Route-Constrained Family Shopping Optimization

Design Document

Team Number: 34
Client: Goce Trajcevski
Advisers: Goce Trajcevski, Ashfaq Khokhar
Team Members/Roles:

Tavion Yrjo - Meeting Scribe, Backend Engineer
Colin Willenborg - Frontend Engineer
Erich Brandt - Web Developer
Elizabeth Strzelczyk - Web Developer
Christian Baer - Backend Engineer, Data Analyst
Colin Thurston - Trello, Tester, Algorithm Developer

Team email: sdmay21-34@iastate.edu
Team website: http://sdmay21-34.sd.ece.iastate.edu

Revised: 10/22/20

Executive Summary

Development Standards & Practices Used

- Agile
- Version Control
- IEEE development standards

Summary of Requirements

- A route to drive
- Mobile and desktop application
- Database full of store locations and items with their prices
- Optimized by user spending and distance
- Multiple users

Applicable Courses from Iowa State University

- Software Engineering 185: Into to Problem Solving I
- Computer Engineering 186: Intro to Problem solving II
- Computer Science 227: Intro to Object Oriented Programming
- Computer Science 228: Intro to Data Structures
- Computer Science 309: Software Development Practices
- Computer Science 319: Software Construction and User Interface
- Computer Science 327: Advanced Programming Techniques
- Software Engineering 329: Software Project Management
- Software Engineering 339: Software Architecture and Design
- Computer Science 363: Intro to Database Management Systems

New Skills/Knowledge acquired that was not taught in

- Web api
- Kotlin
- React

List of figures/tables/symbols/definitions (Work In progress)

