CS3354 Software Engineering
Final Project Deliverable 1

UTD Campus Pathfinding Application

Members of Group 4
Abdullah Akbar
William Bumpass
Jenain Khan
Vishakh Nair
Benjamin Rubarts
Michael Villordon
Adam Wajahat

1. Final Project Draft Description:

Optimization of paths from one point to another can be tricky, especially on the UTD campus where navigation is often complicated, and the shortest path often difficult to find. Thus, in an effort to make campus navigation more friendly, we propose an application that would allow for the quickest route across campus to be decided with just a start and end point. Paths - intended to be taken by those on foot - would be similar to other pathfinding applications such as Google Maps, though limited only to the UTD Campus and surrounding amenities (i.e. testing center).

- Would be a useful tool for campus guests or those adjusting to the changing campus
- Could be expanded upon to larger areas around campus to include distant apartments
- Setting path navigation ahead of time could allow for pre-planned paths to display at certain times
- Could be integrated into the UTD Mobile app, allowing for more concise school tool usage
- Allows for color coding or categorization of recurring routes
- Sharing of routes and directions through the app and other mediums
- Accommodates routes affected by restrictions (disability, shutdown areas, construction, etc.)
- Creation of agendas for daily routes utilizing calendar application integration (such as Google Calendar)
- User path preferences may be set (prioritize inside walkways vs. outside walkways, etc.)

Proposal Feedback:

Good proposal and fair distribution of tasks to group members.

In your final project report (deliverable 2) please make sure to include the following:

- A thorough search to find similar application implementations. Please cite these work using IEEE citation format provided on Final Project Specifications document.

- Please make sure to differentiate your design from existing similar applications by including extra features into it.

- Please make sure to explicitly specify those differences by comparing your design with those existing similar applications.

Thank you

Feedback Response:

- We will attempt to do a thorough search on the implementations of similar applications. Our main target for observation would be Google Maps - given its popularity and similarity in nature - though it will not be the only case. We will also analyze the Starship delivery app, which sets routes for the delivery robots according to a map of the campus. As we address the work of these implementations, we will cite them using the IEEE citation format.
- Our project will be different from other similar applications through the following features:
    - In order to make our application more student-oriented, we will allow users to input routes between specific classrooms instead of just buildings. Including inter- and intra-building maps allows for users to find harder-to-access classrooms.
    - To allow for more user-friendly reporting, people can submit various blockages and obstacles found along a path. This will allow others to note the blocks on their paths. Few other applications allow for the direct and intentional submitting of accidents and other obstacles.
    - We will also integrate student life with the application; users can submit events happening on campus and they will appear on the campus map. Clicking on the events will show the closest routes to get to them.
    - Our project would be integrated into another application (UTD Mobile) so the database could autoupdate with information about events and other possible obstacles on paths.
- In addition to the distinct features, we will mention which features also appear in the application implementations we are comparing, as well as which features we are leaving out in our application.

2. Github

Link To Project Repository: https://github.com/vishakh15n/3354-group4

3. Task delegation

- Address feedback to the project proposal: William Bumpass, Abdullah Akbar, Benjamin Rubarts
- Creating Github repository (3354-group4) and adding group members and TA to it: Vishkah Nair
- First commit to the repository (README file): Jenain Khan
- Second commit to the repository ("project_scope" file): Adam Wajahat
- Commit to the repository (Project Proposal): Adam Wajahat
- Commit to the repository (Deliverable 1): Michael Villordon
- Commit to the repository (Deliverable 2): Benjamin Rubarts
- Add link to repository: Michael Villordon
- Software process model evaluation: Jenain Khan, William Bumpass, Vishakh Nair, Benjamin Rubarts
- Requirement evaluations:
    - Functional: William Bumpass
    - Non-functional: William Bumpass, Abdullah Akbar
- Use Case diagram(s): Adam Wajahat, William Bumpass
- Sequence diagrams: Michael Villordon, William Bumpass
- Class diagram: Benjamin Rubarts, Michael Villordon
- Architectural design (choosing and "implementation"): Abdullah Akbar, Michael Villordon, Jenain Khan
- Project scheduling: Benjamin Rubarts, Adam Wajahat
- Cost methods details and price of project: Adam Wajahat
- Costs of hardware: Vishakh Nair
- Costs of software: Jenain Khan
- Costs of personnel: Michael Villordon
- Effort "evaluations": Vishakh Nair
- Test Plan for Software: Adam Wajahat, Vishakh Nair
- Comparison to similar designs: Abdullah Akbar, William Bumpass
- Conclusion: Michael Villordon
- References: Everyone
- Presentation Creation: Everyone
- Implemented Code (whole code or just JUnit test code): Everyone

4. Software Process Model Evaluation:

We are using the V-model, a variation of the waterfall model, in our project. We intend not to cut corners and simply release multiple versions as in the prototype model, add separate

features at different times and find ways to bring them together as in the incremental process model, or release more progressive prototypes as in the spiral model. We intend to plan everything out one step at a time and release a complete application, but we also want the flexibility of not having to start completely over when we detect a mistake.

