Real-time Re-Recommendation System for POI Visits
Design Document

Team 22
Client & Advisor: Goce Trajcevski
James Eehuis
Dheepak Nalluri
Andrew Peters
John Smolinski
Geng Sun
sdmay20-22@iastate.edu
https://sdmay20-22.sd.ece.iastate.edu

Executive Summary

Engineering Standards and Design Practices

1. IEEE 12207-2017 - ISO/IEC/IEEE International Standard - Systems and software engineering -- Software life cycle processes The standard above establishes a common framework for software life cycle processes. It includes processes, activities, and tasks that are to be applied during the acquisition of a software system, product or service during supply, development, operation, maintenance and disposal of software products with the involvement of stakeholders to achieve customer satisfaction.

2. IEEE 14764-2006 - ISO/IEC/IEEE International Standard for Software Engineering - Software Life Cycle Processes – Maintenance The standard above describes detailed management of the maintenance process in the software life cycle processes. It also includes different types of maintenance.

3. IEEE 29119-5-2016 - ISO/IEC/IEEE International Standard - Software and systems engineering -- Software testing -- Part 5: Keyword-Driven Testing The standard above defines a consistent solution for Keyword-Driven Testing to allow the process of software testing to be much more efficiently.

Summary of Requirements

- Map/Routing System
- POI listing and targeting system
- Database interface

Applicable Courses from Iowa State University Curriculum

- Com S 309
- SE 319
- Com S 363
- Com S 327
- Com S 228
- Com S 227
- Com S 311
- SE 329
- SE 339

New Skills/Knowledge acquired that was not taught in courses

Android Studios
Server Setup

Table of Contents

1 Introduction

1-1 Acknowledgement

This project is sponsored by Iowa State University. The project client and faculty advisor is Goce Trajcevski. The final product will belong to and is owned by Iowa State University.

1-2 Problem and Project Statement

Product wanted

The product that is to be derived from this project is an application that can handle the routing calculation and maintenance of multiple points of interests from one point to another. Users should have access to customizable timing and route re creation for quick and easy to understand creation.

General problems

Application needs a cohesive design for ease of user access to the setting of both route requesting and creation. These UI designs need to be able to help the user to quickly set not only multiple points, but the timing constraints they have on said points in order to request accurate route information.

Proposed Solutions

We decided upon a table based approach to usability. This table will allow us to not only add points for users to go to but also set designated time in which they will leave in order to get correct eta routes from point to point. This table also is easily malleable by allowing users to simply switch points up and down in the table allowing for quick creation of a full route between any of the points.

1-3 Operational Environment

The end use of this application is expected to work on android devices and browsers. Internet access is required to fully use the functions of the application. GPS location detection is needed to take hold of any feature in the product that requires to know where you are.

1-4 Requirements

Functional Requirements

- Server Side routing able to generate a route with given information from a user
- User side POI listing and timing tables inputting function
- Packaging system to connect user and server easily and send data
- Database interfaces to store information
- Notifications of updates to route

User Interface Requirements

- On click POI information
- Live mapping of location and route use
- POI listing menu and timing input
- Search destinations

1-5 Intended Users and Uses

Users of this application are to be expected to be anyone with mobile android devices or have access to a browser. The product should be scalable to any number of users at a fairly decent cost to person ratio. Users would be using this application to plan out full day's worth of routes and determining the plausibility of certain time constraints and routes. They may also use this application simply to look for places to go under certain terms in a given area and have a route and time given to. These routes are expected to be live so that they may be changed at any time and give the user back the updated map. Finally, all live maps that users are using are expected to be updated with changes when found and notify the user.

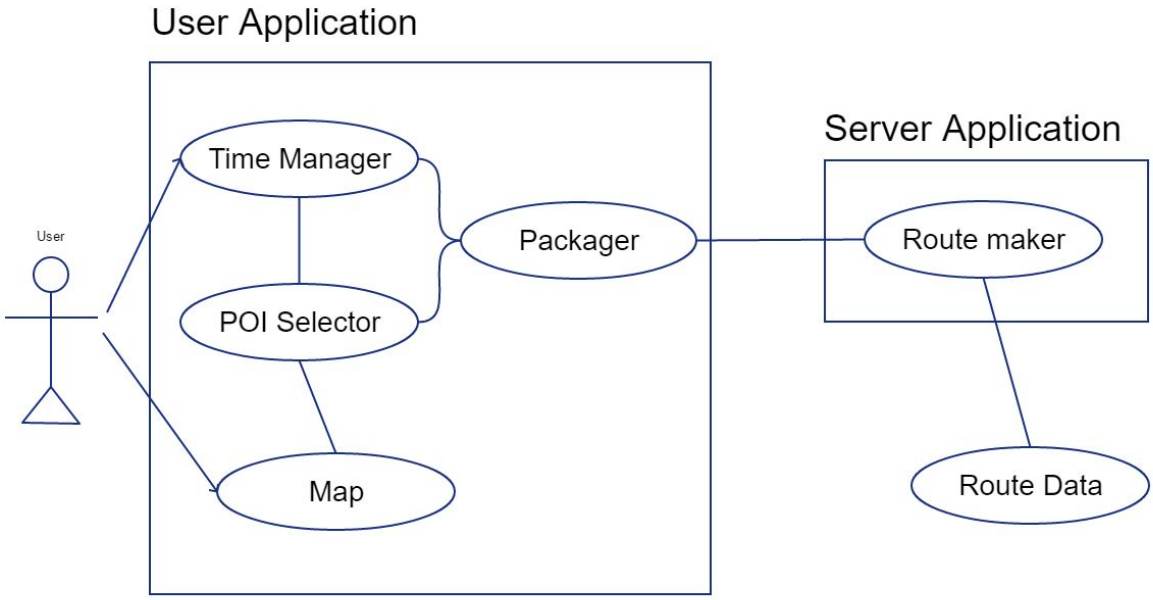

Figure 1.a: Use-Case Diagram for Scenario 1 and 2

Scenario 1 (single user full input use)

