sacrifices some versatility between platforms for the final product, so Typescript remains as a backup.
    - MySQL was chosen for database use and persistent storage due to ease of use and experience among the group.

Status	Implementation Method	Pros	Cons	Source
	Python	-Every team member is well versed and knowledgeable in python -Would take get working immediately -No research overhead	-Finished app will not be easily transferrable to mobile	python.org
	Typescript	-Finished app will be easily transferrable to mobile -JavaScript backwards compatible	-Not all team members are well versed in typescript/javascript -More research will be required during development	typescriptlang.org
	mySQL	Easily interfaceable with Python or Angular, necessary for backend and persistent data	i cant think of any sql is easy and fast	
	MongoDB	kent is an expert, security, necessary for backend and persistent data, semi-easy interface with Python	Cost involved without a free trial or student license which expires	

- **Frameworks that will be most useful to facilitate web development and features**
    - As stated in the above section, Flask was chosen due to learning curve issues, while Angular Ionic remains as a backup in case of any cataclysmic issues with Python.

Status	Implementation Method	Pros	Cons	Source
	Flask	Every team member is well versed and knowledgeable in python, would take very little learning to get working immediately	Not every team member has worked with flask	https://flask.palletsprojects.com/en/2.1.x/
	Ionic Angular Framework	Clean look, simultaneous android, iOS, and web development	Can be restrictive, uses TypeScript	https://ionicframework.com/docs/components
	Angular	Easily transferrable to mobile many built in features and libraries, Google JavaScript API compatible, JavaScript features available	Not every team member has worked with angular	Angular.io

- **Server space / services that will be utilized to handle web requests**
    - For the server, the team plans on using Google Cloud Platform[6].
    - Google cloud platform was chosen over AWS due to previous experience and cheaper resources available for the team.

Status	Implementation Method	Pros	Cons	Source
	Google Cloud Platform	Cheap, team members have experience with it.	Not as user friendly as AWS	https://www.cloud.google.com
	Amazon AWS	Free student account, user friendly	Not as cheap as google cloud	https://aws.amazon.com

Best Method Summary:

These decisions are based on cost, ease of use, ease of creation, and the team members' previous experience. For Python Flask, the pros of all the team members being well-versed in python outweighed the use of any other language or framework for creating a web application. The reason a web application has been chosen as the development environment for the app was that Python Flask (a web development framework) works very well for quick development, and web applications are the most inclusive for users because browsers are ubiquitously available. In addition, with extra time near the end of development, a web application can be ported to mobile. The team decided on the various APIs mostly due to cost overhead, and partly as a result of

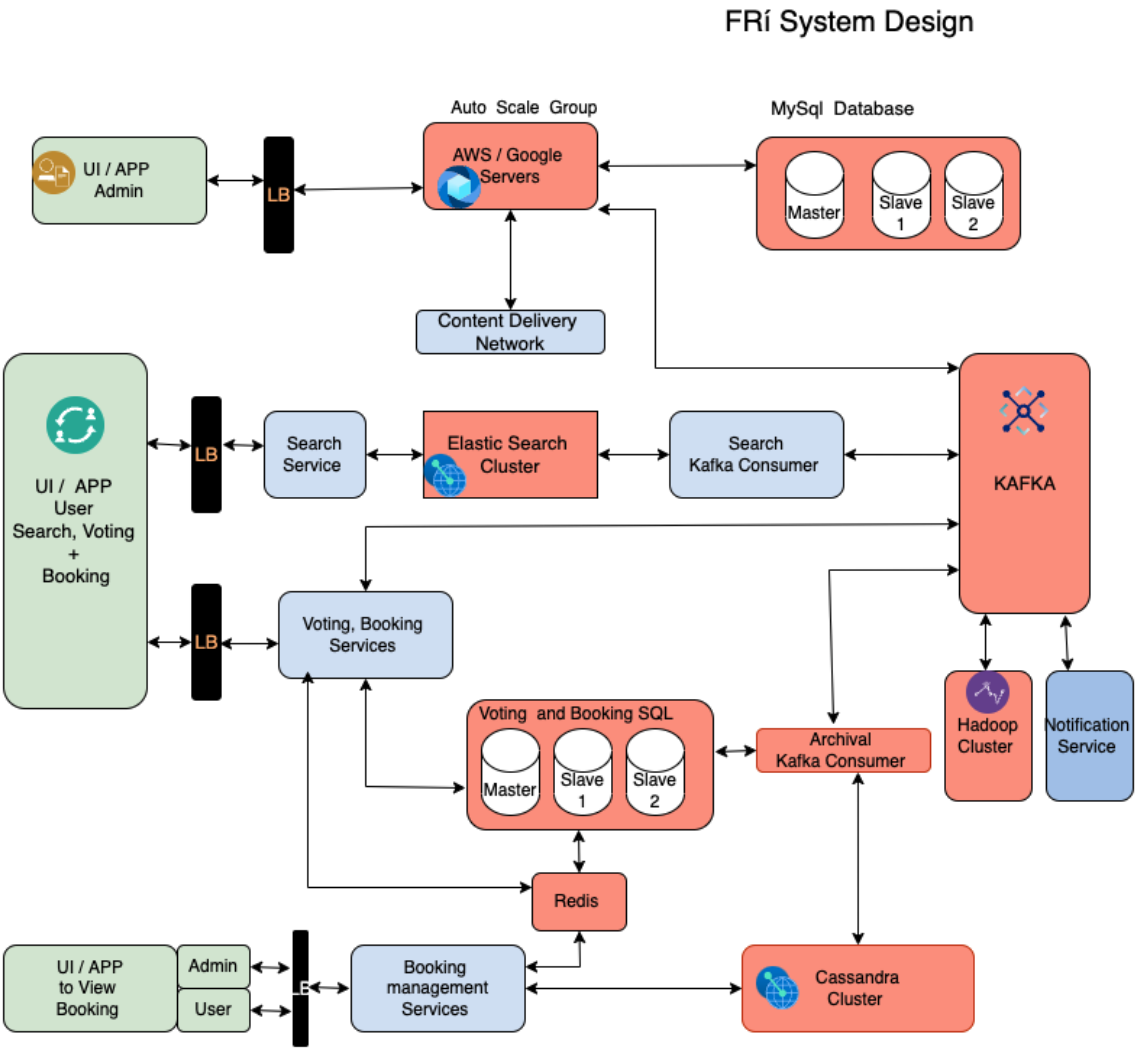
restrictions from other APIs not allowing student access. MySQL was chosen for ease of use during development as a backend service, and because it is easily interfaceable with Python.