The V-model Software process is based upon the concept of verification and validation where testing occurs at each level of development. The next phase can only start once the previous phases are complete and correct making the V-model simple, efficient, and best suited for smaller projects, such as our Pathfinding Application. Since we have clearly defined requirements for our Pathfinding Application as mentioned in our description the process of the V-model of going from broad to refined matches our application well, as we can go from requirement analysis to design phases. We also found it to be advantageous how the V-model allows for errors to be caught in earlier phases.

i. Pros:
    1. Fully documents code
    2. Validates and reviews at all levels
    3. Good for smaller projects

ii. Cons:
    1. Requires long specifications
    2. A large amount of time spent planning
    3. No iterations
    4. Intensive testing after each step
    5. Poor with larger projects

5. Requirement Evaluations:

Functional:

i. The user shall be able to request paths for a desired route
ii. The webserver/database shall be able to handle multiple requests in parallel
iii. The webserver/database shall be able to gather information on Comet Cab routes
iv. The webserver/database shall be updated with information on blockages through construction and accidents
v. Users shall be able to store their recent route requests for review/reuse later.
vi. Routes should be generated within 10 seconds of query at maximum
vii. User preferences for routes will be selected through a list format, allowing for multiple selections over different categories (disabilities, inside/outside)

Nonfunctional:

I. Product Requirements
    1. Usability Requirements:

a. The app interface must be easy enough for a regular student or faculty member to use for general purposes
b. The interaction with the map display should be similar to other apps that use map routes, like Google Maps or Uber.

2. Efficiency Requirements
    a. Performance Requirements:
        i. The mobile app should be able to run on both iOS and Android devices, provided they have up-to-date operating systems and are from after 2010
    b. Space Requirements:
        i. Google Maps takes up a little less than 400 Mb of storage, so our app should take up no more than half a gigabyte of storage.
3. Dependability Requirements:
    a. As our app is nonessential, a continuous downtime of at most 24 hours is permissible. We should do monthly maintenance checks on the servers to make sure no errors cause the downtime to exceed this limit. Furthermore, the webserver/database should be able to handle 10000 concurrent users without affecting its performance.
4. Security Requirements:
    a. In order to enhance ease-of-access and ensure only enrolled students/faculty are setting up events, we would want to integrate our app with the UTD NetID system. This means that our backend authentication system must meet the security requirements the Office of Information Technology requires in applications integrated with UTD NetID.
    b. Communication between the mobile application and the backend servers should be encrypted to ensure the privacy of the user's input.

II. Organizational Requirements:

5. Environmental Requirements:
    a. The frontend application should work fundamentally the same on iOS and Android so that moving from one OS to another does not cause any inconveniences for the users.
6. Operational Requirements:
    a. The routing feature of the software should allow people on campus to select two locations on campus and have the software calculate and display the optimal path between both points.
    b. The location search bar should also save past locations searched so the user can find the current route to them with just one tap (like Google Maps saves previous

queries). Such locations should be saved locally on the user device to preserve privacy.

c. The event organizing feature should allow users to add events and their locations, and then specify who can view the event details (public or limited to specific users). Then the respective users should see the event pop up on the campus map, along with its details.

