Status Summary

Project title: Mountain Hike Planner

Names of team members: Aaron Asibbey, Roman Di Domizio

In the initial phase of the Mountain Hike Planner project, our team IntelliJ IDEA .We integrated JavaFX to craft a dynamic and responsive Graphical User Interface (GUI), enhancing user interaction and visual appeal. Additionally, we incorporated the Jackson library to handle JSON processing, allowing for seamless serialization and deserialization of data. Overall, we have set up everything we initially needed to start our project.

Moving on to the structure of our application, we laid down the fundamental architecture by crafting essential Java classes. We (Roman and Aaron) split the work of the classes in half, but we also worked in person and helped each other with each of the classes so both of our efforts have been put into all of these classes. Main.java serves as the gateway to our application, setting up the primary stage and the initial scene settings. Trail.java was designed as a model class, that holds the characteristics of a trail, such as its name, distance, and elevation—attributes crucial for hikers when selecting a trail. The TrailDataService.java class is tasked with managing trail data, primarily focusing on loading this data from a JSON source, which integrates seamlessly with our usage of Jackson. For the user interface logic, TrailSelectionController.java was developed to handle interactions related to trail selection and that is also where we have developed our list of drop downs and displaying the trail details when selected , and GearChecklistController.java was prepared as a scaffold for future implementation of gear checklist functionalities.

UML Class Diagram Components

Main:

- No attributes.
- Methods:
    - start(Stage primaryStage)**: void**
    - main(String[] args)**: void**

Trail:

- **Attributes:**
    - name**: String**
    - distance**: double**
    - elevationGain**: double**
    - estimatedTime**: int**
- **Methods:**

        - **Getters and setters for all attributes.**

TrailDataService:

- **Attributes:**
    - dataFilePath**: String**
    - objectMapper**: ObjectMapper**
    - trails**: List<Trail>**
- **Methods:**
    - loadTrails()**: void**
    - saveTrails()**: void**
    - addTrail(Trail)**: void**
    - removeTrail(Trail)**: void**
    - updateTrail(Trail, Trail)**: void**
    - getTrails()**: List<Trail>**

TrailSelectionController:

- **Attributes:**
    - trailDataService**: TrailDataService**
    - view**: VBox**
- **Methods:**
    - initializeUI()**: void**
    - updateDetails(Trail)**: void**
    - getView()**: VBox**

GearChecklistController:

- **Attributes:**
    - checklistItems**: ObservableList<String>**
    - view**: VBox**
- **Methods:**
    - initializeUI()**: void**
    - addItem(String)**: void**
    - saveChecklist()**: void**
    - generateChecklistForTrail(Trail)**: void**
    - getView()**: VBox**

Relationships

- Main has a dependency on Stage from JavaFX.
- TrailSelectionController and GearChecklistController each has an aggregation relationship with VBox from JavaFX to represent their respective views.
- TrailSelectionController uses TrailDataService to get and update trail data.
- GearChecklistController uses ObservableList<String> to manage checklist items.

To create this UML diagram, I will use tools like Lucidchart, draw.io, or Microsoft Visio, which provide easy-to-use interfaces for designing and arranging class structures and relationships.

BDD Scenarios

Scenario 1: Selecting a Trail

Feature: Trail Selection

Scenario: User selects a trail from the list

- **Given the application is open and the trail list is loaded**
- **When the user selects a trail named "Rocky Ridge"**
- **Then the details of "Rocky Ridge" are displayed including distance, elevation gain, and estimated time**

Scenario 2: Adding a Custom Item to Gear Checklist

Feature: Customizing Gear Checklist

Scenario: User adds a custom item to their gear checklist

- **Given the user has generated a gear checklist for "Rocky Ridge"**
- **When the user adds "Sunscreen" to the checklist**
- **Then "Sunscreen" should appear in the checklist items**

Plan for the next iteration:

We intend to greatly improve user interaction features and add a great deal more functionality to the Mountain Hike Planner program in its next iteration. We intend to add more code to our platform in order to enhance trail data handling and make checklist modification easier. Our main goal will be to improve the Trail Data Management system so that users can interact with the data in a more dynamic way and access more extensive trail information and real-time updates. We want to implement a location API so that it will have the real function of pulling up nearby hikes for wherever users will be using it. Additionally, we will improve the User Interface for Checklist Customization so that users can easily make, edit, and store customized gear lists in addition to adding personal things to them. More sophisticated JavaFX components will be integrated, and design patterns like Factory, which creates item instances, Singleton, which maintains a single instance of the application's main window, Observer, which updates the user interface in real time, and Strategy, which selects adaptive gear items, will be strategically used to accomplish this.

Two three-hour meetings a week will be held by our team to guarantee the timely completion and comprehensive testing of these changes. These meetings will significantly help our team stay in sync, especially since we want to strike a balance between on-site and remote development work. In order to maintain a stable and user-friendly program, we will concentrate

on coding during these sessions, evaluating each other's contributions, and regularly testing the new features. In addition to facilitating quick feedback and rapid changes, the collaborative setting will improve the development process overall and guarantee that we meet the project's milestones on time.