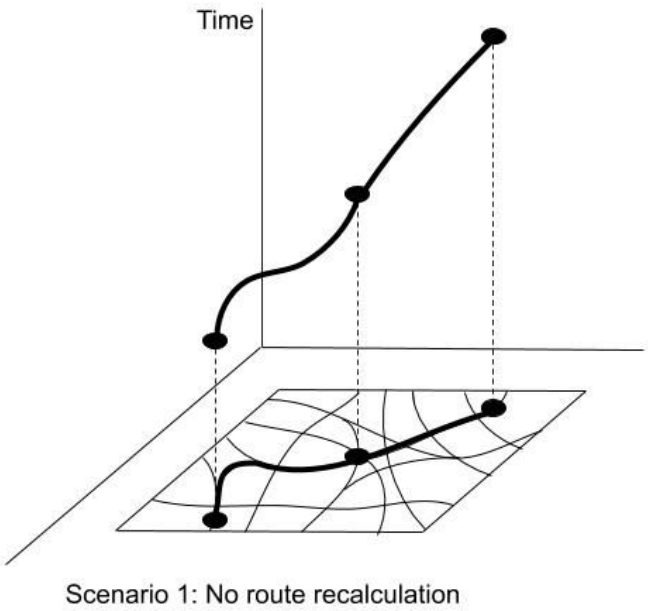

Figure 1.b: Visualization of scenario 1

1. Scope:
    a. POI system
2. Primary actor:
    a. Application end user
3. Use Interests/Intent:

   a. Input a series of points of interest(POIs) to get a route between each of them that is most efficient
   b. All routes must fit the user's input time constraints between POI's (eg Between POI A to B will be from 6:00 pm to 7:00 pm as a constraint)
   c. Users will be notified and updated in changes of route, timing, and/or POI changes while on route.
4. Preconditions:
   a. The POI are given
   b. The user has a connection to the server
   c. The user location is given.
5. User wanted end goal
   a. The user gets routes to the POI(s) in the desired time.
   b. The user gets new routes if the user modifies POI(s) and time constraints change.
6. Main Success scenario
   a. Input the POI into the system
   b. Input time constraints into the system & request route
   c. POI and time constraints are packaged to be sent to the server
   d. Server receives package and request all routes and times (that happen for that route in the time constraint given) from the API
   e. Server packages routes and times and sends it to user
   f. The user gets the package and begins to calculate if the route and time fit within given time constraints.
   g. The user shows all routes between POI(s) and the time between them.
7. Extensions (Alt Routes / fail routes)
   a. User takes out the late POI from route
      i. Update the user map without the route change
   b. User takes out the POI in the middle/beginning of the route
      i. Go back to step C in the main success scenario and calculate the route again with the users current position.
   c. A route within the bounds of user time constraints is not possible
      i. Send back info with time not viable
   d. User spends more time at one of the POIs than expected
      i. Update expected time to destination and check if time constraints still viable
   e. Change in time constraints change
      i. Reroute user from step C.
   f. Poi does not exist
      i. Send user notification of POI does not exist

g. Route between POIs do not exist
    i. Send user notification that route does not exist between the told POIs

Scenario 2 (change of route due to outside factors)

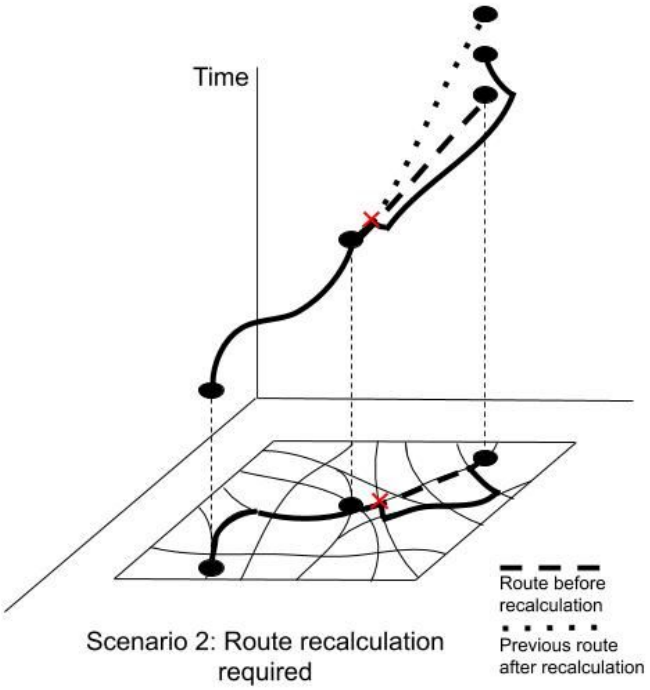

Figure 1.c: Visualization of Scenario 2

1. Scope
    a. POI route system
    b. Does not include user input
2. Primary actor
    a. Application end-user
3. Use interests/intent
    a. Real-time update of changes in the user route or time of routes due to outside factors (eg construction, car crashes, spikes in traffic, etc)
    b. Updates are timely but not overbearing, no need for an update about something that will not affect the uses for several hours till now where it may be resolved. (or maybe the user will have preferences)
4. Preconditions
    a. A successful scenario 1 has happened or a route and other information are there to be checked with new information.
5. User wanted end goal

   a. To be notified of changes in route and/or time of routes.
6. Main success scenario
   a. User system communicating with Server which is receiving data from the outside world
   b. User system asks Server for an update on the current route
   c. Server will check time distribution data against outside world current time distribution data
      i. If no difference in time distribution then tell the user no change
   d. Server will tell the user system that the route has changed and sends the new route data
7. Extension (Alt Routes/ fail points)
   a. Server does not respond for a certain amount of time, the user gets notified of a connection failure
   b. If user fails to meet the time constraint, the system updates the estimated times of arrival
   c. If the user fails to take the directed route, the system recalculates the route and gives it to the user to decide.
   d. If a road is no longer available and no other route exists, the user is notified of the failure

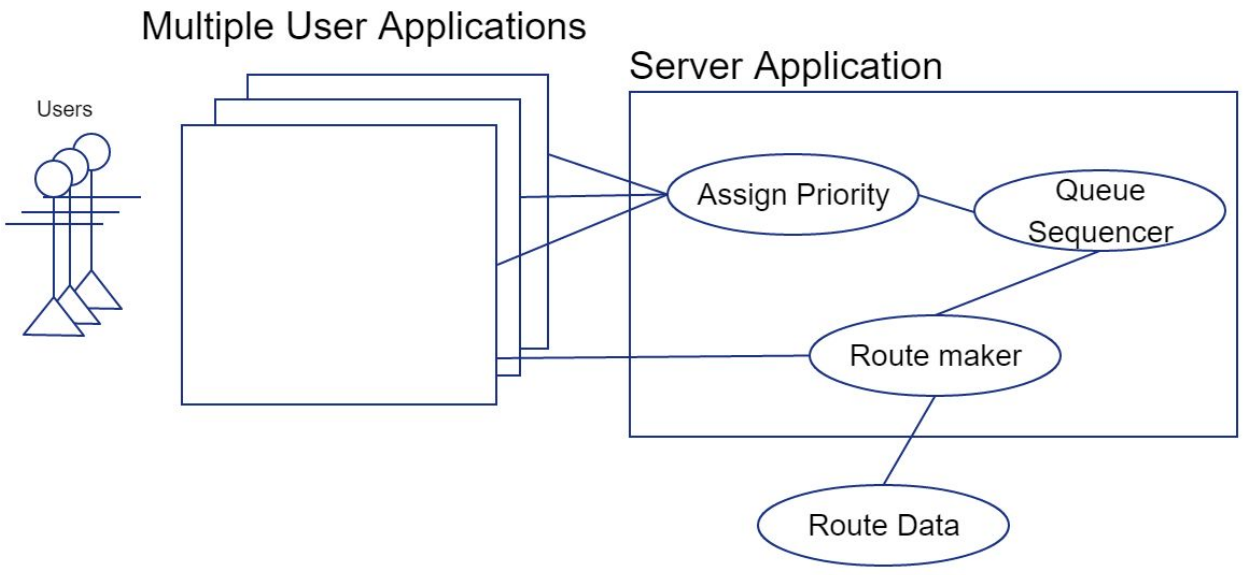

Figure 1.d: Use-Case for Scenario 3

Scenario 3 (multiple users info needed at once)