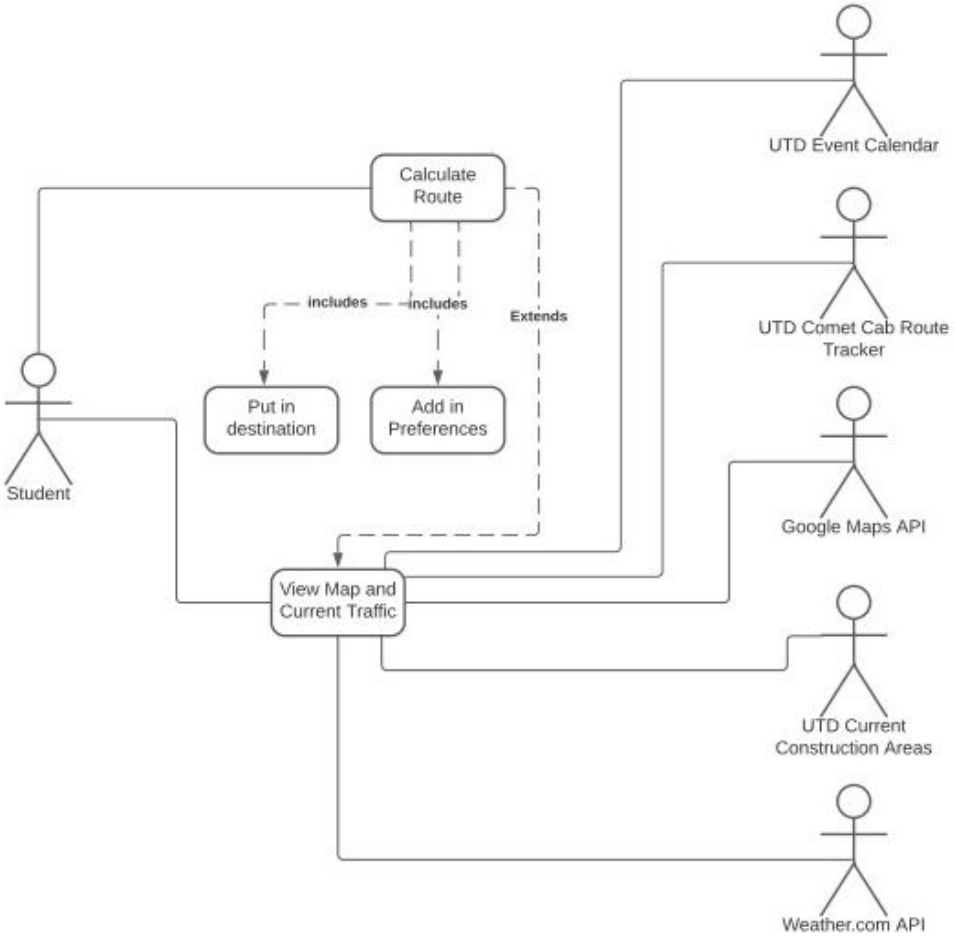
3. Development Requirements:
    a. The software should include a backend server system and a frontend mobile app system that should run on both iOS and Android.

III. External Requirements

1. Regulatory Requirements:
    a. Privacy of information, the export of restricted technologies, intellectual property rights, etc. with regards to the software should be audited by the University of Texas at Dallas.
    b. Any updates to the software's privacy policy or any changes made with regards to how user data is handled will need to be approved first by the appropriate boards of the University of Texas at Dallas.
2. Ethical Requirements:
    a. User privacy is a top priority and NetID's will only be used to ensure any user organizing a campus event is actually a UTD personnel.
    b. An "About" page shall provide general information regarding the implementation of the app, as well as detailed information of the processes which occur through the app.
3. Legislative Requirements
    a. Accounting Requirements:
        i. All funding regarding the app should be agreed on beforehand between us, the app developers, and the University of Texas at Dallas. Since no money is involved in the actual app, the only accounting to consider is the money required to maintain the necessary app servers.
    b. Safety/Security Requirements:
        i. Users should be able to report any misconduct that happens at an event displayed on the app; such reports should be immediately forward to the authorities.