Possible backup/contingency choices have also been chosen. Google Flights API, Google Maps API, and Mapbox API will all be options if the current choices prove difficult or impossible to work with due to unforeseen circumstances. If Python Flask is not viable for the web app for any reason, Angular Ionic using TypeScript will be used. If Google Cloud costs too much for the application size, or there are development issues involved, Amazon AWS will be used to host the team's server.

Design and UI/UX

Introduction: This section comprises of the system architecture graph for the application, flowcharts that will map out the decisions of the user in the application, a backend entity-relational diagram that will match the application's MySQL database, pseudocode for different functionalities, and UI maps/mockups.

System Architecture Graph:



Stack:

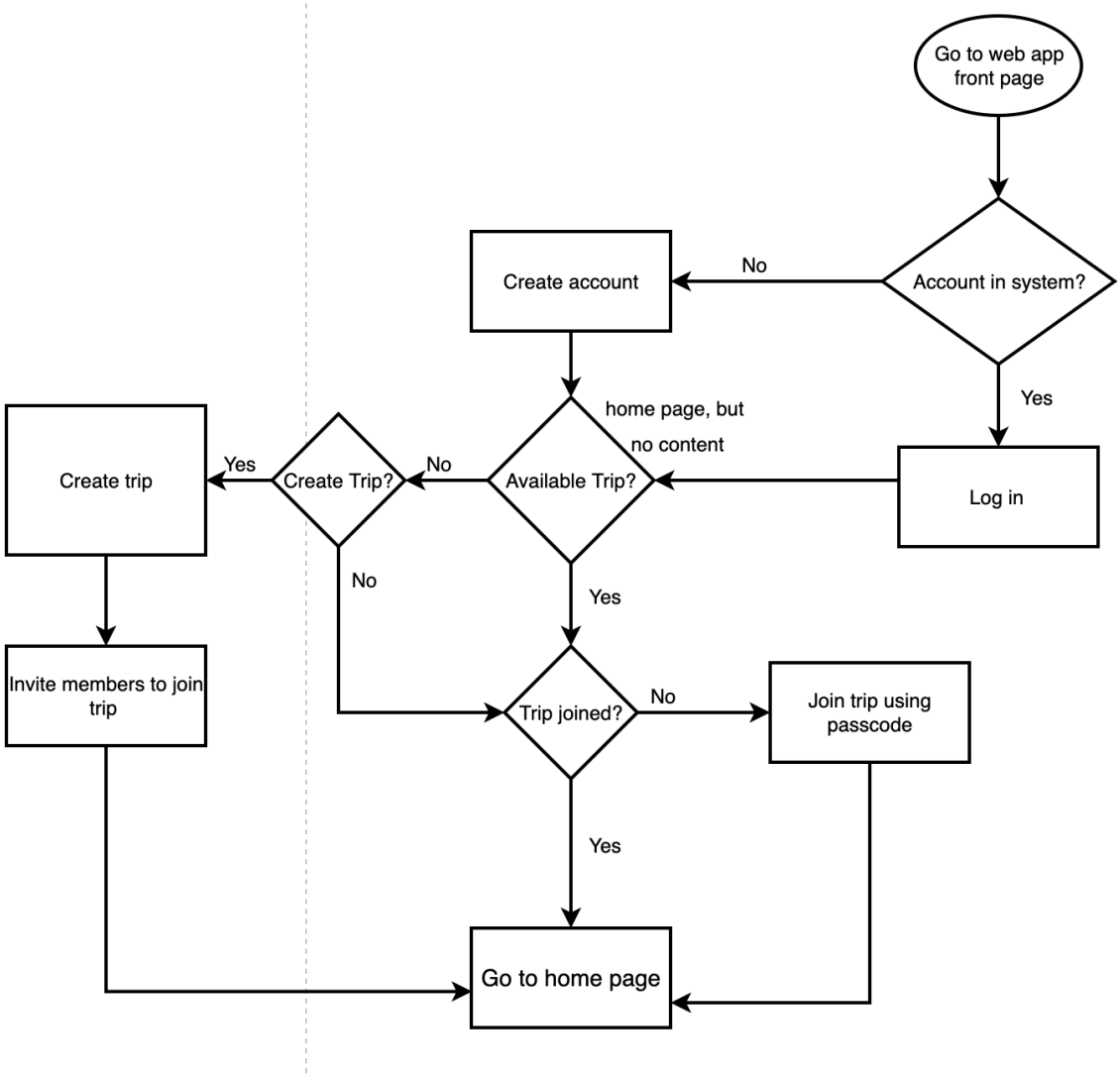
Python paired with Flask micro-framework will cover both frontend and backend components. Visual components will be designed through Axure RP 10.