1. Scope
    a. Server backside
2. Primary actor
    a. Server
    b. Multiple waiting Users
3. Use interest/intent
    a. Multiple users request information from the main info server at once.
    b. Determine who gets their information first if multiple users are in line to receive information from the server.
4. Precondition
    a. This may be happening during any other scenario where information is gained from the main server.
5. User wanted end goal
    a. Get the information they need in a timely manner
6. Main success scenario
    a. Multiple user systems send requests to the server
    b. Each request is time stamped on when they arrived.
    c. Each request is put into a queue for process on the server
    d. Based on the type of request their priority to be processed will be different
    e. Based on constraints (not yet defined) same type request will then be weighted
    f. Time of arrival will outweigh all other priorities after a set amount of wait time if a request hasn't been processed.
    g. Top priority requested based on previous factors will be done first
    h. After a request is done, the queue and priority list will be updated accordingly.
7. Extensions
    a. Failure scenarios.
        i. If a user can not be found to send the request info back to, info is dropped.
        ii. If information can not be found for a request see scenarios 1 & 2 for details (dependent on type) of what to send back.
    b. Other paths.
        i. If a user is seen to have more than one request in a queue all other except the last request are dropped from the queue.
        ii. If users have the same timestamp for priority pick at random the user to process first.
        iii. If other constraints are the same to other requests check the timestamp for who waited longest. They get processed first.

c. Constraints to note
    i. Users should be limited in how often a request can be made for updates to current maps.
    ii. The server should have an upper and lower bound in attempts to send info to a user before dropping it.
    iii. If the server queue is full a message to those not able to get in will be told so and advised that request may take longer than expected.
    iv. Users are only able to have x amount of POIs that can be routed at once unless they agree to understand that more POIs will take longer to gather info on.

1-6 Assumptions and Limitations

The assumptions for this project are listed:

- The maximum number of simultaneous users is currently unknown but the project is scalable
- Having access to databases for routing and mapping purposes
- Having access to the server for user prioritization of information

The limitation of this project are listed:

- This app will be used within a limited area which means the user will only use this in the USA
- The POI will be generated based on Google API and social communication tools

1-7 End Product and Deliverables

This project has multiple end goals and deliverables. First, of which is a context-sensitive suggestion system, this system will allow our users to ask it for optimal POI routes. The system will look at the user and data given from the outside world and generate an optimal route for the user. A second major key deliverable for this project is a mobile application that will enable our users to use the system described above. Currently our expected end product will be a Meta-Learning or Deep-Learning module, however, our group is lacking experience designing things of this nature. So if we are unable to progress with Deep\Meta learning we will look into making a reactive based system instead.

2 Specifications and Analysis

We discussed the general overview of the project.

2-1 Proposed Design

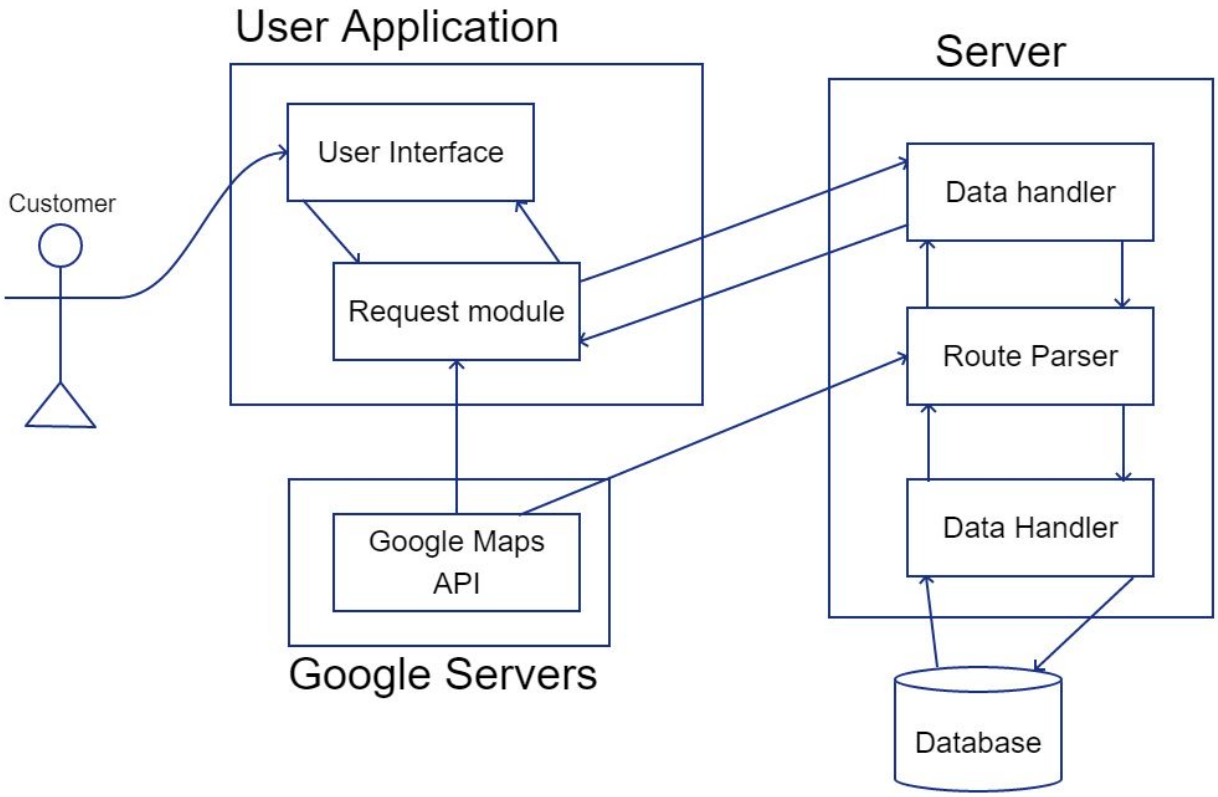

Figure 2: System Overview

The system consists of two main sections: the front-end portion and the back-end portion. Taking a look at Figure 2 above, our front-end consists of the user application that the user would be interacting with. While our back-end encompasses our server, the Google servers requesting, and our database.

