TESTING STRATEGY AND PLAN

for

NAVATAR

Version 1.0 approved

Prepared by Matthew Berger, Liam Gomez, and Connor Parkinson

Advised by Professor Eelke of the UNR CSE Department

March 17, 2017

Contents

1 Abstract

Navatar is an indoor navigation system for visually impaired students. Unlike existing systems that rely on expensive equipment, Navatar only requires an Android smartphone. The phone's GPS, accelerometer, and compass are used to approximate a user's location and movements and the phone's speaker is used to provide auditory directions to rooms within a building. Buildings on University campuses often have the GIS maps and the design required for Navatar's accelerometer and compass based localization, which is why students are the intended audience for the project. With the application's open source status and modular design, crowdsourcing efforts can be used to integrate maps for buildings on any campus. The application is still under development and Team 6 is working on establishing acceptance criteria and testing strategies for the implementation of a variety of new features.

2 Project Updates and Changes

Team 6 has been working on implementing the planned features in accordance with the project schedule. The team has also made progress with cleaning up the code base and debugging issues encountered with the existing features. There has been a strong focus on making the repo easy to work with and to understand. This involves both setting up an easy way to work with the existing code, and refactoring the existing code carefully in order to improve the readability and viability of the codebase. This is being done through adding regression tests to cover as much of the codebase as possible, and then carefully refactoring and re-running regression tests to check for breakages.

In an effort to make the open source project repository on github more approachable, the team has developed an easy way to create a development environment for Navatar using Vagrant and Docker. On windows, a developer who is interested in developing Navatar can simply clone the repo and type 'vagrant up' in the terminal (with vagrant and virtualbox installed). This will spin up a virtual machine that will then automatically install docker on itself, and then pull in the docker images used for Navatar. In our case, the docker image used for Navatar is one containing the Android-SDK and uses docker-compose to setup the necessary forwarding for the X server on the virtual machine, allowing the android SDK to be seen visually and interacted with.

The Geo-Fencing functionality has been implemented to poll the GPS for the device location, and if accuracy of the readings is too low, the application attempts to use the last known location of the device. When an accurate GPS reading or recent last location is found, the location is compared with the coordinates of supported campuses and buildings. The appropriate building and campus are auto-selected if the user is determined to be within the geo-fences. Efforts are now being made to establish an elegant solution for inputting and storing coordinates for future supported campuses and buildings.

The route history feature has been implemented to store route history to a JSON encoded file in local device storage. Along with storing the route history, the route reversal feature is now implemented, giving the user the option to reverse their navigation on route completion. While both of these features are functioning as intended there are several bugs and edge cases that need to be handled before these features are ready for production. It became apparent that some code refactoring is necessary after implementing these features, especially with the code pertaining to route selection. Efforts are currently being made on refactoring this error-prone code.

The gesture detection feature has been implemented in order to provide a way to control the application and to interact without having to speak out loud in a noisy environment. The gesture controls are system-wide and can be hooked up to any component. The screens will state what they are in the application, and the entire screen will be available for performing gestures, allowing for consistent actions across the entire application. There will be a gesture for announcing out loud what actions can be taken through specific gestures upon request. This should allow for the best user experience possible.

3 Acceptance Criteria for Components & New Features

3.1 Route History - Liam Gomez

This component is responsible for logging and storing a user's previously taken routes. Storing the previous taken routes allows for a route repeat feature to be implemented, where a user has the option to view their route history and start a new navigation from that history. Storing route history also allows for a user to a reverse a given navigation after reaching their destination.

3.1.1 User Stories and Acceptance Tests

1. As a user, after opening Navatar, I should be given an option that takes me to the route history page so I can view my route history.

   AC1. - Route history option is displayed on homescreen if route history exists.

   AC2. - Route history option is greyed out/unclickable or not present if route history does not exist.

   AC3. - Route history page displayed when option is selected.

   AC4. - A user can go back from the route history page to the main page.

2. As a user, on the route history page, I should be able to select a route from my history so I can repeat that navigation.

   AC1. - Previously taken routes are loaded and displayed when a user enters the route history page.

   AC2. - Previously taken routes are displayed with campus name, building name, and room numbers.

   AC3. - Displayed routes are ordered from most to least recent. AC4. User can select a route and immediately start navigation.

   AC4. - User can select a route and immediately start navigation.