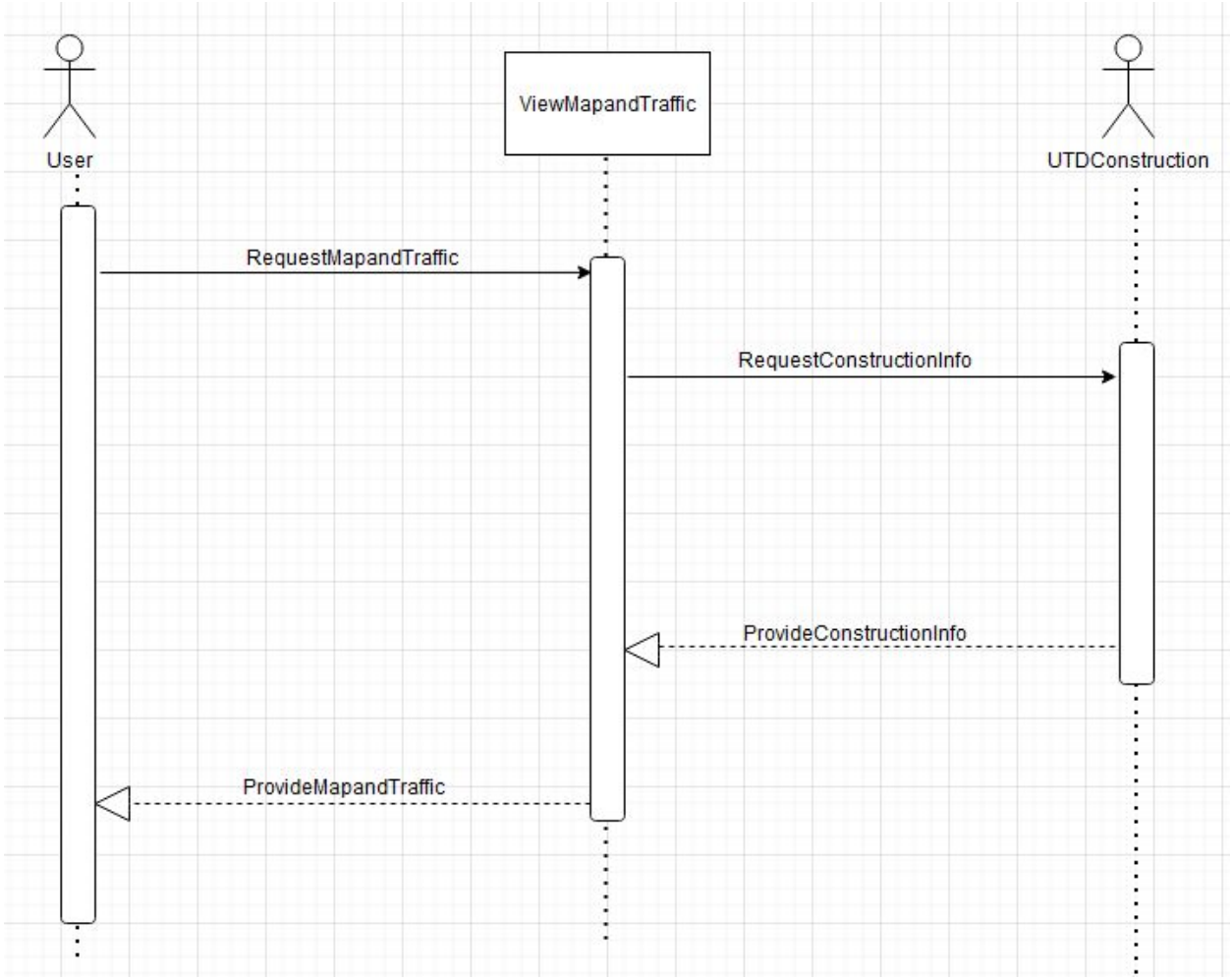
ii. A no-leniency policy will be put in place so that organizers of illicit events on the app will be removed after one offense.