The User application is made up of two parts, the user interface and the request module. The user interface displays three main elements. The first user interface element is the map display. The second element is the search bar where the user will be able to enter their points of interest. The last element is the list of current points of interest that the user has selected. In order to display the map, the user interface receives the map visuals from the Google Maps API. The request module is the brains behind our application. It is responsible for communicating with our server. It sends the

POI's that the user selects to the server and also receives the route calculated by the server and sends it to the user interface to be displayed.

Our server communicates with Google's servers, specifically with their Maps API as well as their Google Maps Anomalies. Our server takes data from both these input sources.

The server consists of 3 different parts: controller, route parser, and data handler. The controller is responsible for converting the request messages to actual information that the route parser can use. The route parser then takes this information and creates the optimal route by using both the Google Maps API and the Google Maps Anomalies. The data handler takes the information from the front end, organizes it, then stores that data.

2-2 Design Analysis

Our design analysis so far has been working because our prototype system architecture comes together in an explainable fashion.

One of the strengths of this project is scalability. If this project is expanded in the future to handle a larger amount of users. Another server can be strategically placed geographically, increasing the number of users the system can handle while also increasing the performance of users with a greater amount of distance to the server. The databases will be inconsistent, however, the data collected will be used for the routes that affect the users that use that specific server. As such, the synchronization of databases will be unnecessary as the synchronized data will be largely unused.

Weaknesses of this design are the intolerance of system failures. Route miscalculations are hard to detect failure for certain scenarios as a large number of alternate routes must be recognized and analyzed. Bugs may likely exist in the final product that are invisible to both the developer and customer as resources for this project are limited. Byzantine failures will also go largely unchecked as of this current design. Server capacity is limited and would multiply the resources that a request would occupy. This design is still the best option because it satisfies the customer's requirements and constraints within the resources allocated for this project.

Continuing, we will start designing the system architecture in more detail (Exploring subsystems).

2-3 Development Process

The first semester consisted of designing the project. This development was done using the waterfall model. The waterfall model was used because the design of the project is a step by step process that does not have a large amount of time variability. The waterfall model also does not have the large overhead of meeting every morning, making it the optimal development model for the project design for the first semester.

The second semester was done using an Agile method, specifically Kanban. Since we spent the first semester planning the project, planning and discussing the project in the second semester was not needed. The Kanban development process was kept track of using a Trello board. Kanban specifically cuts down on the number of times that the team needs to meet, so that the team can spend more time working than talking.

2-4 Design Plan

The plan for our design was composed of several milestones, each one consisted of a process of quality checking at each completion point. As mentioned above our system was composed of two main sections, and as such we have developed a series of steps to achieve our goals with each one.

Our first section that received the focus of development near the beginning was the backend system. This system saw most of our focus since we have determined it to be the backbone of our project. The first main milestone that we laid out for this section is the development of a clean and useful database. Leading to the next milestone which consisted of seeing that our backend system could communicate with some mock front end. This communication was a simple one way from server to mock points. Once we accomplish the task of sending data down, we shifted gears to focus on the process of sending data up. Making our next milestone the creation of a reverse round trip, meaning that we developed a mock front end to demo out our functionality of data being sent to our server and reacting accordingly. These are the primary milestones for this section.

Our second main section as listed above is our Front-end map UI. This system is dependent on the milestones listed above, and thus our secondary main focus and

received less attention at the early stages. Our first main milestone is the task of displaying data onto a map. This task consisted of using the data created in our backend and plotting it out for the user onto our map. Our next milestone for this section is to enable conversations to be established between our front and backend. This consists of an established protocol by which the front end and back end can standardize communication.

All of the milestones given above are a general idea of how we expected to design our project. The milestones allow for more indication as to why we labeled our major sections as major sections, and give a more clear indication as to what our plan looked like during development. This plan meets the client's requirements because it satisfies the customer's needs for the product to function.

3 Statement of Work

3-1 Previous Work and Literature

Similar Products

- Google Maps
- Bing Maps
- Waze
- Here WeGo
- MapQuest
- CityMapper

Our project differentiates from the existing products on the market through ease of use. The biggest added functionality to our project is the ease of use provided when trying to find nearby POIs. As such, our product differentiates by providing the user with more convenience.

3-2 Technology Considerations

Tech usage advantages.

- Android UI API and Google Map API makes it easy to construct a useful GUI for us to use for both inputted information and output view for a MVC
- Java nodes will help with server creation with it able to hold and use a database for use in this project.

Drawbacks

- We had to learn how to properly use and understand the limitations of the APIs that we intended to use.

Other considerations

- Currently directly having the user's phone request information from an API could work but testing is needed.

3-3 Task Decomposition

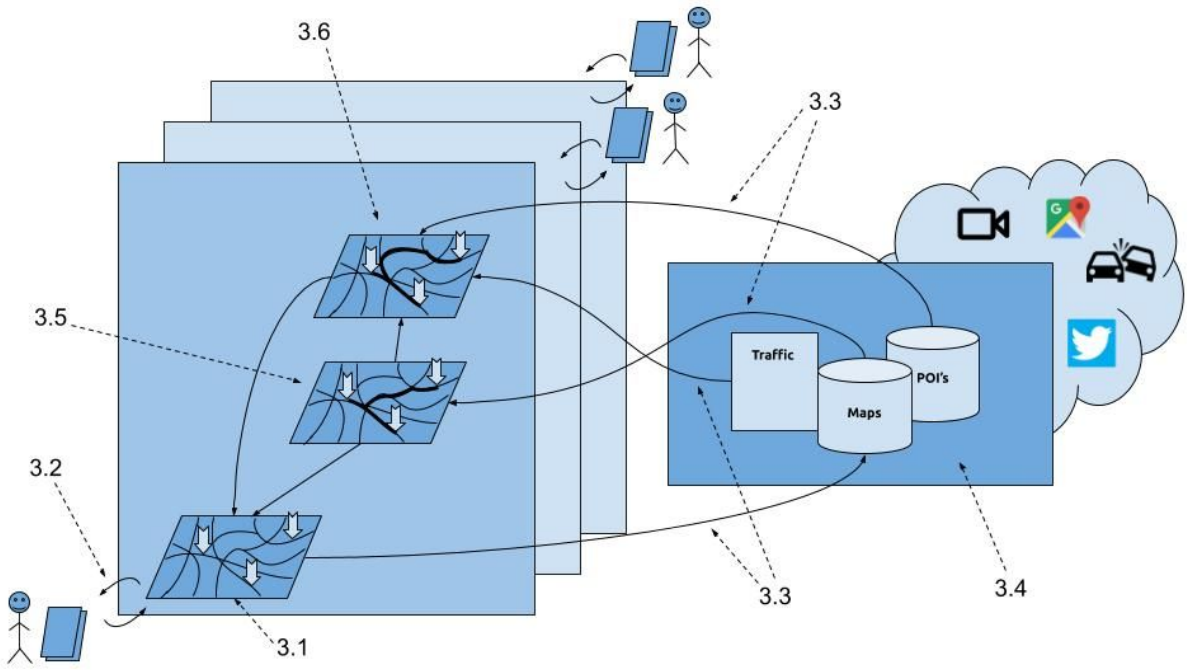

Figure 3a: System Diagram

Each member of the group is assigned a different role to the project
Individual Tasks and Collaboration

