**Final Project Report**

1. Team Member Contributions:  
     
   **Misty Penner** **(Time Spent: 15 hours):** Fixed frontend bugs involving app rotation crashing the device. Added the last feature, Update Task, to the front-end and verified that it worked via writing tests. Designed an app icon by hand in gimp. Helped write and update documentation based upon changes made.

**Patrick Chen (Time Spent: 25 hr):**

* Fix bugs below, organize backend/frontend code in the repository, and make all server API endpoint controller+route.

Bug fix 1: Fixed [notification spam (made notification persistent)](https://github.com/bvpenner/CPEN321Project/commit/c5b0095948bd68b40362ffec1feae6dea0165d1d)

Bug fix 2: D[raw optimal route on map; map recreation error fix](https://github.com/bvpenner/CPEN321Project/commit/5a0fe722cd5433a309bee22b66709e164929480f)

Bug fix 3: Finalize incomplete update task [api](https://github.com/bvpenner/CPEN321Project/commit/744a63d9e97a0a72de4125f80ba93d5ebb089043)

Bug fix 4: Changed communication into https

Bug fix 5: Modified frontend to request into cert signed Https; Change backend using HTTPS

* Refactored entire repository (back+front+test) and did code clean up:

Clean 1: Move all backend code into folder /backend, resolve dependecies

Clean 2: Move all frontend Android app into folder /frontend, resolve dependencies

Clean 3: Make the server api into router + controller structure

* Document fixing, added previously missed information into the document and rejustified magical number sin non-functional requirments.
* Record videos for part of the backend code and backend test (usecase1: manage task, usercase2: task geofencing), and work on the presentation slides

**Jingyang Cui (Time spent: 17h)**

Debugged find optimal route backend end code & added addition test cases for coverage. Record half of video #3 & #4, a portion of video #1. Updated and worked on documentations for complexity and etc..

**Amaan Siddiqi(Time spent: 17h)**

Improved UI, collated and edit Video 1 and also the frontend video.

1. Github Repository Link and Commit SHA of Final Release:  
   Github Repository Link: <https://github.com/bvpenner/CPEN321Project>  
   Commit SHA of Final Release: 185b7f7e589cffec57378c5781586af48f021e6b
2. Location of Tests:  
   Back-end Test Locations:

Unmocked test

backend\_test/backend\_nomock.test.ts

Mocked test

backend\_test/backend\_mock.test.ts

Non-functional test

backend\_test/nonfunctional.test.ts

Front-end Test Locations:  
 Base Testing Structure:  
 app/app/src/androidTest/java/com/example/cpen321app/BaseUITesting.kt  
 Manage Tasks:  
 app/app/src/androidTest/java/com/example/cpen321app/ManageTaskTesting.kt  
 Find Optimal Route:  
 app/app/src/androidTest/java/com/example/cpen321app/RouteTesting.kt  
 Task Geofencing:  
 app/app/src/androidTest/java/com/example/cpen321app/GeofencingTest.kt

1. Manufacturer and Model of Physical Device:  
   Device Manufacturer: Samsung  
   Device Model: Galaxy S7
2. IP Address of Back-End Server and Domain Name:  
   IP Address of Back-End Server: 13.216.143.65:3000  
   Domain Name: Not applicable.
3. All necessary details to run the app:  
   A Google account is required to log in.
4. Use cases that implement required complexity and how:

The find optimal route use case is our complexity. The challenge is we need to find the optimal route sequence for multiple tasks while considering a number of things, such as a task’s start time, latest reaching time, estimated duration, and location. It’s a NP-hard problem, and for k tasks, it generates k factorio number of possibilities.

In our implementation, we first use Google Map Distance Matrix API to construct a graph of pairwise travel time between any task and user locations. Each node value denotes the a task’s estimated duration, and each edge denotes the travel time.

Starting from current user location & user time, the algorithm iteratively explore all possible path sequences, by considering adding one task at a time. If at any time step some tasks cannot be reached on time anymore, that path will be stopped early to reduce computation. At the end, the algorithm will produce a list of viable paths with its respective total travel time. It will choose the sequence with lowest time and send it back to the user.

Notably, our algorithm also considers the case if you arrive earlier than a task could be started. In this scenario, the wait time would also be included in the total travel time calculation.

1. Technical extras:

* Instead of making many API calls to get all the pairwise travel time information between any two locations, we found the best API (Google Distance Matrix) and squeeze everything into a single API request. This significantly reduce the cost and reduce the wait time.

1. Project Limitations:
   1. The app can only handle route plans of tasks that occur on the same day.
   2. The app only has Google authentication for login, which makes the app only accessible to Google users.
   3. The two main use cases rely heavily on Google Maps APIs. If the services are down, there is no walk-around.
2. Reflections on Generative AI Use:
   1. AI Technologies Used During the Course:  
      - Copilot  
      - ChatGPT
   2. For which phases of the overall Software Engineering process and which concrete tasks in these phases was Generative AI the most helpful for and why?

* Implement the basic structure of the server, initiate testing, and install dependency. Help with documentation, help organize documents, and fix grammar.
  1. For which phases of the overall Software Engineering process and which concrete tasks in these phases was Generative AI the least helpful for and why?
* Helpful during the requirements elicitation and system architecture design phases. It’s helpful to ask them about the feasibility of some ideas and help when consolidating ideas.
  1. Anything else to share about AI’s impact on your Software Engineering tasks?
* Don’t overrely on generated content, and always ned to confirm the correctness of this content.