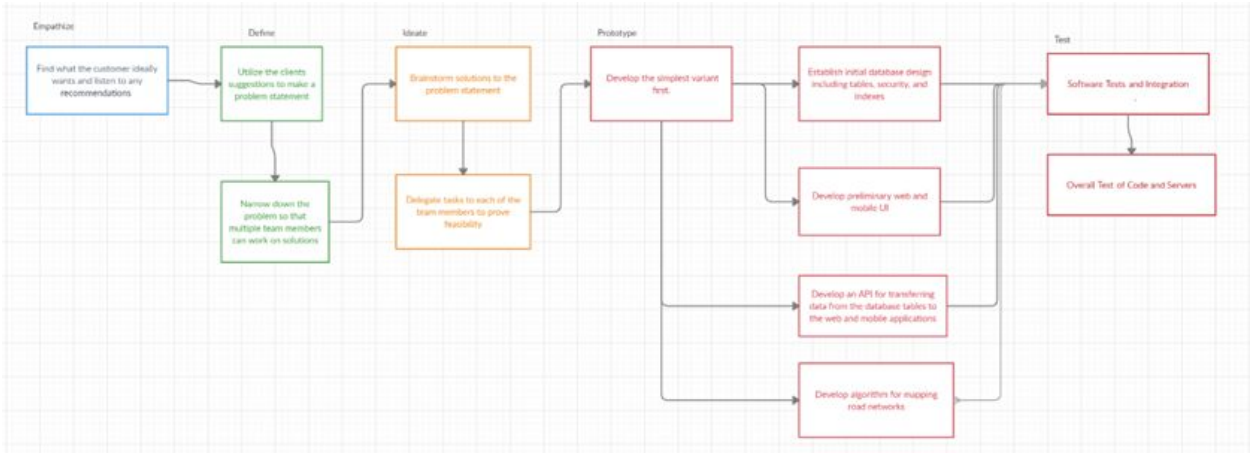


Figure 1 Design Process Diagram

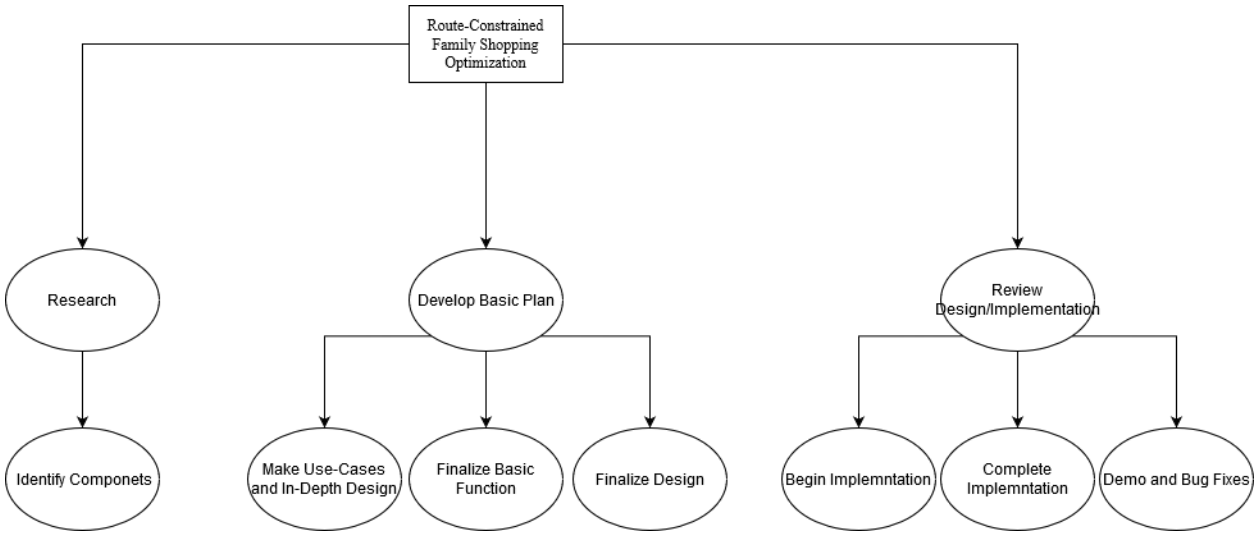


Figure 2 Task Decomposition

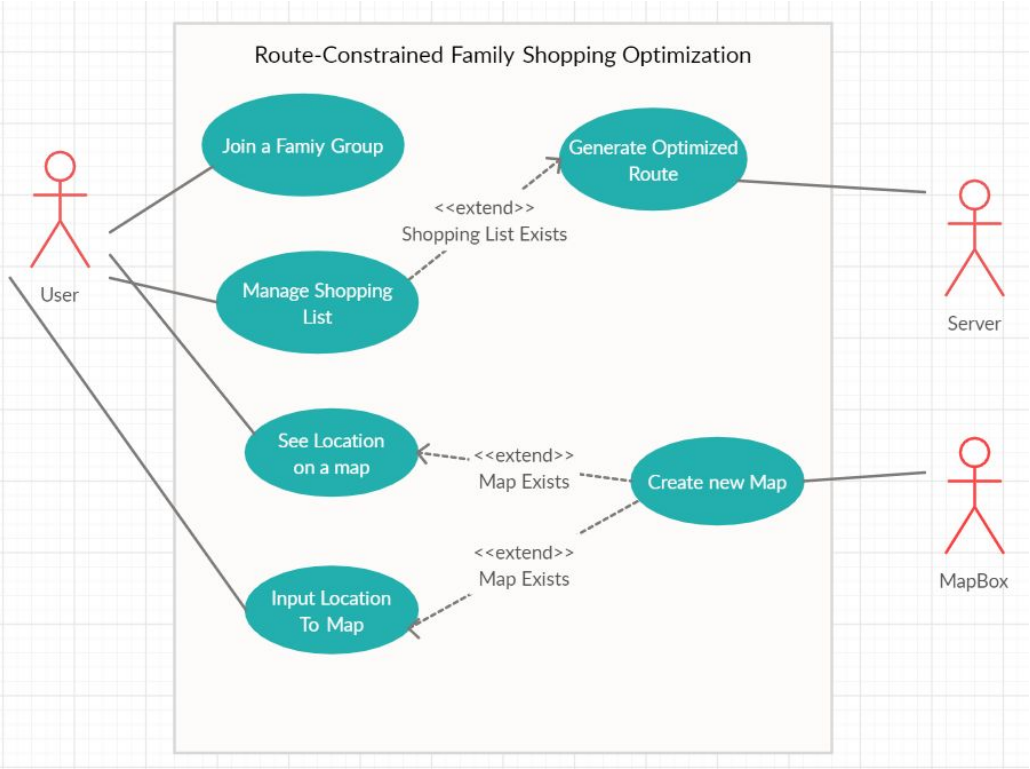


Figure 3: Use Case Diagram

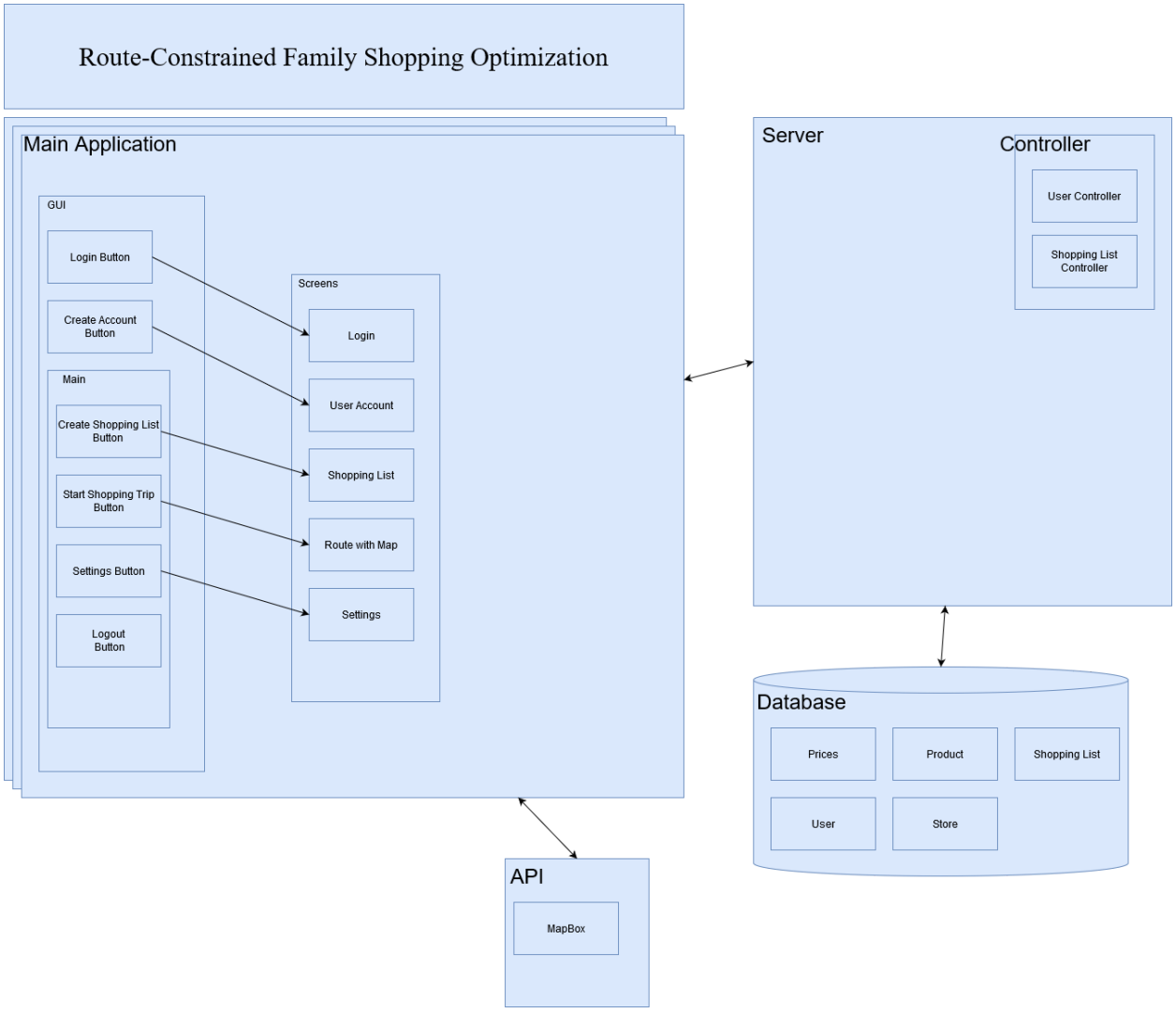


Figure 4 Block Diagram

1 INTRODUCTION

1.1 ACKNOWLEDGEMENT

The Route-Constrained Family Shopping Optimization team would like to thank both our advisors, Professor Goce Trajcevski and Ashfaq Khokhar for meeting with us on a weekly basis and helping guide us through the design process of this project. We would also like to thank the Iowa State University College of Engineering for giving our team access to professional guidance, resources and experts.

1.2 PROBLEM AND PROJECT STATEMENT

The goal of this project is to design and implement an application for both mobile and desktop that would help families coordinate and optimize their shopping routes. Families will also be able to add constraints to further customize their shopping route by including things like the family's starting location, the prices of the items, the distance from the store, and others. The application will then generate the optimal shopping route for that family.

The problem will involve constructing algorithms to determine the optimal shopping paths, getting data for item prices and store locations, and the constraints. The solution for both mobile and desktop will be the same. In the application, users can join a group with their family members so everyone can see the starting locations of each family member in the group as well as the family's current