- Database: MySQL relational database
- Server: Apache HTTP server hosted on Google Cloud
- Libraries: Flask includes a majority of the needed libraries, may utilize other standard Python libraries such as math, beautiful soup, and scrapy for web scraping data from travel websites

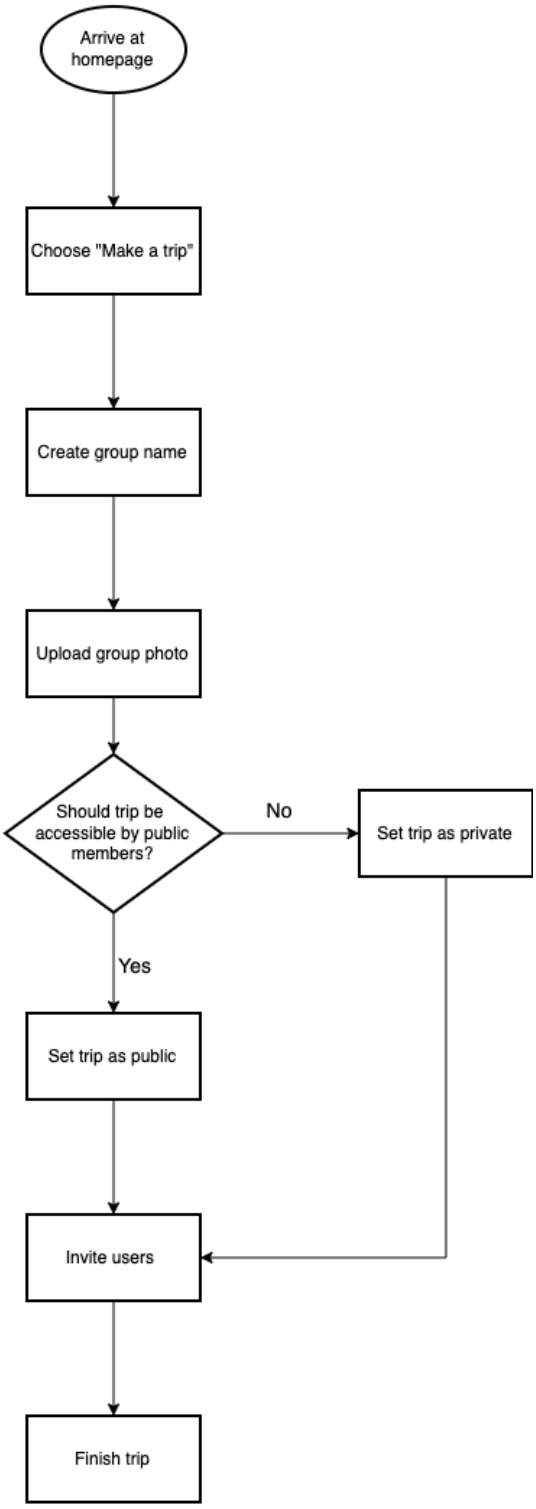
Decisional Flowcharts:

Each of the following flow charts show a decisional path a user will take for a certain part of the application. Ovals in the diagrams indicate the starting step, diamonds indicate points where a yes or no decision can be made, and rectangles indicate atomic (unchangeable) steps that are taken by the user.