6. Use case diagram:

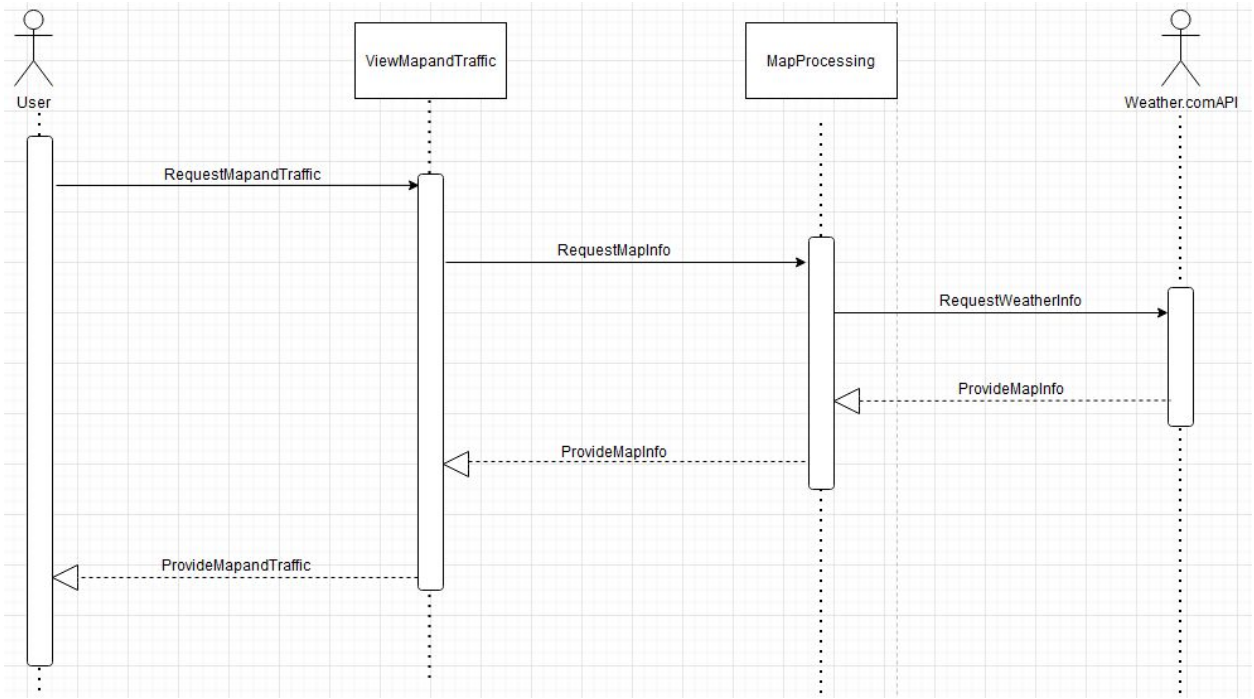


7. Sequence diagrams:

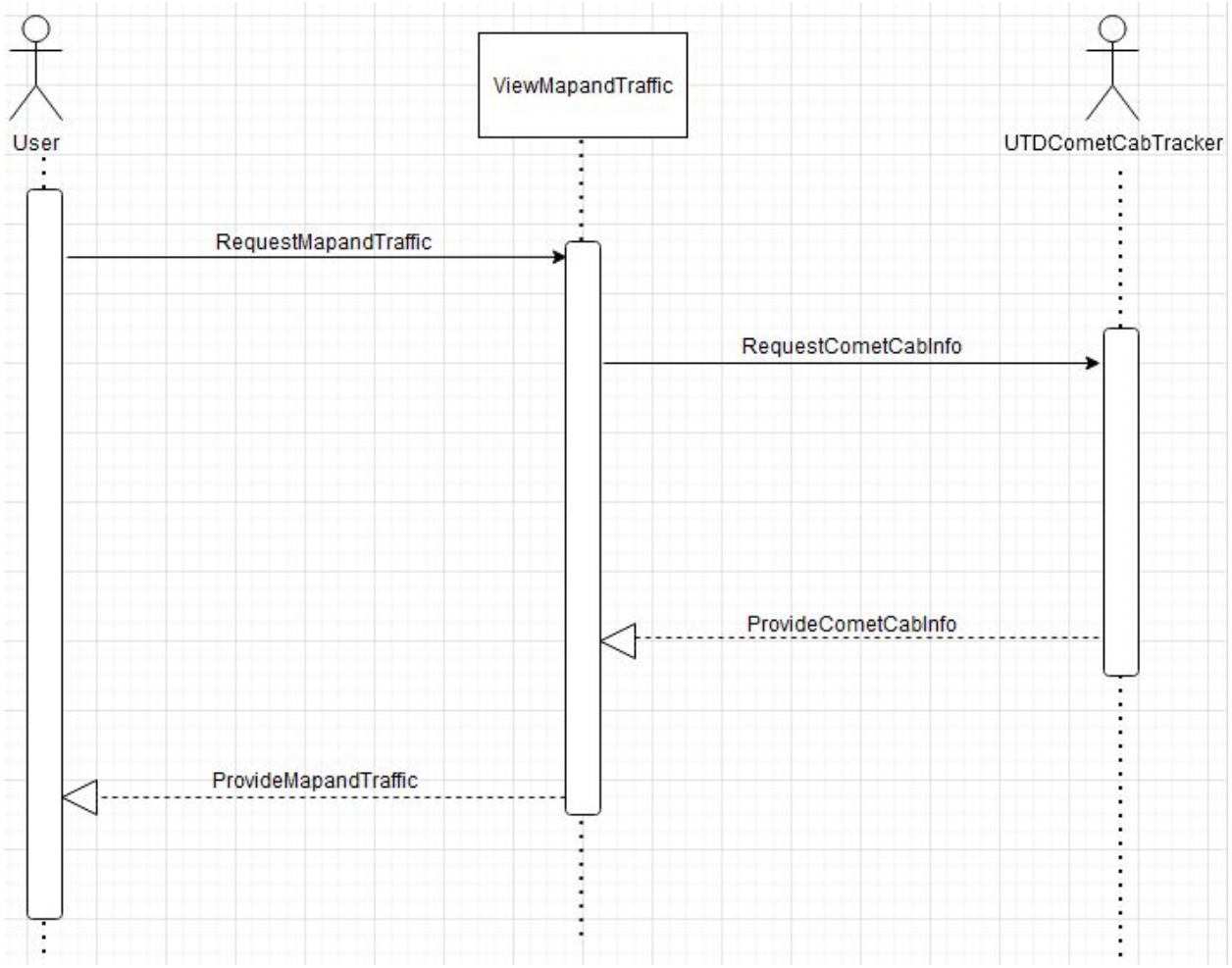
(for View Map and Current Traffic use case in connection with UTDConstruction)