shopping list. To make the route, the application will check the family's shopping list to find items within the family's maximum distance they are willing to travel, and pick the store that has the items with the best price, thus optimizing spending. Through a map system the route would be shown and take into account the distance constraint. With all of this the application will output an optimized route for all family members.

1.3 OPERATIONAL ENVIRONMENT

Since this project will have a mobile and desktop application, the physical environment will depend on which platform the user is currently accessing. On the cyber side, we would use online systems and local systems to create the application and keep it maintained.

1.4 REQUIREMENTS

The Route Constrained Family Shopping Optimization project will have constraints to optimize the user experience. The constraints will limit the scope of the problem to make the algorithms more efficient. Additionally, there will be functional and nonfunctional requirements that our project aims to meet in order to fulfill user needs.

Constraints

- Radius of the map of stores and locations
- The time it takes to travel to different stores
- Starting the trip from home vs. varying locations
- Start time of the trip

Functional Requirements

- Store location accuracy
- Outputting the closest store with desired items with respect to distance/time to travel
- Output fastest travel time to any given store at desired start time

Nonfunctional Requirements

- Routes must generate in real time
- SQL Data must be in real time
- Application must be intuitive and easy to read

1.5 INTENDED USERS AND USES

The main user for this application is a family who wants to minimize their distance traveled and money spent while shopping.The final result is a route that is optimized for a single family member on spending, and the constraint of distance, for efficient shopping. Figure 3 shows a use case diagram showcasing the use cases of the project.

Users

- Individuals
- Family members

Uses

- Join a group with family members
- See shopping list of user and family members
- Add or remove items from shopping list
- See location on a map
- Input location to the map
- Show their optimized route

1.6 ASSUMPTIONS AND LIMITATIONS

The assumptions and limitations will assist in the decision making process for this project. Assumptions help create a more defined idea for the intended user, as well as also defining the limitations of this project.

Assumptions:

- The customer must create an account to be routed to stores
- The customer has a phone or PC to access this application
- The customer has access to an internet connection.
- Maximum number of users is unknown however the project can be scaled to accommodate more users.

Limitations:

- This application will only work inside of the United States
- Routes and Maps will be generated using MapBoxAPI
- Loading times will be determined by the quality of the users internet connection.

1.7 EXPECTED END PRODUCT AND DELIVERABLES

The final expected end product for this project will be made available at the end of the second semester. The Gantt Chart is available in section 2.1 Task Decomposition.