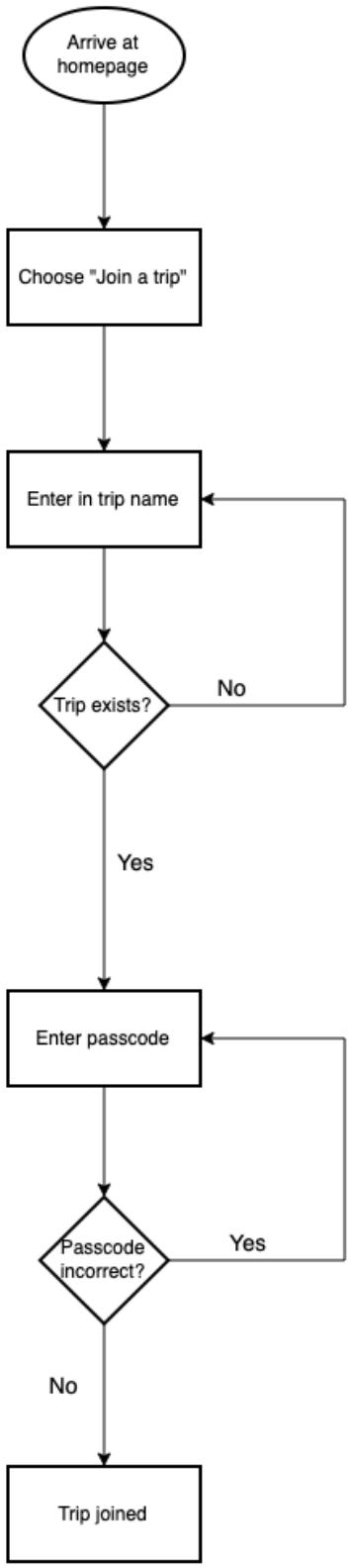
The first flowchart depicts logging in to the homepage of the application and checking for an available trip. The user will then decide if they would like to create or join a trip, and then execute the actions that will lead them to the home page:



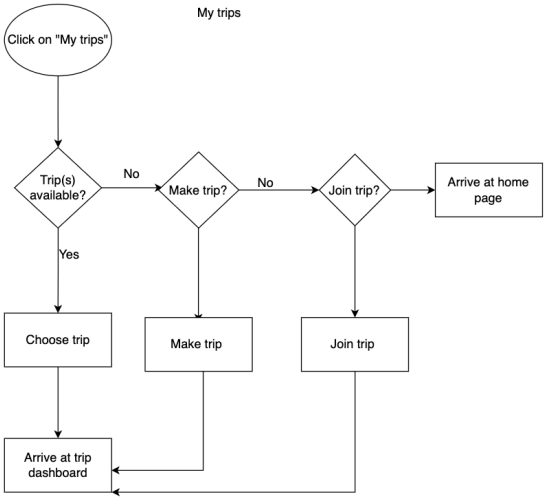
Creating a trip from the home page:



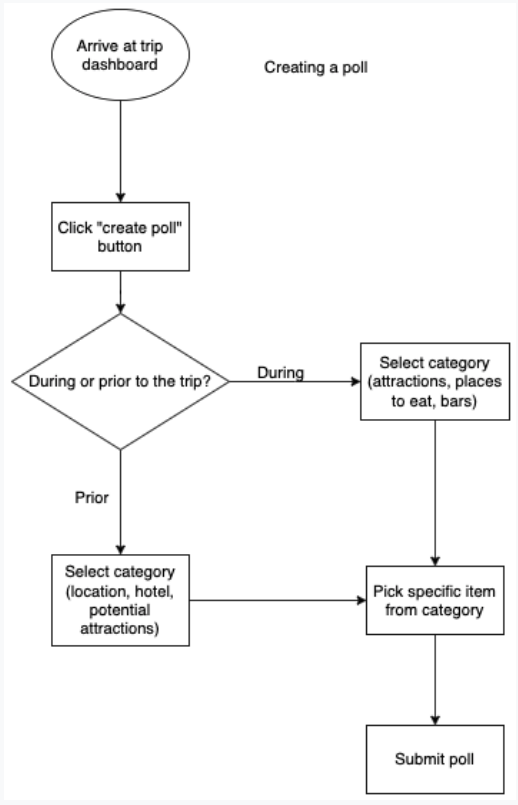
Joining a trip from the home page:



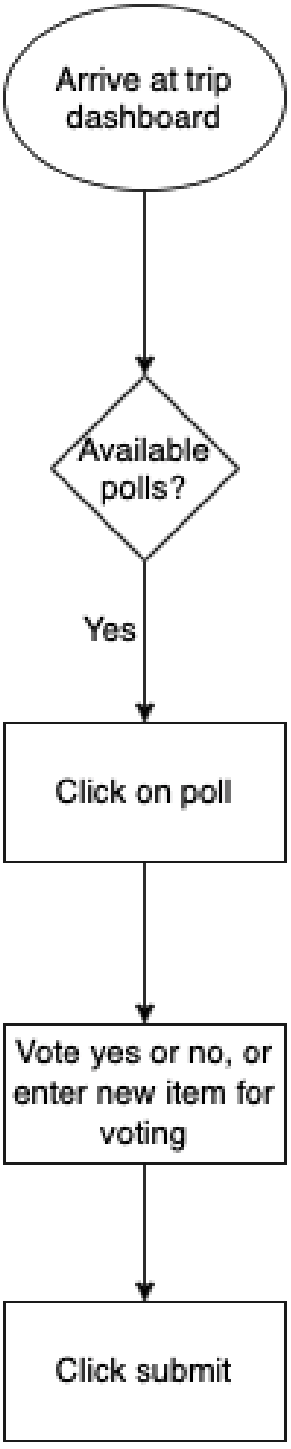
Choosing a trip from My Trips:



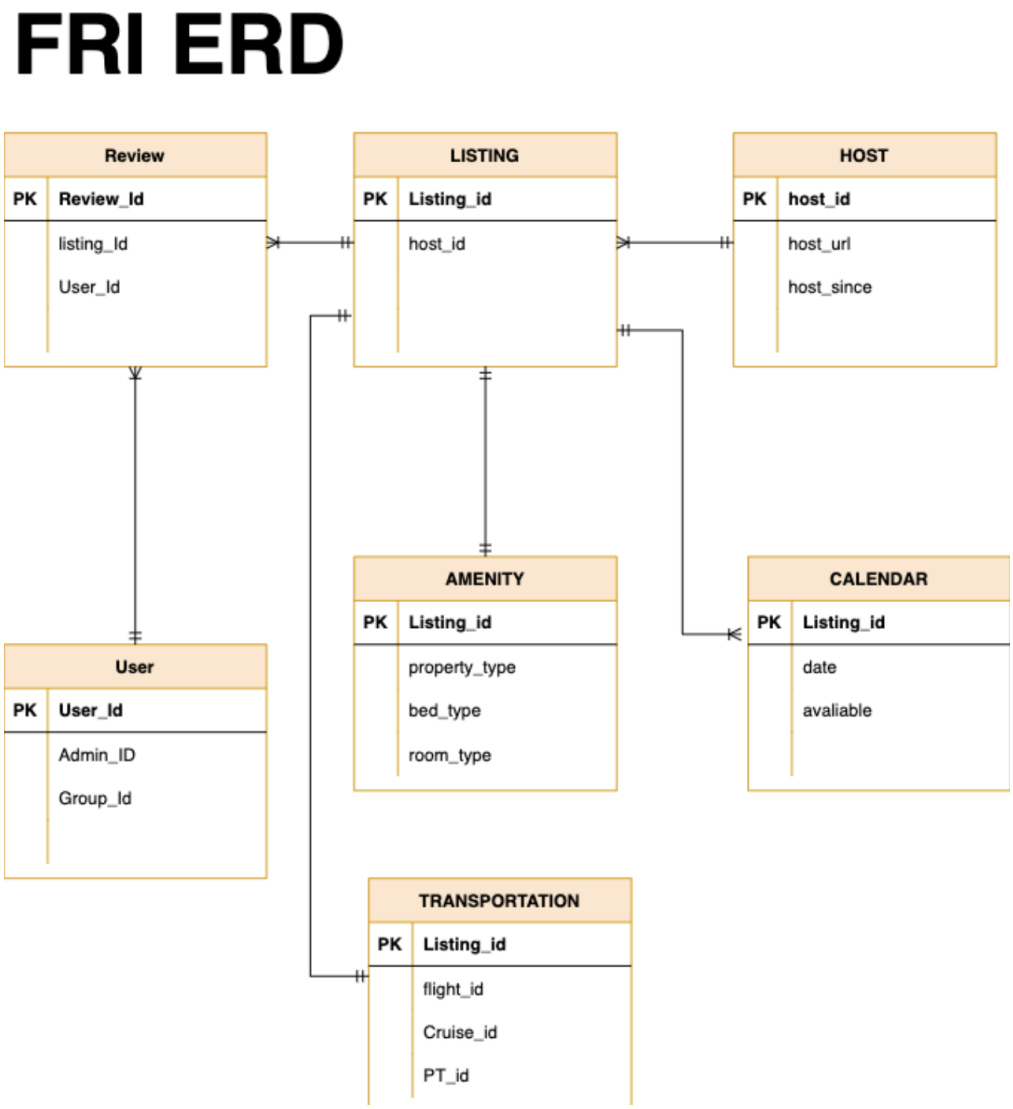
Creating a poll from trip dashboard:



Participating in a poll from trip dashboard:



Backend Entity-Relation Diagram



The above diagram depicts how a relational database is structured for this application. The user entity is linked with the listing entity creating many to many relationships. Unfortunately, it's not directly possible to implement this kind of relationship in a database. Instead, you have to break it up into two one-to-many relationships, these relationships occur when a record in one table is associated with multiple entries in another. The review entity is created to serve as the link table between the user entity and the listing entity. Each record in the link table would match together two of the entities in the neighboring tables (it may include supplemental information as well).