3.2 Gesture-Based Controls - Matthew Berger

This component is responsible for providing an additional input method to the user that is outside the existing Google Talkback integration. For a user with visual impairments navigating content heavy menus and input fields can be quite tedious. To improve this, a gesture based input system will be implemented and integrated with the existing user interface. Gesture based input will allow the user to make gestures on their devices screen that control the applications behavior.

3.2.1 User Stories and Acceptance Tests

1. As a user, I want the option to be able to control the application without having to speak in noisy environments.

   AC1. - A user can accept application prompts using a gesture.

   AC2. - A user can reject application prompts using a gesture.

   AC3. - Application can be signaled, by a gesture from a user, to repeat instructions.

2. As a user, I want to be able to long press to add a landmark.

   AC1. - The long press gesture activates landmark addition functionality.

   AC2. - User can input name of landmark and confirm.

   AC3. - Landmark is saved for future navigation.

3.3 Geofencing - Connor Parkinson

This component is responsible for the automatic selection of the campus and building based on the user's current location. This component will use a combination of the smartphones GPS, cellular, and Wi-Fi to provide locational data. This locational data will then be compared with the known geographical boundaries of available campuses and buildings.

3.3.1 User Stories and Acceptance Tests

1. As a user, I want Geofencing so that I don't have to select the campus I am on.

   AC1. - User is prompted to provide location access permission during first geofencing attempt.

   AC2. - GPS is activated and location is found, or last known location is polled if GPS is not available.

   AC3. - Campus is automatically selected if location is on a supported campus.

   AC4. - Building selection screen is shown if campus is auto-selected.

2. As a user, I want to revert to existing campus and or building selection if Geofencing fails (is not accurate enough or times out).

   AC1. - When GPS is inaccurate or unable to be retrieved, last known location is used.

   AC2. - When last known location is not available or outdated, user is notified that geofencing was unsuccessful.

   AC3. - Manual campus selection page is displayed when geofencing fails.

   AC4. - Manual building selection page is displayed when location is only accurate enough for automatic campus selection.

4 Testing Workflow

4.1 Happy Path Workflows

4.1.1 Route Reversal Workflow

Ideally this workflow should be validated when both route history and route reversal are fully implemented. This workflow will be validated inside the Scrugham Engineering and Mines building on campus as it is the building with the most amount of testing.

1. Open Navatar and choose any reasonable origin and destination rooms.
2. Begin navigation and complete route successfully.
3. Verify upon route completion that a reverse route option is given.
4. Click the reverse route button and begin navigating the reversed route.
5. Complete reverse route navigation successfully and verify its accuracy.

4.1.2 Geofencing Workflow

This workflow has already been validated for automatically selecting the UNR campus, but automated building selection has yet to be verified. This will be validated once the GIS maps and geographical coordinates for the various building on UNR's campus have been input into Navatar.

1. Open Navatar on UNR's campus, ensuring the phones GPS is active and Navatar has the proper permissions within the android operating system.
2. Verify the app automatically selects University of Nevada Reno as the active campus.
3. With Navatar open walk into the Scrugham Engineering and Mines building.
4. Verify Navatar automatically selects SEM as the current active building

4.1.3 Gesture Based Input Workflow

This workflow focuses on verifying the user has the ability to use gesture based inputs to interact with Navatar. More specifically having the user long press the screen to add a landmark. This workflow will be validated once landmark based navigation and custom landmarks have been improved and implemented.

1. Open Navatar in SEM and begin a new navigation within the building.
2. During the navigation stop at any landmark.
3. Press and hold the screen for a few seconds and verify an option is given to add a new landmark.
4. Add and save a new landmark at current position. Repeat the route just taken and verify landmarks presence during navigation.

4.2 Negative Testing Workflows

4.2.1 Negative Route History Workflow

There are various edge cases in route history that must be validated, while some of the error handling is already implemented, many are not. This negative testing workflow will be validated upon implementing the remaining error handling cases.

1. User has no route history because they have yet to successfully complete a navigation. Is the route history option still displayed to the user upon opening Navatar? The option should not be displayed.
2. A user cancels or fails to complete a given route, does this given route appear in route history afterwards even though it was not successful?
3. An impossible route is selected, for instance origin and destination rooms are the same. The application should notify the user of the impossibility and fail to save the impossible route to route history.
4. Upon completing a route the route reversal feature is activated, is the reversed route the exact inverse of the initial navigation? Or has the Navatar selected a different path for the reversed route. The reversed navigation should follow the same path as before to avoid confusing the user assuming the reversed route is possible within the confines of the building.