- John - Construct a GUI for the front end user to view the Map of their routes on the web client. (3.1 in diagram)
    - Input menu for POI
        - Individual POI routing.
        - Add and remove POIs.
        - Able to change route start point or will continue from the last POI or current position.
    - View of all routes between POIs (3.5 in diagram).
    - Marking routes between the selected POIs (3.6 in diagram).
- James - Update and manage database. (3.4 in diagram)
    - Database is able to handle queries by the user
    - Server can use database for server logic
    - Database stores user entries
- Andrew - Handle web client design for ease of use
    - Create multiple pages for the client to be able to easily navigate to a specific function

    - Web page must be visually appealing to consumer
- Dheepak - Package system
    - Server end able to package and unpack information for transit to the user (3.3 in diagram)
    - User end can unpack a request
    - Read and modify routing data
- Luke - Construct a GUI for the front end user to view the Map of their routes on the mobile client. (3.1 in diagram)
    - Input menu for POI
        - Individual POI routing.
        - Add and remove POIs.
        - Able to change route start point or will continue from the last POI or current position.
    - View of all routes between POIs (3.5 in diagram).
    - Marking routes between the selected POIs (3.6 in diagram).
- Dheepak - Server setup and logic (Involves 3.4 in the diagram)
    - Server has to satisfy web page requests when HTTP requests are made
    - Server must handle the logic of the server
    - Server must be accessible to the clients and running at all times

3-4 Risks and Risk Management

- Costs
  Our project is mainly going to use Google Maps API. However, Google charges fees based on the number of requests. We may go over budget as we develop more and more functions to our application for using too many requests. In order to prevent our project from this risk, we communicate with our clients to update the cost estimation frequently and use the API requests in a sensible range.

- Knowledge of area
  Most of us are going to use maps API for the first time. We needed to take time to learn how to use it and combine that with our backend. Therefore, the implementation time of tasks exceeded the estimated time. In order to protect our project from this risk, we used sprint cycles to keep track of each teammate's status and update this with our adviser and client. We changed our milestones in time to help our project delivery successfully.

3-5 Project Proposed Milestones and Evaluation Criteria

Project Milestones:

- Server setup
    - A server is set up such that it can receive and send replies to a basic client message.
- Web application
    - The server displays a webpage when connected to on port 8080(HTTP).
- Mobile application
    - A mobile application can connect and retrieve data from the server.
- Have a working framework
    - A working framework consists of a working mobile application, web application, and server that all connect and communicate with each other. This includes having a proper user interface with which the system can be accessed and used.
- Basic functionality map
    - The basic map implements most of the features available in most map routing applications such as delivering a route to a POI to the user. At this point, a user can use the application to get from one point to another.

Our team incorporated an Agile style of development with our client. Each time a milestone gets completed, the project is shown to the client. If the client accepted our design, then we moved forward with the project. If the client rejected our design, we will redesign the issues with the product and show the client the new design.

3-6 Project Tracking Procedures

Trello (cf. 3b)

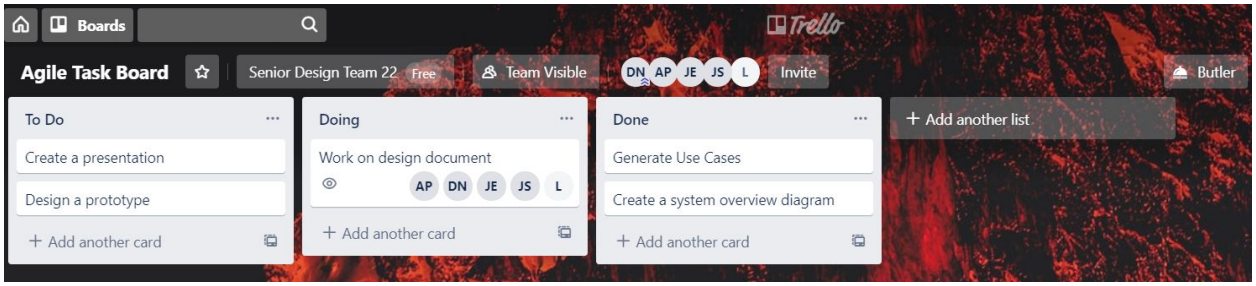

Figure 3b: Trello Board

A Trello board will be used to keep track of task statuses. The Trello board will also be used to discuss issues or details about certain tasks. An example of the Trello board is shown in figure 3b.

Weekly Meetings
Our team will have weekly meetings to discuss things that we have gotten done for that week. Face-to-Face communication is vital as it is more effective at getting ideas across and miscommunication is less common. Our weekly meeting will also allow us to understand overall project progress, whether someone is hitting a roadblock, vital tasks are needed to be done, project redesigns, etc.

GitHub

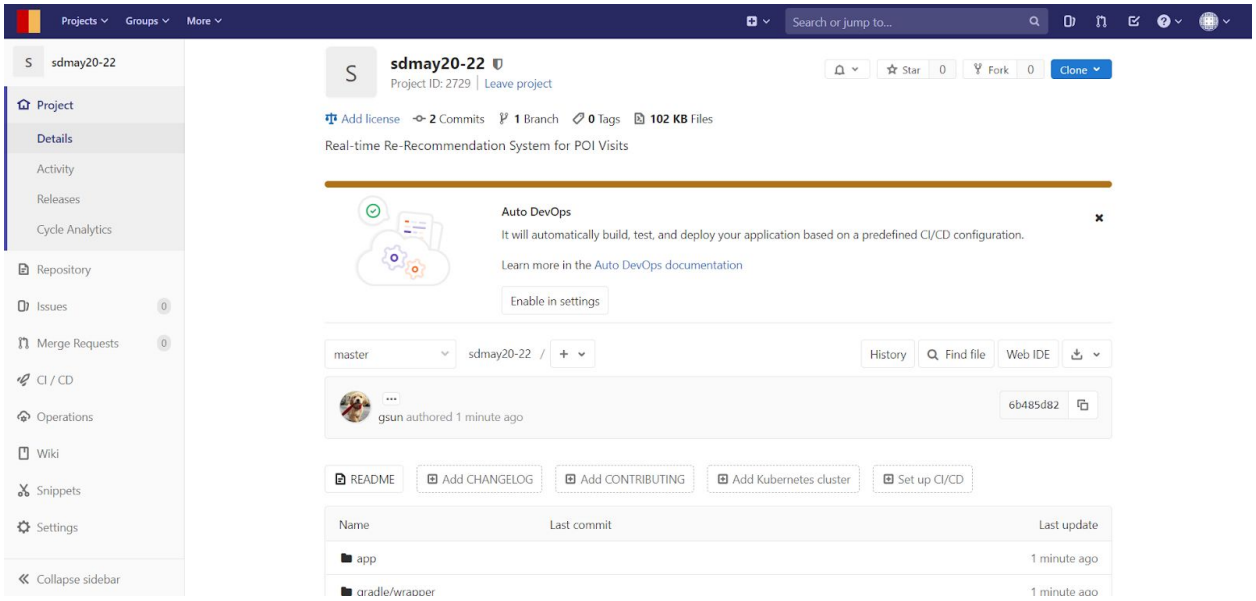

Figure 3c: GitLab Repository

Our GitHub project will keep track of our code for the project and make it accessible for all team members. This will keep track of all changes made, allowing reverts and side projects. A sample GitLab project is shown in figure 3c