The deliverables for the first semester will be as follows:

1. **September 10th:** Identify data models and data sources to be used.
    - This deliverable will include the our main data structure to be used in the database. It will also include populating our database from store APIs.
2. **October 10th:** Finalize the plan for the different variants of the problem/solution.
    - This deliverable will include identifying and enumeration of the different problem cases. Then developing the solutions to each of these variants.
3. **October 25th:** Finalize the selection of development platforms and provide architecture design.
    - This deliverable will include a final decision of our development platform choices.
4. **November 10th:** Finalize the algorithmic solutions; device use-cases and test-cases; develop test-plans.
    - This deliverable will include a decision on algorithmic decisions for distance and time constraints. It will also include our use-cases, and test-cases.
5. **November 20th:** Finalize and submit the design document; prepare presentation.
    - This deliverable will include a completed design document, as well as a completed presentation for the first semester deliverables.

The second semester of the project will begin on January 25th. This semester will be focused on development of the application as opposed to first semester's planning stages.

The deliverables for the second semester will be as follows:

1. **February 1st:** Revise the design; assign development roles.
    - This deliverable will include revisions to the project to refine the design. Development roles will be assigned for each individual.
2. **February 22nd:** Complete unit testing; begin integration testing.
    - This deliverable will include the end of unit testing and integration testing will begin.
3. **March 10th:** Provide alpha-version and an implemented scenario.
    - This deliverable will include the first build of the project and be able to do the most basic scenario that was created.

4. **March 20th:** Finalize the revisions; release beta-version; run another set of end-user testing of functionalities.
    - This deliverable will include more revisions that will be final with a second build of the project. More testing will also be done.
5. **April 5th:** Finalize the user-manual, prepare for public release.
    - Not yet defined.
6. **April 15th:** Deploy the final version at Github; start the final report and presentation preparation.
    - This deliverable will include the final build of the project and the beginning of the final report/presentation.
7. **April 30th:** Final presentation and demo; submit final report.
    - This deliverable will include the final presentation, a demo of the working product, and the final report for the project.

2. SPECIFICATION AND ANALYSIS

2.1 TASK DECOMPOSITION

In this section, we break down each of the stages of the development process for the project. The Gantt chart below illustrates how long each stage should take, and when it will be started upon. The sections are explained more in detail below as well. See figure for diagram.

1. Identify Components
    a. In this stage, we would be talking with our client and start determining preliminary design. This would also include finding all the requirements and a general timeline when things should be done.
2. Research
    a. For this stage we would start researching components that would be a part of the solution. This would include researching getting item data from the stores, getting the road network information from an area, and the distance from stores to stores. Development environments that would work well with the ending product.
3. Develop Basic Design Plan
    a. Make a dynamic timeline
    b. Define a concrete solution.
    c. Cost and risk analysis.
    d. Determine end users and preliminary look of end product.
    e. Discuss and revise preliminary design with clients.

4. Make Use-Cases and In-Depth Design
    a. Make scenarios for each step of design.
    b. Build design for each component of our solution.
5. Finalize Design
    a. Set in stone the design of each component.
    b. Create a component diagram that shows how each component is connected to each other.
6. Finalize Basic Function
    a. Get a basic application working on either Android or desktop
    b. Be able to interact with the application with buttons or text boxes.
    c. Iterate through each scenario and build the required component functionality for each.
7. Review Design
    a. After the winter break review the design in order to refresh our minds.
    b. Add any details that we might have missed.
8. Begin Implementation
    a. Work with the basic functioning application.
    b. Make databases for the data that is going to be needed.
    c. Connect those pieces of data to be shown on the screen.
    d. Begin working out the algorithms for the route generation.
9. Complete Implementation with Testing
    a. Complete algorithms with simple inputs.
    b. Use the data being pulled in to interact with the algorithms.
    c. Design a better looking and functioning UI.
    d. Start testing the application for integration and unit testing.
10. Demos and Bug Fixes
    a. Get a working product that can be demoed.
    b. Fix any bugs that appear while testing and demoing the product.

