Final Report

This document includes:

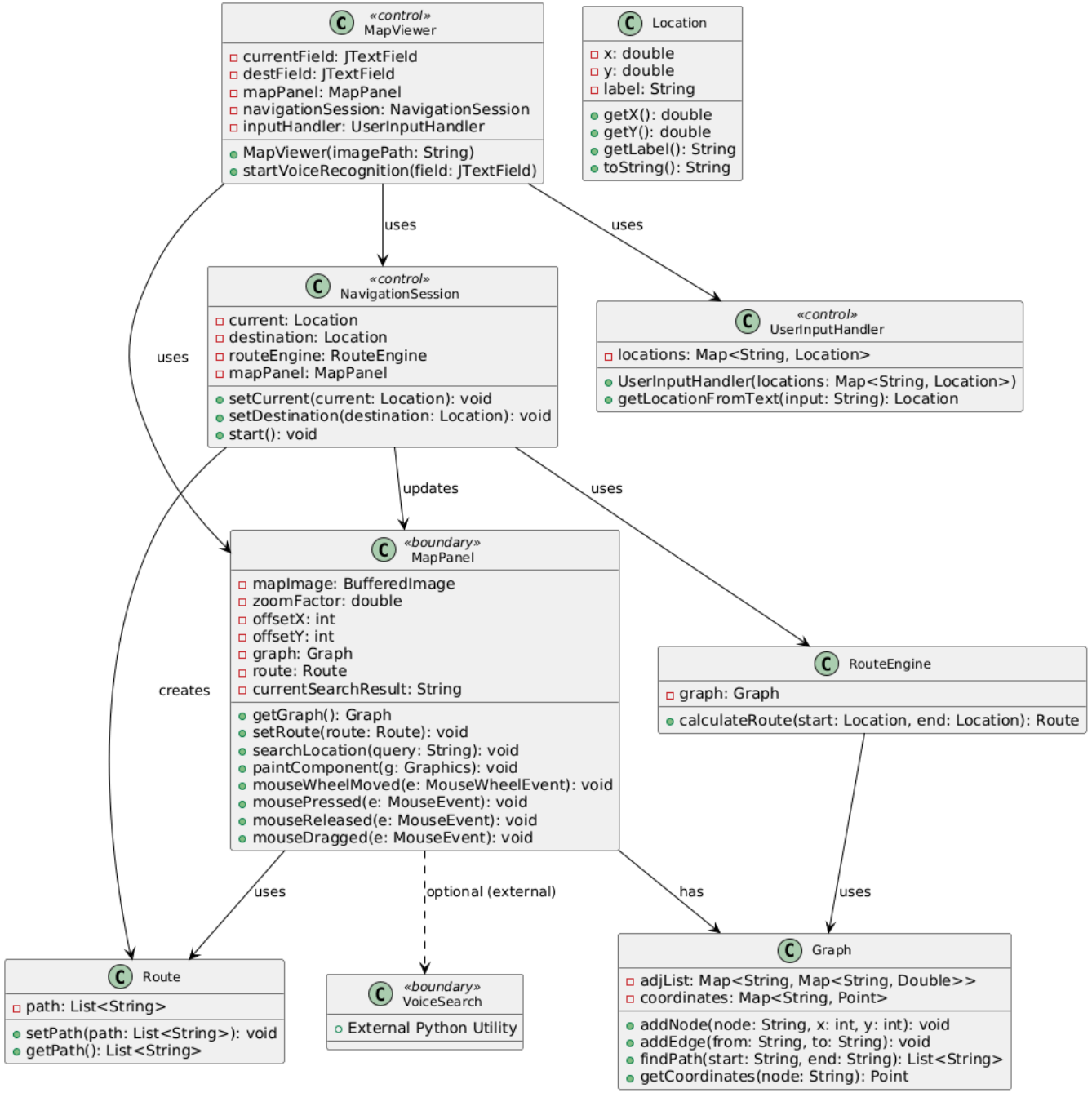
- A **UML Diagram** to visualize the code/design architecture.
- A thorough **Product Backlog** with user stories, priorities, and efforts.
- A **Test Plan** that includes the functionalities of the app and a list of tests we plan to run at the end of Sprint 3, with test conditions, inputs, and expected outputs.

