Jonathan Arban

CSC 415 Software Engineering

Assignment 4

3/24/16

Updated 4/19/16

***Where Did I Park Last Night? Parking Application***

**Requirements:**

1. System must able to track location of user
2. System must use an applicable map application
3. System must be able to drop a pin on location from a touch of a button
4. System must be able to get a path back to pin from another location
5. System should have functionality to be given a parking space number if applicable from a form fill box
6. System should have ability to track parking time if parked at a parking meter
7. System should have ability to give floor number if parked in a garage
8. System should be portable to other devices
9. System must be responsive to the user’s direct interaction with touchscreen
10. System should be adaptable to fit multiple portable screen sizes

**Use Case Description**

**Use case:** Make a marker of parking location via mobile app. Navigate back to marker.

**Primary actor:** Car owners

**Goal in context:** To navigate to a marker designated by the driver’s intention by a map application via location tracker.

**Preconditions:** System must be configured to use map application; have location tracking engaged.

**Trigger:** The car owner wants to find his car after he parks.

**Scenario:**

1. The car owner parks his car in some legal parking space.
2. The car owner takes out his phone and opens the parking application.
3. The application displays multiple fill forms for parking space number, meter time, and garage floor as well as touch button that says PARK.
4. The car owner fills in credentials and presses PARK button.
5. The application now displays a button that says NAVIGATE.
6. The car owner goes about his business.
7. The application keeps time of meter (if given) and notifies owner that time is running out.
8. The car owner opens the application on his phone.
9. The car owner presses the NAVIGATE button.
10. The application will now bring up Google Maps with a navigation point to the location of the car with a button saying STOP.
11. The owner follows the map to locate his car.
12. The owner arrives at his car and presses the STOP button.

**Exceptions:**

1. Location is not enabled on device
2. Google Maps is down for maintenance

**Frequency of use:** Moderate frequency

**Open Issues:**

1. Can the location tracking be used by hackers to track a person’s movements?
2. Is the navigation to the car safe for pedestrians in city locations?

**User Interfaces:**

The user interfaces as displayed on separate PDF files are simplified to the bare minimum need that each screen will need to get the information required. There will be no learning curve for the only items that need to be entered are in fill in form and the initiation buttons are much larger and more stand out than the others. In this vein I feel that the users can easily enter all the data they need and everything else is literally at the touch of a button.

How these aspects follow the Eight Golden Rules are as follows:

1. **Strive for Consistency:** Each page uses the same type of fonts and outline with a simple touch of a button to get to the next step of the application.
2. **Enabled Frequent Users to Use Shortcuts:** The interface is so simplistic I don’t feel there is any need for shortcuts. Normal users will only get faster of navigating the application as they use it.
3. **Offer Informative Feedback:** The fill in forms are clearly labeled. If location is not enabled it will tell the user that it cannot access the current location.
4. **Design Dialogs to Yield Closure:** The user will have be given a confirmation screen before pressing the navigation button so he/she will have two opportunities to see their inputs before closing the application.
5. **Offer Simple Error Handling:** If a user will enter the strings that they deem correct and the application cannot tell if the user entered anything incorrectly. With the nature of certain parking spaces the application will accept letters and numbers. The meter will be on a timer dial and therefore impossible to enter anything other than a correct time.
6. **Permit Easy Reversal of Actions:** Pressing the back button on the phone will bring up the previous page.
7. **Support Internal Locus of Control:** The user controls all the inputs for the application and it simply just keeps the data on file until the user stops the application. The application cannot overtake any of these options.
8. **Reduce Short-Term Memory Load:** The application holds data given by the user and always has it on display on each of the screens. The user will not have to remember anything; it is all there on the application.

**Other Requirements:**

The application will not need many different modules. It will utilize basic variables that are filled in by the user and more or less store the information and have it visible to the user while the application is opened. The meter timing will always be running in the background and can be used as a push notification to the user at a time of the user’s choice. The layout of the code will be easy and narrowing down where something went wrong should not be terribly difficult at all.

There is no need for global variables for this application. The variables of the space number, floor, and time can be defined privately. There will be no alteration of the variables once they are entered so there should be no altercations between classes and no data fumbling.

Dijkstra’s algorithm is already implemented into google maps, thankfully for me. It looks at the starting and ending node and looks at all the possible edges (in this case streets) and finds the shortest path. This is an elegant algorithm used in all GPSs and is easy to calculate based on the vicinity of the car to the user.

The data structures are simple strings and time. That is all that the user will really need to use, and can be read back easily with a simple get method.

**Test Case Design**

There are not many elements to this application. The most difficult testing that will have to be done is in google maps with saving a location and keeping track of the user’s current location. I’m proposing doing regression testing for the most part and building the application essentially on a screen by screen approach. It will utilize bottom-up integration and start with the easy modules such as the fill in forms and then proceed with the map integration. Since this project is built on Android, which uses java, I can run unit tests in Eclipse and ensure that the output is correct. Not only this, but eclipse also offers a debugging tool which I can utilize by creating my own breakpoints while I’m building and make sure that the outputs are valid.

Here is a sample of the test cases I will utilize during construction:

|  |  |  |  |
| --- | --- | --- | --- |
| **Functionality Tested** | **Inputs** | **Expected Output** | **Actual Output** |
| GetSpaceNum() | 45 | 45 | 45 |
| GetGarageFlr() | 2 | 2 | 2 |
| GetMeter() | 15 | 00:15:00 | 00:15:00 |
| SetDest() | Latitude, Longitude | Marker on location | Marker on location |
| GetLocation() | Latitude, Longitude | Marker on location | Marker on location |