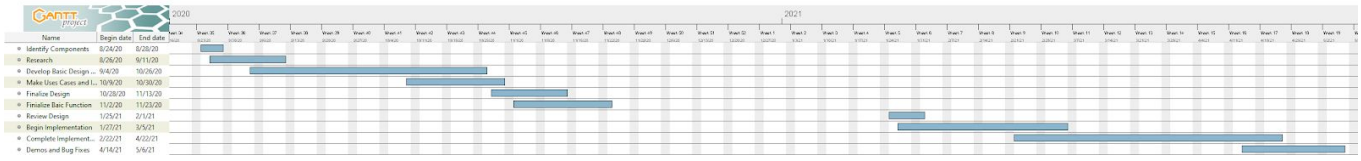
2.2 RISKS AND RISK MANAGEMENT/MITIGATION

Task/Component	Risk	Risk Probability	Risk Mitigation
Identify Components	Component is not identified or a component is incorrectly identified. Not all requirements are gathered	0.7	Meet with Client again to firmly establish requirements and components
Research	Information found is false or Information not available	0.6	Compare research results with multiple credible sources.
Develop Basic Design Plan	Preliminary design is not what the client asked for.	0.6	Meet with Client again to firmly establish requirements and components
Making Use Cases and in Depth Design	Scenarios do not correctly reflect how the end user will use the application	0.4	None
Finalize Design	Component Diagram incorrectly shows how each component is connected to one another	0.7	Meet with team and remake component diagram to make sure that each component is correctly connected
Finalize Basic Function	Scenarios were incorrectly formed on false requirements	0.2	None
Review Design	Design Document is missing information	0.8	Add any details missed in the design
Begin Implementation	Databases and Server cannot be accessed by the Web Application	0.5	None
Complete Implementation with Testing	Testing reveals bugs in code or implementation incomplete	0.6	Fix bugs and meet with whole team to fix issues regarding implementation
Demos and Bug Fixes	A bug is encountered that cannot be fixed	0.3	None

2.3 PROJECT PROPOSED MILESTONES,METRICS, AND EVALUATION CRITERIA

For the first semester our milestones are: To have a semi-completed design done halfway through the semester. The final milestone for this semester would be a completed design and design document. For the second semester our milestones will be the different levels of functionality for our project. Single user multiple stores, single family multiple stores, multiple starting times, travel_distance vs. travel_time, and Scalability. These milestones can be measured by their functionality and if they work. Another milestone will be the completion of the web application and mobile application which will be measured again by their completeness and functionality. One way to do this is to get test users and get feedback throughout the development process to get a continuous stream of feedback and evaluation criteria.

2.4 PROJECT TIMELINE/SCHEDULE



See Section 2.1 for a detailed description of each task associated in the Gantt chart.

2.5 PROJECT TRACKING PROCEDURES

Our team is planning on using Trello for project tracking and management and will be using GitHub for the file tracking and collaboration. We will also be using discord to conduct team meetings and communication which will also be used to keep track of meeting notes, assignment deadlines, and important information/announcements we want to have pinned.

2.6 PERSONNEL EFFORT REQUIREMENTS

Task	Effort Requirement (Person hours)
Identify Components	3
Research	12
Develop Basic Design Plan	30
Making Use Cases and in Depth Design	60
Finalize Design	36
Finalize Basic Function	30
Review Design	30
Begin Implementation	24
Complete Implementation with Testing	300
Demos and Bug Fixes	90
Total	615

2.7 OTHER RESOURCE REQUIREMENTS

There are no physical parts and materials needed to complete this project. The work will be completed on various coding platforms. These platforms are Android Studio with Koltin, React.js, and ASP.NET with c#.

2.8 FINANCIAL REQUIREMENTS

The MapBox API that will be utilized will cost $0 for every 50,000 requests per month. The team will most likely stay within this 50,000 request limit, so the API will have no cost to the team. Hosting a server for the web and mobile application will cost between $40-$70 a month. Publishing the application on the Android store is a $25 one time payment. In total, the total cost for this project will not exceed $120.