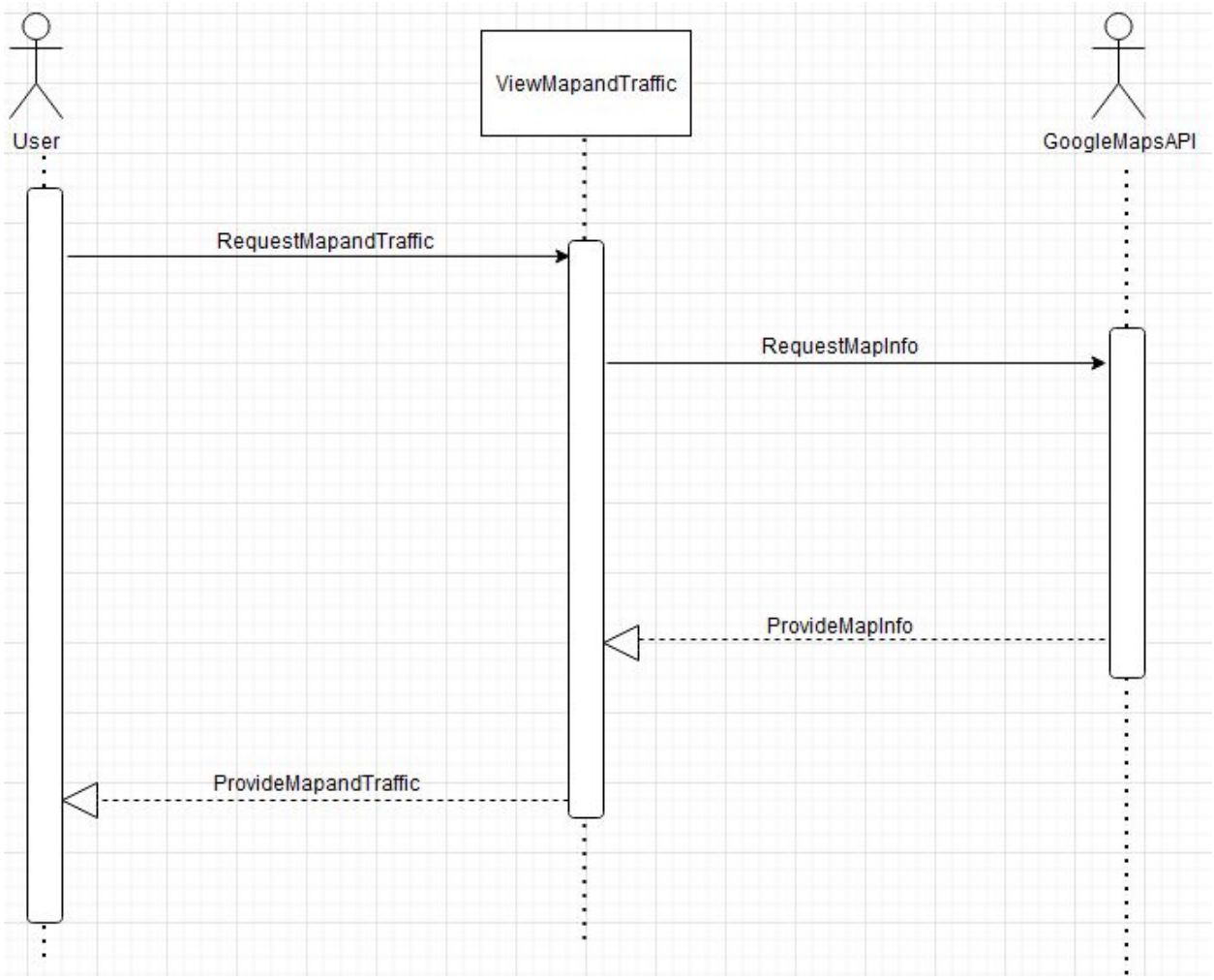
(for View Map and Current Traffic use case in connection with the Weather.comAPI)