4.2.2 Negative Geofencing Workflow

Due to the unreliability of GPS signals around campus and in buildings, it is important that Navatar reverts to manual selection of routes if the application is unable to acquire an accurate location. This will be validated once more accurate coordinates of UNR's buildings have been imported into Navatar.

1. No GPS based or recent location can be determined. Does the application revert to manual selections for the required inputs needed to begin a navigation?
2. The phone provides a valid location that is outside of any boundaries known to Navatar. Does the application notify the user that they are not within an area currently supported by Navatar?
3. No accurate user location can be determined, does Navatar continue polling the GPS system? To avoid draining the phone's battery Navatar should stop sending requests for GPS locations after a number of failures.
4. A location is given that is equidistant from two building supported by Navatar. Does the application revert to manual selections for the building? The application should revert to using manual inputs in this case to avoid placing the user in the wrong building.

5 Testing Strategy

For Navatar, there will be a variety of tests necessary for smooth development and assurance of completed features. The types of tests that will be conducted can be broken down into three components. The low level unit tests will be run against the codebase for each commit. As features are developed, they will be moved to a feature branch and then tested by members of the team until they are considered stable enough to merge back into the development branch. Acceptance tests will account for the requested features for this project. These are high level end goals and will be carried out toward the end of the project's life cycle for the semester. These will be accompanied by user tests by blind students at UNR and will be adjusted according to this useful input that we could not have foreseen. The project will be considered successful and 'complete' when the acceptance tests are all passed, and a suite of unit tests has been built and can be used for regression and refactoring.

5.1 Test Types

5.1.1 Acceptance Tests

Each team member will be responsible for testing another member's feature. This ensures that no developer is testing features that they may have developed directly. This allows for a more realistic and thorough testing experience, preventing the happy path from being taken due to expectations of how the software is supposed to function based on the code written for it.

The responsibilities for each team member are as follows:

1. Liam Gomez - Geofencing
2. Connor Parkinson - Gesture-Based Controls
3. Matthew Berger - Route Reversal and Navigation History

5.1.2 Unit Tests

There will be a suite of unit tests to test classes, objects, and methods in the codebase at a low-level. This suite will have as close to full coverage as possible ideally. Mocking may be utilized if necessary, and new tests will be written when developing features necessary to eventually make the acceptance tests pass. During the refactoring and code cleanup, this is vital because any regressions will be quickly detected in the form of failing unit tests. Any deviation from intended functionality in the codebase before refactoring is going to be highly visible and therefore easy to adjust to while working on the project. In addition to this, objects will work as expected and methods will function properly if they are passing the unit tests, and can be safely reused and freely adjusted under the condition that they continue to pass their corresponding unit test.

5.1.3 User Tests

In addition to acceptance tests, Team 6 will attempt to find and ask for the help of blind students at UNR to exercise Navatar in its prototypical state. The goal of this project was to mold Navatar into a more consumable form and to begin work on features that can be extended by future developers in the open source community. Having a solid foundation is important. The blind students will be asked to follow the workflows listed in section 4 of this document, and success or failure of these workflows will indicate the solidarity of the foundation of Navatar.

5.2 Reporting Bugs

Any defects found will be reported and distributed amongst the team via the issue tracker on the upstream github repository fork of Navatar, as well as assigned to a category on the Zenhub board on the same repo.

The protocol is as follows:

- A team member must be assigned.
- List reproduction steps.
- The issue must be added to any relevant Zenhub epicsset of tests.

6 Team Contributions

6.1 Matthew Berger

- Created LaTeX document
- Summarized the project updates for gestures as well as Vagrant and Docker setup
- Outlined the testing strategies

6.2 Connor Parkinson

- Completed the abstract
- Summarized the project updates for Geofencing
- Outlined acceptance criteria for use cases

6.3 Liam Gomez

- Summarized the project updates for route history
- Completed happy path testing workflows
- Completed negative testing workflows

6.4 All Team Members

- Responsible for peer review, editing, and formalization of this document as well as the knowledge contained herein.