3. STATEMENT OF WORK

3.1 PREVIOUS WORK AND LITERATURE

There is much literature and previous work available for research to aid in the development and design of the project. After surveying multiple options through the research of this literature, the group has decided upon utilizing MapBox for a road network API. MapBox has developed an API for generating road networks and maps which can help the project generate shopping routes. The advantage of using MapBox is that it is free for the purpose of this project, and can generate an accurate map of an area. We also decided upon utilizing a modified version of Dijkstra's shortest path algorithm and the A star (A*) algorithm for calculating the shortest path from a starting location to a desired store. The advantages of using a modified Dijkstra's algorithm is that the algorithm does not have to traverse every edge in a graph, improving the efficiency of the algorithm. A disadvantage of using Dijkstra's algorithm is that it does not take into account negative edge values, which can result in a less efficient path.

Previous Work

- MapBox
- Dijkstra's Algorithm
- A* Algorithm
- Distance Indexing

Literature

- "Distance Indexing On Road Networks" by Haibo Hu, Dik Lun Lee, and Victor C. S. Lee

3.2 DESIGN THINKING

In Figure 1, the design process diagram developed by the team can be found. The main "define" aspect that has shaped the current design is "Utilize the client's suggestions to make a problem statement". The team has worked very closely with the client, meeting weekly to go over progress made and the expectations of what work needs to be done in days to follow. This has allowed the team to properly understand the expected requirements and make design decisions that best fit the project. For example, by meeting with our client consistently, the team was able to choose the proper technologies to develop the project.

The "define" stage was followed closely by the "ideate" stage, where the team has mainly focused on "brainstorming solutions to the problem statement". In this stage, the team has established the simplest variants of the solution, and worked our way up to the most complicated variant. This determines the timeline of later development, as well as the use cases for each variant.

3.3 PROPOSED DESIGN

To properly fit our design to the problem at hand, we must break down the problem into smaller ones. Once we have solutions to the smaller problems we can combine them together and have a fully finished design.

Routing Algorithms :

For our project we must find the most efficient path between a user and the stores they wish to visit. To solve this problem, a pathing algorithm is needed. The two pathing algorithms that we came up with to solve this problem were the A* algorithm for time dependent shortest path and Dijkstra's algorithm for shortest path. We chose to use A* as we needed the time factor of the algorithm. This will help us complete the following requirements : Outputting the closest store with desired items with respect to distance/time to travel | Output fastest travel time to any given store at desired start time | Routes must generate in real time | Routes must generate in real time

Storage of User and Store Data:

To properly store the information that we get from users and stores we must generate tables in our database. To do this efficiently we normalize the data before putting it into the database. All store information such as name and location is stored in one table. All product and price information is stored in another table. All User information such as name and address are stored in one table. All shopping list information is stored in another table. Normalizing the data allows us to update, delete, and add faster to the database. It also decreases the amount of repetitive data we have. This design will complete the requirements : SQL Data must be in real time | Store location accuracy

Creating a UI:

The UI would be made both on mobile, with an Android application, and web application, with React. For the Android application we will be using Kotlin and Android Studio to develop the Android app. This will allow us to use Google's most recent Android development tools. The UI will attempt to give a clean and easy to use interface for users. This will be achieved by creating pages with minimal information and buttons that will be clear and concise for users to avoid confusion. These same ideas will be used in the web application giving the users the information they need and not cluttering the screen with unnecessary buttons and information.

Web Scraping Store Data:

For the project we need to be able to get the item and the price from the stores in a certain distance to the database. We will use web scraping to access the stores in an area and grab the item that is on the shopping list from the UI. This would put that item and the price of it into the database in order to compare prices to help with the route optimization.

Hosting a Server:

For the project we need to host a server to contain all elements of the project solution. This server will include API, UI, and the MYSQL Database. It is important that each piece of the project is hosted in a place where the other components can access and communicate with it. If this were not the case then the project would not work. Hosting the server for our project is tied to all of the requirements discussed in earlier sections.