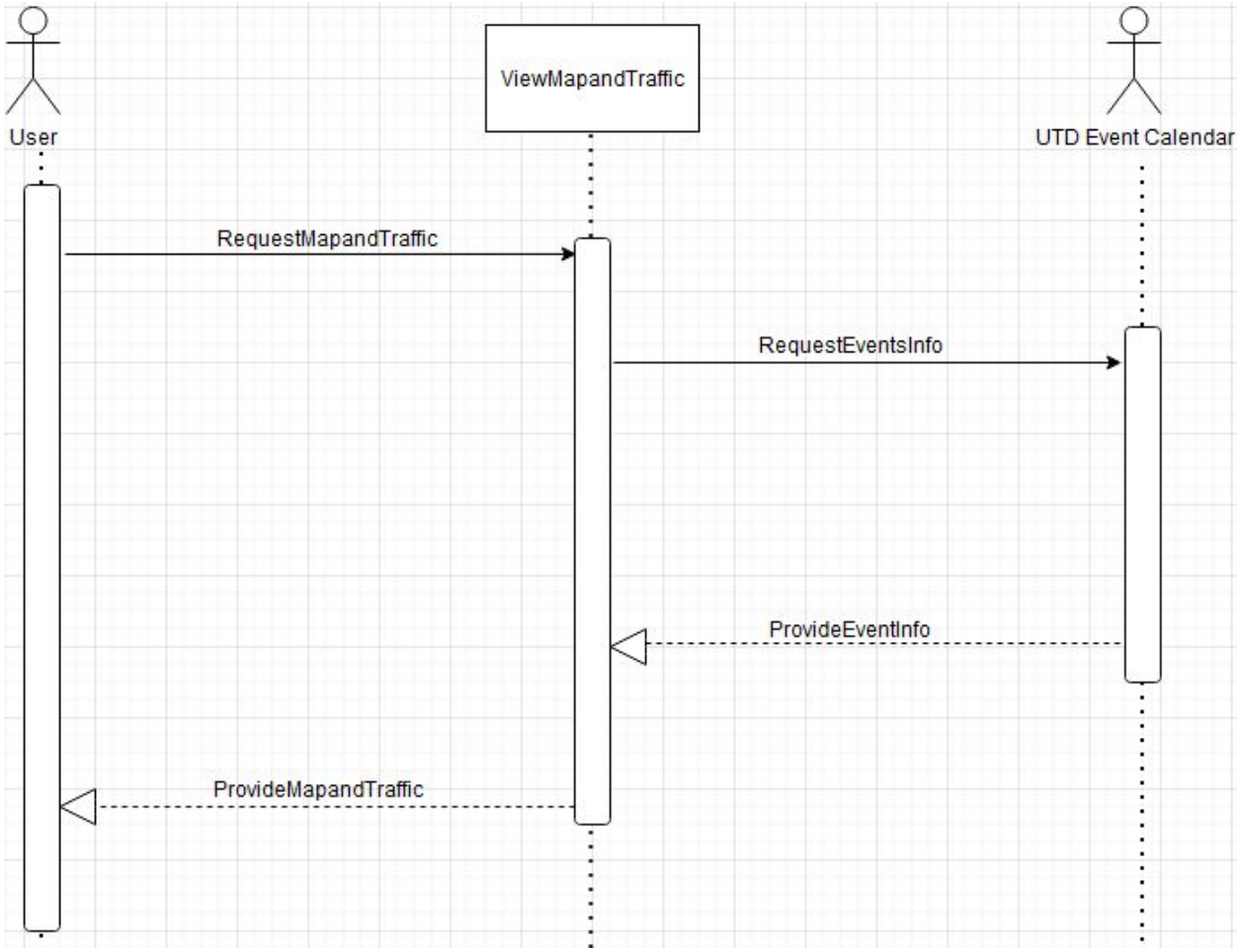
(for View Map and Current Traffic use case in connection with the UTDCometCabTracker)



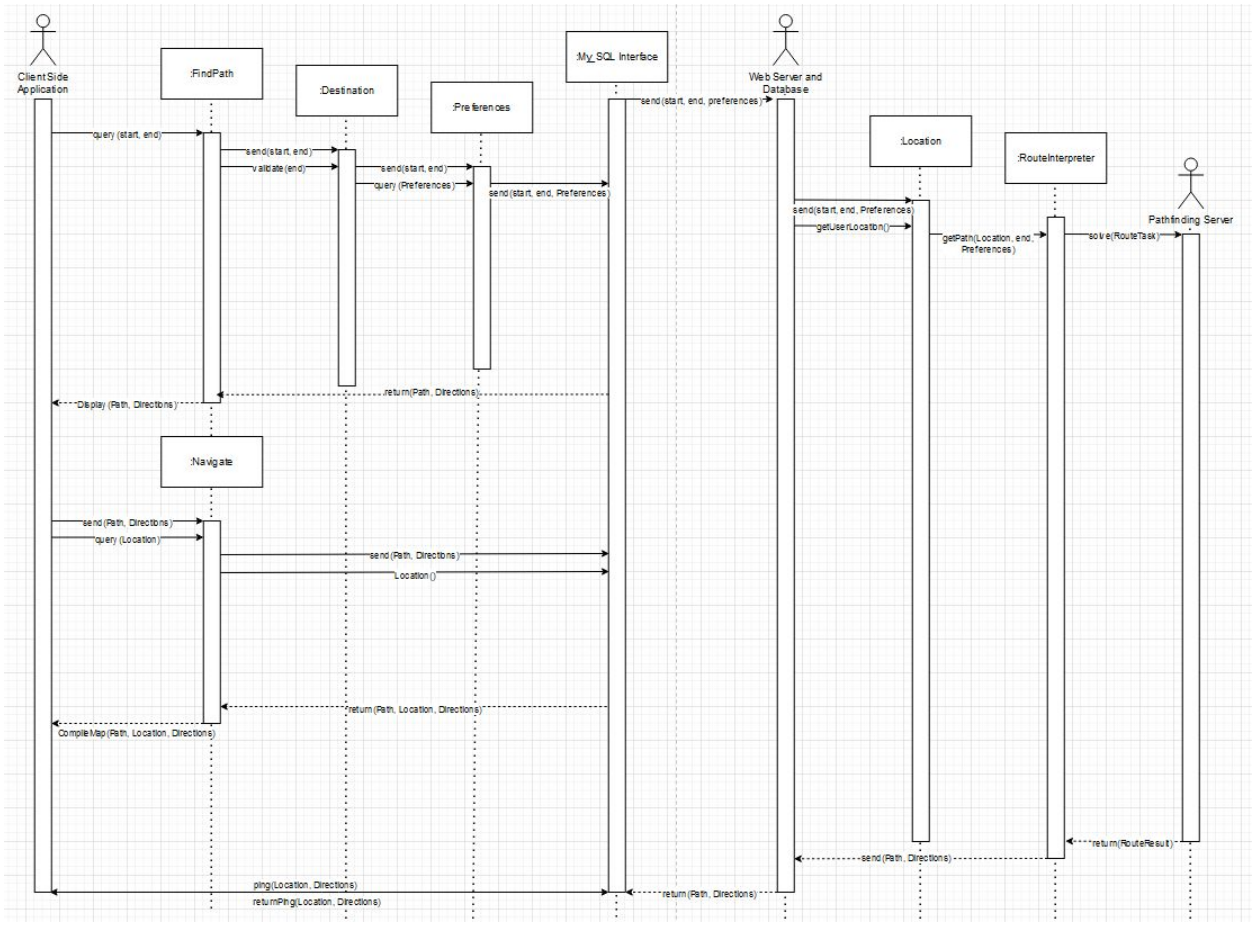
(for View Map and Current Traffic use case in connection with the GoogleMapsAPI)