3-7 Results and Validation

<u>Expected</u>

1. User sends in POIs and is able to view a route
2. Route made are the most efficient

3. Server is able to process user requests in an efficient manner
4. GUI is user-friendly and easy to use

High-Level Confirmation of work

1. Use of the made system can determine if the POI routing works
2. Comparison between our individual routes and other routing systems can prove this
3. System log manual and auto checking helps with telling if a certain ordering that is desired is found
4. Control group using the system with feedback will tell if GUI is friendly or not.

4 Project Timeline, Resources, and Challenges

4-1 Project Timeline

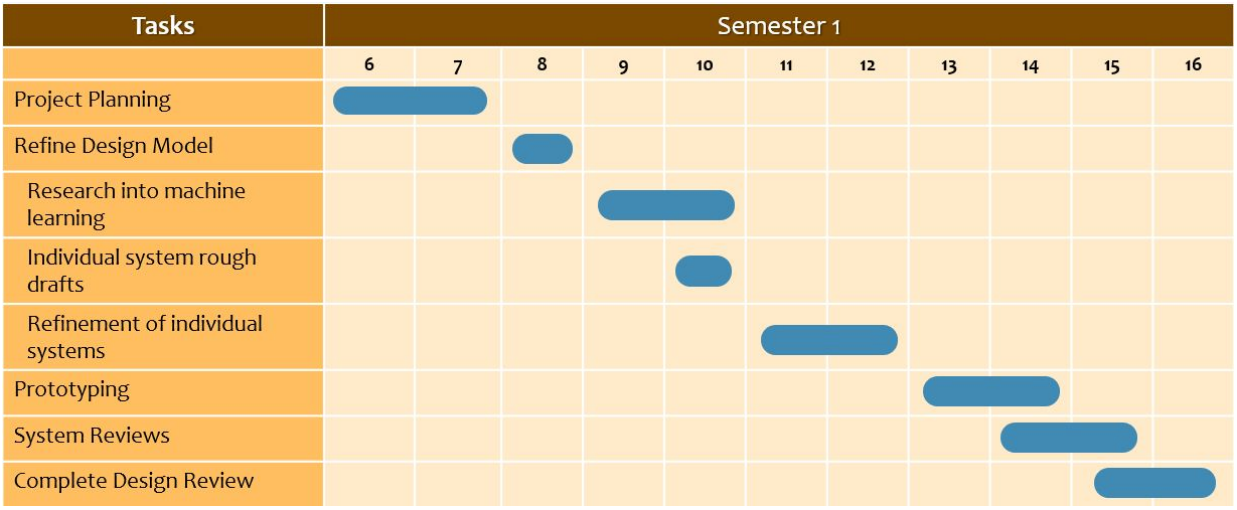

Figure 4.a: Gantt Chart for Semester 1

For the first semester, the team designed the project.

- Weeks 6-7 are project planning which planned the scope and purpose/goal of the project.
- Week 8 refines our design model such that consolidated guidelines are set for the project.
- Weeks 9-12 involve dedicated research and discussion on what software would have been used for the project. This includes what languages and frameworks will be used in the project and the architecture that will allow for the machine learning to be incorporated into the project.
- Weeks 13-14 were setting up the GitHub project with the desired framework and getting environments running. Quick tests were done to test the environment and a prototype of our project will be made.
- Weeks 15-16 were used for consulting our client and getting approval. The two weeks were used to refine any parts of our design that needs refining and completion of a design document.

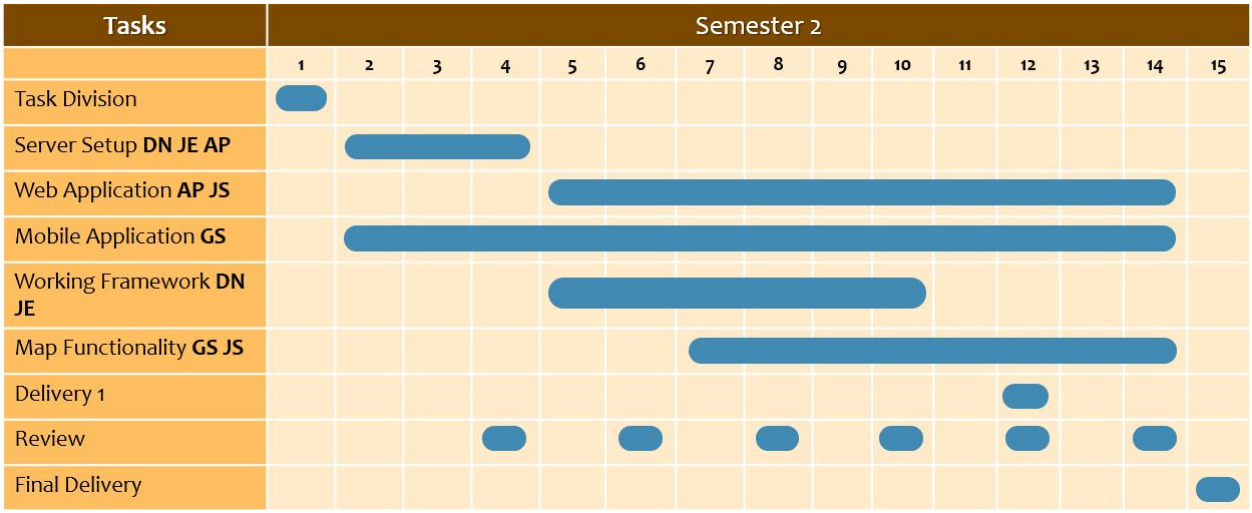

Figure 4.b: Gantt Chart for Semester 2

The second semester consisted of most of the project's development using the Agile process.