Team Members (Group 62)

- Japteshwar Singh (Read Me/Developer) — 501272180
- Yash Ganwani (Developer/Video) — 501276953
- Adam Wiacek (Client/Product Backlog/Developer) — 501102714
- Joona Huomo (Developer) — 501236987
- Kyle Huynh (Developer/Test Plan) — 501250558
- Ekra Razzak (Developer) — 501237891

https://github.com/newbthatstuff/CPS406w2025-GoWhere.git

UML Diagram


«control»
MapViewer
currentField: JTextField
destField: JTextField
mapPanel: MapPanel
navigationSession: NavigationSession
inputHandler: UserInputHandler
MapViewer(imagePath: String)
startVoiceRecognition(field: JTextField)
Location
x: double
y: double
label: String
getX(): double
getY(): double
getLabel(): String
toString(): String
uses
uses
uses
«control»
NavigationSession
current: Location
destination: Location
routeEngine: RouteEngine
mapPanel: MapPanel
setCurrent(current: Location): void
setDestination(destination: Location): void
start(): void
«control»
UserInputHandler
locations: Map<String, Location>
UserInputHandler(locations: Map<String, Location>)
getLocationFromText(input: String): Location
updates
uses
«boundary»
MapPanel
mapImage: BufferedImage
zoomFactor: double
offsetX: int
offsetY: int
graph: Graph
route: Route
currentSearchResult: String
getGraph(): Graph
setRoute(route: Route): void
searchLocation(query: String): void
paintComponent(g: Graphics): void
mouseWheelMoved(e: MouseWheelEvent): void
mousePressed(e: MouseEvent): void
mouseReleased(e: MouseEvent): void
mouseDragged(e: MouseEvent): void
RouteEngine
graph: Graph
calculateRoute(start: Location, end: Location): Route
creates
uses
optional (external)
has
uses
Route
path: List<String>
setPath(path: List<String>): void
getPath(): List<String>
«boundary»
VoiceSearch
External Python Utility
Graph
adjList: Map<String, Map<String, Double>>
coordinates: Map<String, Point>
addNode(node: String, x: int, y: int): void
addEdge(from: String, to: String): void
findPath(start: String, end: String): List<String>
getCoordinates(node: String): Point

Product Backlog

Requirement	User Stories	Initial Priority	Task	Estimate (*)	Actual Efforts	Status
Accurate information displayed on the map	Exact street names need to be provided.	med	Integrate the map with correct labeling and valid data [MapPanel.java]	2	1	✅ Completed
	Addresses of all the surrounding buildings need to be displayed	med	Fetch and display relevant address data [MapPanel.java]	2	1	✅ Completed
TMU Facility Differentiation	TMU buildings must be highlighted and names provided (INCLUDING & ESPECIALLY ACRONYMS)	high	Add a database of all TMU locations, adding them to the relevant locations [MapPanel.java]	2	1	✅ Completed
Information Input	The student should be able to input their exact location	high	Implement a search bar for students to type in	1	2	✅ Completed

			their address, [UserInputHandler.getLocationFromText()]			
	The student should be able to input their destination	high	Implement a search bar for students to look for their destination. [RouteEngine.calculateRoute()]	1	2	✅ Completed
	Audio voice input should be registered and used.	low	Integrate speech-to-text for location input. (uses voice_search.py external utility)	2	1	✅ Completed (external utility)
Requestable Narration	The program should provide directions audibly if requested.	Medium	Develop a voice guidance feature.	1	1	❌ Not Completed
Smart Destination Pathing	The app should route students to their destination using efficient routing	high	Implement a pathfinding algorithm for optimal navigation	4	5	✅ Completed

	techniques		(e.g., Dijkstra's). `[NavigationSession.start()]`			
Map Interaction	Users should be able to pan around and zoom in/out on the map.	Medium	Implement gesture-based zooming and panning controls. `[MapPanel]`	3	3	✅ Completed
Caching & Offline Mode	As a user, I want the app to work reliably even with limited connectivity.	Medium	Cache map tiles and frequently used routes locally, with automatic offline detection.	3	3	✅ Completed
Route Steps	As a user, I want the route to contain readable steps.	Medium	Implemented ordered, readable steps in routes `[Step.java]`	1	1	❌ Not Completed
Assistant Prompts	Display prompts for navigation steps	Medium	Use Assistant to display navigation prompts `[Assistant.java]`	1	1	❌ Not Completed

*Estimate levels	1 → ½ day	2 → 1 day	3 → 2 days	4 → 4 days	5 → 8 days

Test Plan

These are the detailed test cases for the product to test its functionality and ensure it follows the client's requirements.
They are split into two categories:

1. App Functionality
    - UI and general use of the app
        - Buttons all perform the intended actions when pressed
        - Navigation between menus is easy to understand
        - The app correctly perceives all input from the user
2. Map Functionality
    - Map and all its functions once given the correct information
        - Displays the shortest route between the current position and the destination
        - Leads user to the correct destination
        - The user is easily able to interpret instructions on how to navigate
3. Integration & Module Tests
    - The integration and tests of our app.
        - Interactive map functions correctly
        - Inputs & Outputs are handled correctly
        - Navigation functions correctly

1. App Functionality

Test Case ID	Purpose	Steps	Expected Result	Actual
1.1	Manual location + destination input	1. Tap “Current Location” field 2. Type address 3. Tap the “Destination” field 4. Type destination	The system will examine the location and create the best route from the current location to the student’s required destination. [System calculates route using `RouteEngine`]	✅ Pass
1.2	User provides voice input for location information	1. Press the “Voice Input” near the top of the screen 2. Say “current location is ___” and “destination is ___”	The system will interpret the destination and location commands spoken by the student as written input. [`voice_search.py`]	✅ Pass (external)
1.3	User starts navigation	1. Input the location information and destination information through any means 2. Select the user’s desired route 3. Press “Get Directions”	Given the provided information, the system presents the best directions for the student to reach their destination with the selected options. [`NavigationSession.`	✅ Pass

			`start()` begins step-by-step guide]	
1.4	User requests audible directions (one time)	1. Once navigation has begun, press the audio symbol near the bottom of the screen	The system provides immediate audible navigation for the next turn or instruction.	❌ Not implemented
1.5	User toggles audible directions (permanent)	1. Press the 3 dots near the top of the screen and select “options” 2. Press “Preferences” 3. Toggle on the slider labeled “Audible Navigation”	The app toggles ongoing voice navigation on or off based on the user’s preference.	⚠️ Partial
1.6	Offline & caching support	1. User simulates offline mode if the network connection is lost	The app should continue to display cached map tiles and routes without error. Essential navigation functionality must remain operational until the app restores connectivity.	✅ Pass

2. Map Functionality

Test Case ID	Purpose	Steps	Expected Result	Actual
2.1	User pans and zooms into the map	1. Swipe in any direction while pressing anywhere on the map 2. Place two fingers on the map and bring them closer together, and then farther apart	The map should respond to gestures, allowing smooth panning and zooming. [`MapPanel` updates view correctly]	✅ Pass
2.2	Street names are accurately displayed along with building addresses	1. Either during navigation or not, pan the display of the map around 2. Zoom in and out for varying levels of information	Street names and building addresses should be displayed correctly at all zoom levels.	✅ Pass
2.3	TMU facilities are highlighted and display names	1. Either during navigation or not, maneuver around the map until a TMU building comes into view	TMU buildings should be color-coded and highlighted to accurately display their names.	✅ Pass
2.4	The user's location is displayed after information is input	1. After pressing “Get Directions” a green arrow should appear indicating	The user’s current location should be indicated with a green	✅ Pass

		the user's current location	arrow.	
2.5	Route is optimal	1. Input route information 2. Select from the route options 3. Follow the directions	The shortest and most efficient route should be selected.	✅ Pass
2.6	Route recalculates dynamically	1. Deviate from the suggested path 2. Observe new route calculations	The app should dynamically recalculate the route when the user deviates from the original path.	❌ Not implemented
2.7	Accessibility-friendly routes are available	1. Enable 'Accessible Route' mode 2. Input Destination 3. Start Navigation	The app should prioritize wheelchair-friendly and easy-access routes.	❌ Not implemented

3. Integration & Module Tests

Test Case ID	Integration Area	Purpose	Result	Passed Tests
3.1	`MapViewer` ↔ `MapPanel`	Ensures UI displays interactive map.	Map loads correctly and supports panning/zoom.	✅ Pass
3.2	`MapViewer` ↔ `UserInputHandler`	Convert text input to locations.	`UserInputHandler` returns the correct Location object.	✅ Pass

3.3	`MapViewer ↔ NavigationSession`	Pass current and destination locations to `NavigationSession.`	`NavigationSession` sets locations and initiates route calculation.	✅ Pass
3.4	`NavigationSession ↔ RouteEngine`	Generate route based on user input.	`RouteEngine` creates valid Route objects from locations.	✅ Pass
3.5	`MapPanel ↔ Graph`	Use graph data for map and routing.	Graph nodes and edges correctly used for routing display.	✅ Pass
3.6	`MapViewer ↔ External Voice Utility (voice_search.py)`	Provide voice input to fill location fields.	Python script captures input and fills text field in `MapViewer.`	✅ Pass