3.4 TECHNOLOGY CONSIDERATIONS

The technology to be utilized within the project has a variety of forms. The development of a web and mobile application will need different software programs for their development. Additionally, a database will need to be maintained, as well as an API for the map used to find routes. The following technologies are what have been determined to be the best fit for the development of the project.

- Android Studio
  - Mobile application
- React
  - Web application
- ASP.NET API
  - Grab data from the database
- MYSQL Database
- MapBox API
  - Getting the map of the area needed
  - Stores in the location
- Windows IIS Server
  - Server for the backend

3.5 DESIGN ANALYSIS

We have not implemented the design yet so we do not know if the design will work at the moment.

3.6 DEVELOPMENT PROCESS

The main development process for this project is a modified version of Agile. We are using Agile as it will push the project forward through test-driven development. By having team meetings every week the design process is streamlined and allows for a clear understanding of the design goals by every team member.

Version Control and Project Tracking:
For our project we plan to use Github as our version control system so we can all collaborate on the project synchronously and avoid having conflicts with each other. For project tracking and task management we will be using Trello to allow us to assign tasks to each other and track those tasks throughout the development process. Both of these tools are crucial for a team following the Agile development process.

3.7 DESIGN PLAN

The main design plan will be to use the five pillars of design thinking. By empathizing we communicate with our client and find out the requirements for the problem. When defining, we are establishing the requirements and constraints for the project moving forward. In ideation, we determined multiple variants to the problem and ways to solve all of them. Then by assigning roles to each of these problems/solutions we are able to start prototyping. Once prototyping begins, we will start development of the application to figure out what ideas worked and what we can change to improve the final product. We will then start unit testing and integration testing to start refining our project into a quality product.

4. TESTING

4.1 UNIT TESTING

Some of the software components we are planning on testing in isolation include: the algorithms, mobile app, web app, backend database storage, and the web scraping. The reason for this is that this will allow us to individually test each component before we start fitting all the pieces together for the final product. This will also allow us to focus on one area at a time and thoroughly test that aspect of the product. Doing it this way, we can see if something breaks and be able to isolate the problem quickly and efficiently. Some methods for achieving these unit tests will be to use a manually created database and compare with the one that gets created as data is sent, compare a manually created data set for the web scraping to check that it is covering all cases, for mobile we can give it to people to test and create various unit tests, for the algorithms we will be able to give it sample data and have it run through the code to see what the result is and compare it to the expected results.

4.2 INTERFACE TESTING

Interface testing will be done by testing a combination of path finding algorithms (A*), database data, and user interface together. The interfaces will be as defined below:

- Algorithms and database
    - This will be done by testing the algorithms and how they access and interact with the database.
- Algorithms and user interface
    - This will test whether the algorithms affect the user interface like showing the correct route on the map.
- Database and user interface
    - This will test how the user interface takes data from the database and make sure it is displayed correctly.
- Algorithms, database, and user interface
    - Testing everything together to see if the project is working all together.

By testing the units in groups of two before testing all of the units together it will allow quick identification of problem areas in the units.

4.3 ACCEPTANCE TESTING

We will be able to demonstrate that the design requirements are being met by developing use cases to test for functional requirements. The use cases will cover every possible input by a user to ensure that there are no bugs in any section of the code. The application will be run with these particular use cases many times to ensure they are correct.

To demonstrate that the design fulfills the nonfunctional requirements, the project will be demonstrated to both the client as well as other team members to ensure that the application functions satisfactorily. The client's comments and critiques will be used to revise and further develop the project. The application will then be demonstrated again until the client is satisfied with the implementation.

4.4 RESULTS

Since this semester we are not doing any testing on the actual application, we have not done any testing at the moment.

5. IMPLEMENTATION

6. CLOSING MATERIAL

6.1 CONCLUSION

6.2 REFERENCES

6.3 APPENDICES