- The first week was used to divide up tasks and get all members of the group familiar with the environment. Basic tasks were given to each member as small warm-ups.
- Weeks 2-4 sprint were used to set up the server. The beginning of the project was linear in design. Most of the pending tasks that had to be done depended on the server, so only 1-2 members were able to work at a time, hence an extended cycle to ensure the server was able to get done.
- Weeks 5-6 sprint were used to develop the web application. This was not entirely strict as some members were working on the mobile application, but the main focus was on the web application.
- Weeks 7-8 sprint were used to finish designing the mobile application. Some of the web application's development spilled over into this sprint, but the focus was on the mobile application and finalizing the design.
- Weeks 9-10 sprint began implementing the functionality of our project, incorporating Google Maps, designing user interfaces, etc. This included establishing communications between the server and each of the clients
- Weeks 11-12 sprint continued the work from the previous sprint and finished designing a basic version of our project. At the end of week 12, our team reached our first version of the project.
- Weeks 13-14 sprint worked on adding more features to the project. This included finding the nearest POI.
- Week 15 was the final week and was used to focus on documentation for the project.

4-2 Feasibility Assessment

This project is an application that can take multiple points of interest on a contemporary map and take the best possible routes to get to each place in a timely manner. It needs a server, a user-friendly user interface, Android and web developing platform and Google Maps API to implement. Therefore the challenges of the project are as follows:

- Different platforms and APIs:
  Not all of us are experts in both web and android development and Google Maps API. It needed time for us to learn how to use android and web developing platforms and how to communicate them with Google Maps API. We hit roadblocks on some technical issues as we developed this project, which created a huge challenge for us.

- Cost estimation:
  Currently, we estimate the costs based on what we searched online. The actual costs may vary as we developed and continue developing the application. We had to carefully balance our resources such that we did not incur extra costs

4-3 Personnel Effort Requirements

The development of our project was broken down into several different categories. Each one is listed below with a brief description and the necessary work hours.

- Front End:
  - Mapping UI - the way that our user interacts with our mapping interface, we have both a mobile and web interface for this UI.
  - Total man-hours for UI: 100

- Backend:
  - Database - the system that is used to store user information and all other information that we need to store
  - Endpoint creation - the API that our system will use to pass information around.
  - Total man-hours for UI: 150

4-4 Other Resource Requirements

Our major resource requirement is going to be the system that we choose to host our server on. We have deemed ISUs servers to be most apt for accomplishing our goals. In addition, we will need a testing system, this will come in either an emulated mobile system or a physical system. Both of these will be available to us and we intend to use both of them equally.

4-5 Financial Requirements

Our project is one that has relatively low financial constraints or requirements. The main things that are required for us to have is a running server and database system, as well as the need for some sort of device to run our application off of.

Looking at our first requirement we have discussed several options for hosting platforms. We believed that using ISUs servers would most likely end up being the best option for hosting our server system. The server would be local and support would be offered at no additional cost to us.

For the second requirement, we understand that the Android SDK comes with an emulation system and anticipate mainly using this. However, if our client desires to see that application run off of a physical system we have the capability to upload the application to our own devices and test run it from there.

5 Testing and Implementation

5-1 Interface Specification

Software Interfaces

- Cloud services
    - Runs the server and database
- Browsers
    - Web application should be able to run on various browsers
- Android
    - Operating system hosts application
    - Location services used to find the user's location
- Google Maps
    - Maps API used to display the map to the user on both the mobile application and the web application
    - Maps API interfaces with server to report road anomalies
    - Maps API interfaces with server and provides road data

Middleware Interfaces

- Android
    - Gyroscope and accelerometer to measure user velocity
    - Screen inputs to interact with user interface

Hardware Interfaces

- Routers - Although we are not directly interfacing with routers, the functionality of our product requires the use of them.

5-2 Hardware and Software

Hardware

- Computers - Servers are tested on personal computers before moving to CI/CD
- ISU Servers - Changes to the server after initial testing are implemented and pushed to ISU servers using CI/CD in Github
- Android phones - Applications are tested on personal Android phones before being pushed to the market. This also includes testing of gyroscope and accelerometer and screen interactions

- No specialized hardware is used to test our project as the final product should not be using specialized hardware

Software

- Browsers - The web application is tested by connecting to the server and requesting the webpage
- Android Studio - Mobile application code is run and tested in Android Studio
- Webstorm - Tests and runs the Javascript framework that we are using
- Unit testing - Methods in the server can be repeatedly tested before deploying code such that broken functions are detected more quickly

5-3 Functional Testing

Unit test

- Map(User side)
    - Create points on the map and show them respectively.
    - Able to show some visual routes
    - Have unique identifiers that can be applied to the map
- Routing(User side)
    - Able to tell of different anomalies on a route
    - Holds all points or POI with their given information.
    - POI has either information if a route is possible, not possible, or not computed yet.
    - Information in route tables are organized based on the organization requested by the user.
- Time map(User side)
    - Holds a time of stay for all POIs or tells if the time of stay at a POI is unknown
- Packaging system(user and server)
    - Able to connect directly to the internet and ping server and vise versa to user
    - Given a string from one end the other end can read it
    - If the connection is interrupted a set time will then count down to try to reconnect before dropping all information about a request or send.
    - If no connection is found on the user side give a prompt to reconnect and thus try to reconnect when asked.

Integration testing

- UI (user side)
    - Map can show all routes that are stored in its route map.

    - Routes are all shown together correctly from the route map.
    - Time Constraints are shown correctly on the map when you click on a POI
    - Routes on the map shown give an eta on the time it takes to transverse
    - Menu click from map to input, POIs, time constraints.
    - All POIs are linked to some time constraints, even an empty one.
- Packaging everything(user side)
    - A package has a POI object and user-id
- Server-side package(server-side)
    - Package contains everything from the user side still along with the route and time of routes to send back.
    - User side should be able to store and read all data received from the new route data and time of routes.

System testing

- Update(user side)
    - User system sends a request to the server with current data and is able to receive data.
    - Compare data received from the server with old data and compare them. Based on constraints, update old data with new ones.

Acceptance testing

- Users are able to put in POI(s) only and get a route between all back.
- Users are able to put in POI(s) and time constraints and are able to get a route back or given on with the notification that time constraints can not be met.
- A current route is suddenly not the same as newly updated data due to some anomaly and the user is updated on that new info.
- Two users sent in data with different amounts of data between the two. The one with less needed data should always get data first.
- Given a large number of requests with unknown time of request, the server should properly prioritize who gets data, when based not only on how big of a request but also by time waited.

For the testing listed above, you can find the expected verification results down below in section 5-6.

5-4 Non-Functional Testing

Android and browser usage

- Android app that is used for mobile usage
- A browser-based site to use the application through

Usability

- UI
    - Mobile and browser should be similar to keep visuals and usage the same
    - Only need to get to all needed menus within 3 steps
    - Loading of visuals should be quick and timely
    - All menus should have clear naming and places to put in need information and read already inputted info.

Processing

- Server
    - Test amount of users with minimal POIs request that server can process with reasonable feedback times.
    - Test amount of users able to request 5ish amounts of POIs each and the server can process within a reasonable time.
    - Given a long request from one user a bunch of shorter ones from other users and determine a reasonable time for the long request to wait in a queue to be processed.
- User
    - Map visuals with multiple routes should be loaded once data gained in a quick and fair manner.
    - Updates to the map routes should be

Networking

- Make sure the server has access to the internet enough to properly use the APIs it has and to connect to users
- Test if users can connect to server with the mobile app
- Make sure the browser site can be connected to

We have devised a series of expected results for the tests listed above, you can find these results listed out in detail in section 5-6.