HOST −−owns−−→ LISTING signifies the ownership relationship between hosts and listings. Edge direction is important because data can lose its meaning if relationships are created incorrectly. For example, a listing cannot own a host. Overall, the following edge types are allowed by the schema in Figure 1.

- USER −−wrote−−−→ REVIEWS
- REVIEW −−review_for−−−→ LISTING
- LISTING −−has−→ AMENITY
- LISTING −−has−→ BOOKING-DETAILS
- HOST −−owns−−→ LISTING

Pseudocode

Register and verify a user based on user input data, generates user ID.

```
Member:Register(Username,UserID,Password,Email,Phone Number)
   1. Check input parameters
   2. Http POST request to server with parameters
   3. Return success
```

Create a travel group tied to the leader (admin)’s id generated by the app. Can call add member function upon initialization. Private means admin needs to approve of adding members, public means any member can add others.

Admin (subclass of Member):Create group(Group Name, UserID,Group photo (optional),Private/Public)

1. Check input parameters
2. Set photo to default if none specified
3. Set group attributes
4. add_member(admin) with admin permissions
5. Create GroupID
6. Initialize backend database
7. Return success

Will send invites out to the associated username or email regardless of group settings.

Admin:Add member(s)(Username/Email,UserID)

1. Check input parameters
2. For each username, check for userID
3. If userID, send email to user email,
4. Else prompt for email

Send out invites via email (people who don't have the app) or directly through the app. Only allowed if the group is public.

Member:Add member(s)(Username/Email,UserID,Private/Public)

1. Check input parameters
2. If public, Add_member(s)

3. Else return error (no permissions)

Join the group or choose tentative status, generate an user ID. Otherwise decline.

Non-Member:Join group(GroupID, userID/email)

1. Check if group exists
2. If accept,
    a. If userID, add user to database
    b. Else prompt register
3. Elif tentative, add user to tentative buffer, limit permissions
4. Else decline

Enter preferences such as budget and location preferences.

Member:Enter individual preferences(GroupID)

1. Prompt with TBD questions to determine preferences
2. Save preferences under user in database

Delete the group, in turn removing everyone from it.

Admin:Delete group(GroupID)

1. Remove all members (call leave_group())
2. Delete groupID
3. Remove persistent data from backend

A user can remove themselves from the group

Member:Leave group(GroupID, UserID)

1. Remove userID from groupID backend
2. Remove group permissions from user
3. Redirect away from group dashboard in app
4. Remove group from UI

Any member can create a poll to be voted on. This includes the poll settings, what the poll is about, and the voting options.

Member:Create Poll(Start Time,End Time,Options[],Information[], PollID,UserID)

1. Prompt for options[] (axes, choices, other info)
2. Start time = now
3. End time = default or specified in options
4. Create pollID
5. Launch poll, update backend/other user's interfaces

The member who created the poll can edit and change any of the poll settings and options.

Member:Edit Poll(PollID,UserID)

1. Check for PollID
2. Access poll options[] and information[]
3. Prompt for new info

4. Delete information (votes)
5. Relaunch poll

The member who created the poll can delete the poll.

Member:Delete Poll(PollID,UserID)

1. Check for PollID
2. Remove poll from backend
3. Update frontend

Vote on preference of options (flight, location, restaurant, dates, accommodations, etc)

Member:Vote(UserID, Options[])

1. Present options
2. Based on category(flight, location, etc), pick 1-3 (maybe 5) preferences
3. Add preferences to poll information (votes)

User can change their vote on any given poll.

Member:Change Vote(UserID,PollID)

1. Remove original vote
2. vote()

Alter group settings (name, photo, private/public status etc)

Admin:Change group settings(GroupID, Group name, Group photo, private/public status)

1. Check admin permissions
2. Check other input parameters
3. Prompt fields to enter info
4. Set new group attributes

Change status to tentative (unsure if joining), accept (joining), or decline (not joining). Gives member privileges upon joining, gives viewing privileges upon tentative.

Non-member:Change invite status(UserID)

1. Access userID
2. Prompt for new status
3. Call join_group() with new status

Change status of whether the group can see your location or not.

Member:Change tracking status(UserID, Tracking status)

1. Access userID, change attribute in database

View map to see locations of other members.