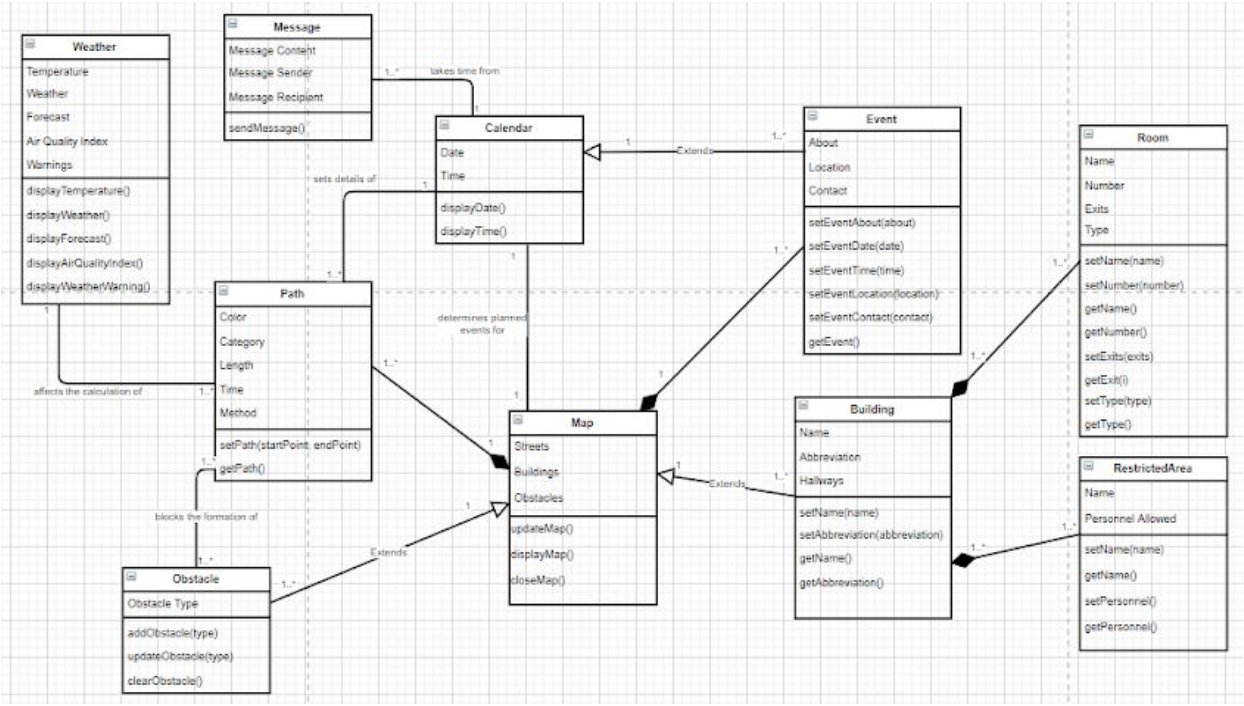
(for View Map and Current Traffic use case in connection with the UTD Event Calendar)



(for the Calculate Route use case; features included use cases Put in Destination and Add in Preferences)



8. Class Diagram:



9. Architectural Design:
   Model-View-Controller Pattern (MVC)