5-5 Process

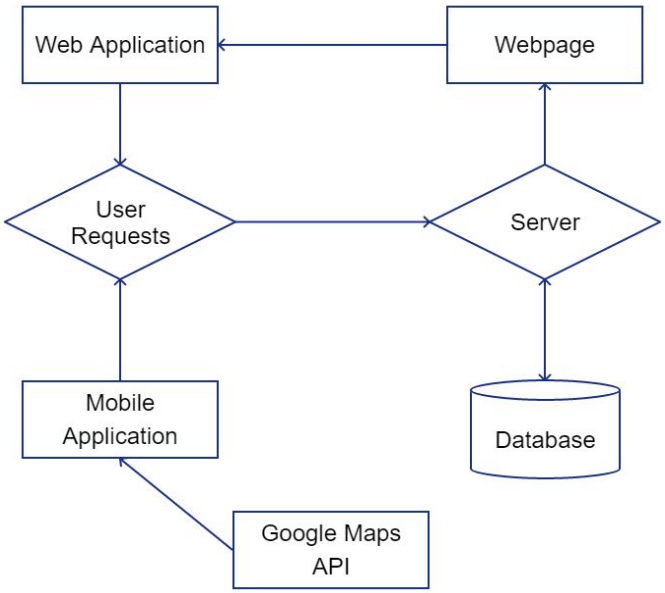

Figure 5a. System diagram.

The server is tested using unit testing to make sure each function is working as it is supposed to. This ensures that issues that persist after the tests that are passed can be narrowed down to the clients connecting to it. These unit tests are automated in CI/CD as tests are rerun every time the server is updated. The machine learning module is also subject to unit testing but is not automated. The unit testing judges the routing system and presents changes and results to the programmer. The programmer then decides whether the outcome is intentional or not.

The web application is tested through common browsers such as Google Chrome or Mozilla Firefox. These browsers connect to the server that is set up using an HTTP or HTTPS connection. The server then delivers the requested web page to the browser, which then displays it. Testing occurs manually through user input into the page. Changes and bugs can be easily observed through the user using that web page.

The mobile application is tested separately on the phone. The user interface is separate from the server and bugs can be resolved using Android Studio. The only communications between the two subsystems are requests to the server and replies from the server. Replies can be tested using the web application as well. Something that works on one application and not the other likely has a bug in the application itself.

5-6 Results

During this verification process, we expect to see that our tests meet the qualifications laid out below.

Functional Test Results

- Unit Tests
  - Map
    - Blips on the map are clickable and interactable
    - Routes appear on the map, on streets, and connecting the correct POI's
  - Routing
    - POI blips remain on the map after the route is created
    - Route is correctly portrayed to user
    - Route status is displayed (i.e. possible, not possible, not computed)
  - Time map
    - Correctly Times the user
    - Alerts user on necessary updates (i.e. missing a time frame)
  - Packaging
    - Application, both mobile and web, are capable of pinging server
    - Data received from the server is readable by the client
    - Time out causes the client to re-ping server in 5 seconds
- Integration Testing
  - UI
    - The UI works the same with test data as it does when backend data is being used
  - Server Side
    - The data sent out by our server remains the same data regardless of where it is being sent to
- System Testing
  - Complete round trip
    - Our user is able to boot up the application and get an initial load from the server with POI's
    - Our user is able to send data to the server and receive data back within a reasonable time frame

        - Our user completes a trip from point A to point B within a given time frame and the system tracks and informs of missed time frames
- Acceptance Testing
    - Frontend
        - We provide our client with weekly updates on the front end
        - Our client has the opportunity to play with the front end and verify we are accomplishing the goals they desire
    - Backend
        - We gave our client weekly updates on how our database is being built
        - We showed our client the payloads that our backend is sending and receiving

Non-Functional Testing

- Android and Browser Usage
    - Android and Browser application allows users to be logged in and requesting routing information in no longer than 90 seconds
- Usability
    - Functionality should be clear and client should be able to intuitively perform all tasks without confusion
    - If connectivity issues occur the user is notified and shown that attempts to recover are being made
- Processing
    - Server
        - Feedback maximum is either met, or user is prompted with an appropriate waiting for a response message
        - Server is still able to meet the request maximum time and the system still correctly returns the “waiting for a response” message
    - User
        - When the payload containing the routing information is received the system draws the route without delay
- Networking
    - Server has consistent internet access and if dropouts occur they are extremely rare
    - Verify that mobile users and browser users have no issues connecting and utilizing our server

6 Closing Material

6-1 Conclusion

For the past semester we have been working on the implementation of our project as well as meeting with our advisor/client. We worked hard in order to develop a product that met the specifications in our design document from last semester. Unfortunately we were not able to fully implement all of those specifications due to the inexperience most of our members had with the software we were working with, as well as multiple other unforeseen circumstances. However we were still able to create a product that fulfilled the majority of most important features. While working on this project we ended up having different roles than what we had originally planned. Andrew and John worked on the front end web application while Dheepak and James worked on the backend server. Luke ended up working by himself with some support from John on the android application. Overall there was a lot that we could have done better during this semester. We could have planned our time out better as well as split the tasks up more efficiently. However, overall this semester has been a tremendous learning experience where all five of us. We have learned just how difficult working on software projects can truly be.

6-2 References

1. “Bing Maps Home Page,” Custom Maps API for Business | Bing Maps for Enterprise. [Online]. Available: https://www.microsoft.com/en-us/maps/. [Accessed: 08-Dec-2019].
2. “Driving Directions, Traffic Reports & Carpool Rideshares by Waze,” Free Driving Directions & Live Traffic Map App by Waze. [Online]. Available: https://www.waze.com/waze. [Accessed: 08-Dec-2019].
3. Google. [Online]. Available: https://www.google.com/maps/about/#!/. [Accessed: 08-Dec-2019].
4. “gmaps.js,” gmaps.js - Google Maps API with less pain and more fun. [Online]. Available: https://hpneo.dev/gmaps/. [Accessed: 08-Dec-2019].
5. MapQuest, “About Us - Your Online Mapping Company: MapQuest for Business,” About Us - Your Online Mapping Company | MapQuest for Business. [Online]. Available: https://business.mapquest.com/company/. [Accessed: 08-Dec-2019].
6. MapQuest, “About Us - Your Online Mapping Company: MapQuest for Business,” About Us - Your Online Mapping Company | MapQuest for Business. [Online]. Available: https://business.mapquest.com/company/. [Accessed: 08-Dec-2019].
7. “The Ultimate Transport App,” Citymapper. [Online]. Available: https://citymapper.com/. [Accessed: 08-Dec-2019].