Member:View other users' locations(User ID, Tracking status,Group map)

1. Open map page,
2. Call map API

3. Access geo-data from database
4. For users with tracking status enabled,
5. Reverse geo-code user, present pin on map

Search any location in the world, the function returns a list of places to stay at the given location. The StayResults list contains a list of locationIDs that make places uniquely identifiable

Member:Search Location(Search entry,StayResults[])

1. Call map API
2. Use built-in search method
3. Present map

User can favorite a location they like and store that information in a favorites list

Member:Favorite Place(LocationID,UserID)

1. Access userID
2. Set favorite as locationID/geocode in database

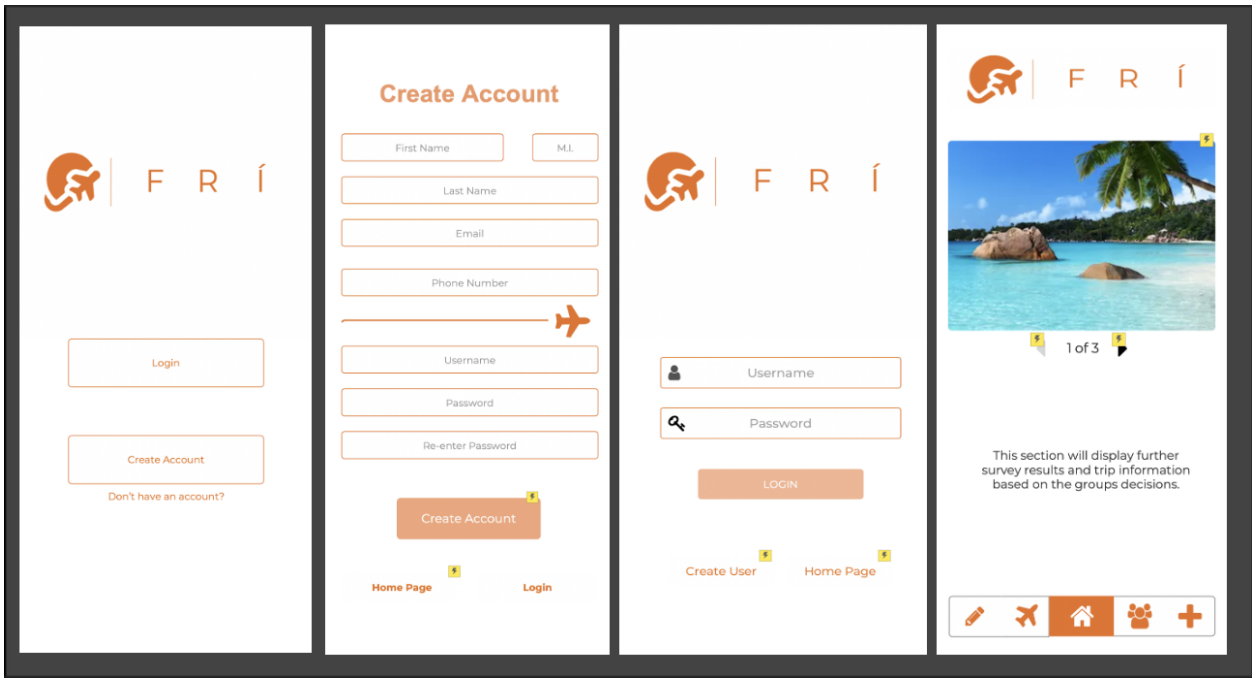
Removes a locationID from the given UserIDs favorite list

Member:Remove Favorite(UserID,LocationID)

1. Access userID
2. Remove favorite locationID/geocode from database

Site Maps and GUI Screenshots

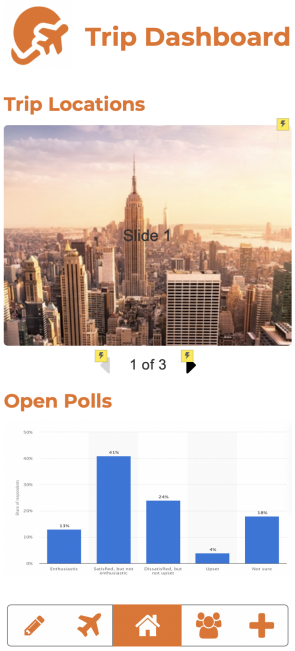
User login and home page:



The above mockup shows the primary screen (far left) where a new user will be taken upon downloading the app for the first time. The second image (middle left) is the "Create Account" page where the user can input their data that the backend database will store. The third image (middle right) is the login screen that a user with an account can use to access the main functionality. The last image (far right) shows the homepage of the app, including a basic

dashboard tour that will have user information and multiple trip's data, and a navigation bar to travel between app pages.

Trip dashboard:

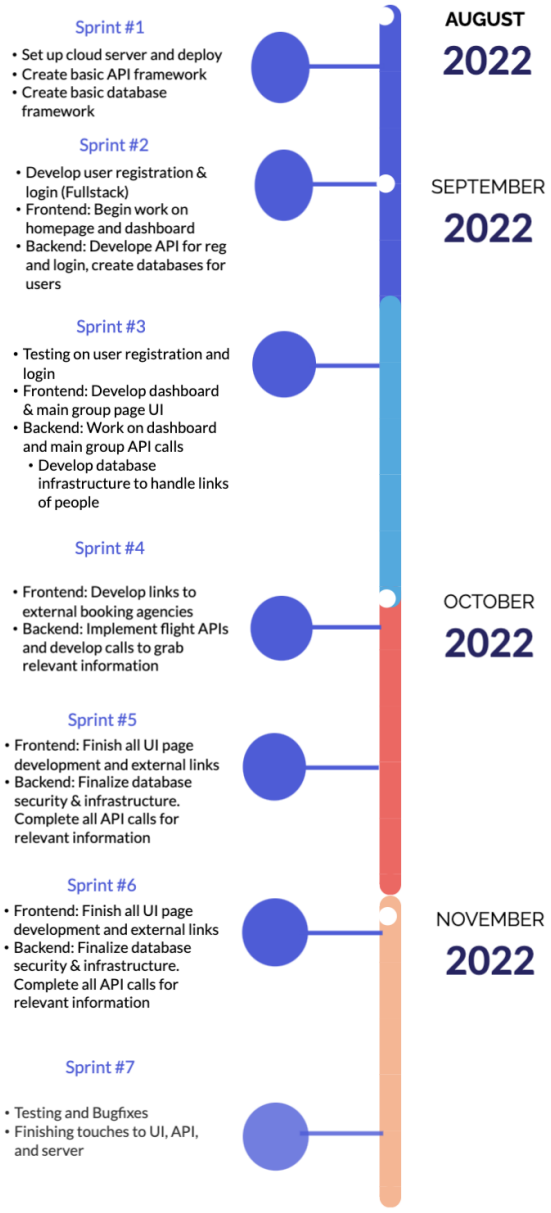


The above image shows a more complete figure of the trip dashboard for a single trip. This will include images, open polls, and decided locations/features of the trip.

Development and Testing

Development Strategy: We will use Agile Scrum because each team member is mostly familiar with it and since it is the most popular in the industry, there is a lot of documentation for it. We will use JIRA for planning and implementing the development cycle. We will be implementing biweekly standups (twice a week) and aim for two week sprints.

Development Timeline\Schedule:



Work Delegation:

- Ankit: Will be working on designing the MySQL database and backend components. This will include initial database development and structure during Sprint #1 & 2, as well as API development and connectivity for Sprints 3-6. Ankit will work with Kent through and in the final sprint work with the frontend team to connect the various components.
- Kobby: Will be working on designing frontend components via Flask and assisting with the database. Kobby will work directly with James to implement the Frontend created during Design, and work as an intermediate with the Backend designers (Kent and Ankit) to ensure connectivity in Sprints 6 & 7.
- Kent: Will work on designing the user interface through Axure RP 10 as well as backend components. Following the initial design Kent will shift focus towards backend development for the majority of the sprints, lending changes and assisting with any frontend design fixes that may occur.
- James: Will also work on designing the user interface through Axure RP 10 and programming the frontend functionalities via Flask. James will domintaley work in the frontend, focusing on design initially with Kent, and then transition in the first sprint towards implementation of the front end. He will also work with Kobby and the backend team in the final sprint to connect the components.

Contingency Planning: Contingency plans are listed earlier in the project related research. The reasoning for choosing our final methods are also shown in this section. If for any reason our chosen methods prove to be unusable or incompatible with necessary features of the app, we will use backup frameworks/tools instead.

Possible Troubles: We plan on using Python Flask for this project, which limits us to web development. Most applications these days can be deployed on both web applications and cell phone apps. Angular serves as a back option since it has a built-in support system that supports both website and app development.

We also anticipate the possibility of running into API problems since most companies either charge exorbitant prices or simply do not allow third party access due to data privacy concerns. Hence we intend to pay for essential API that is needed for this application if they are not publicly provided.

Testing Plan: Blank until next semester

Coming this fall to a Lafferre near you…

Works Cited

1. Ashray Saini, “An Easy introduction to Flask Framework for beginners” Analytics Vidhya, 20 October 2021
2. Yasunori Kirimoto, “Building an Address Search Function with Amazon Location Service” Dev.to, 22 November 2021
3. David Austin, “The Stable Marriage Problem and School Choice” American Mathematical Society, <Unknown Date>
4. Abdulellah A. Alsaheel, “Analysis of Parallel Boyer-Moore String Search Algorithm” Global Journal of Computer Science and Technology Hardware & Computation, 2013
5. Santiago Castro, “5 Reasons Why MySQL Is Still The Go-To Database Management System” Jobsity, 17 June 2021
6. Rajnish Kumar Sharma, “What is Google Cloud Platform (GCP) and Why Should You Choose it?” Net Solutions, 1 June 2021