TheHappyDriver

Jerry Li, James Shih, Kevin Bang

Interactive Prototype Report
CSE 440
Autumn 2011

Group

Group Manager	James Shih
Documentation	Jerry Li
Design	Kevin Bang
Testing	Jerry Li

Problem and Solution Overview

Traffic is always a huge problem for people living in big cities that can cause much frustration and cost a lot of time. Although it is fairly easy to acquire information about current traffic conditions, it is impossible to stare at the computer screen all the time. Thus, our goal is to make an application that runs on any mobile devices (smart phones specifically) and combines the benefits of different resources available to make commuters informed about their daily commute. The idea is to have a mobile application that not only provides the traffic information the commuter needs, but also includes the feature of monitoring the route and traffic conditions for the commuter and would notify them if there is a potential delay on their daily commute route.

Tasks

Easy task: *Checking daily commute route and setting up alarms.*
About to commute to work, and would like to make sure the route is clear. Then, set up alarms to get informed if anything happens to the traffic on your way to work.

Medium Task: *Checking traffic conditions and using the navigation feature.*
It's your first day of work and you are commuting from Bellevue to Seattle. You want to use the trip planner and the navigation feature because you are not familiar with the commute route. Have the application get the best route given traffic, and have the application guide you while driving since the route is unfamiliar.

Hard task: *Getting an alarm from the application unexpectedly while driving.*
On the way to work in Seattle from the east side, like any other day, when unexpectedly an alert pops up from our application saying that the 520 bridge was closed because of an accident happened a few minutes ago, and that therefore you'll need to take another route. While driving, get the best alternate route.

Interface Revision Sketches

We do not have any major interface revisions for our interactive prototype in terms of content. We built the interactive prototype based on our low fidelity paper prototype and the revision sketches we had to address the issues found during user testing.

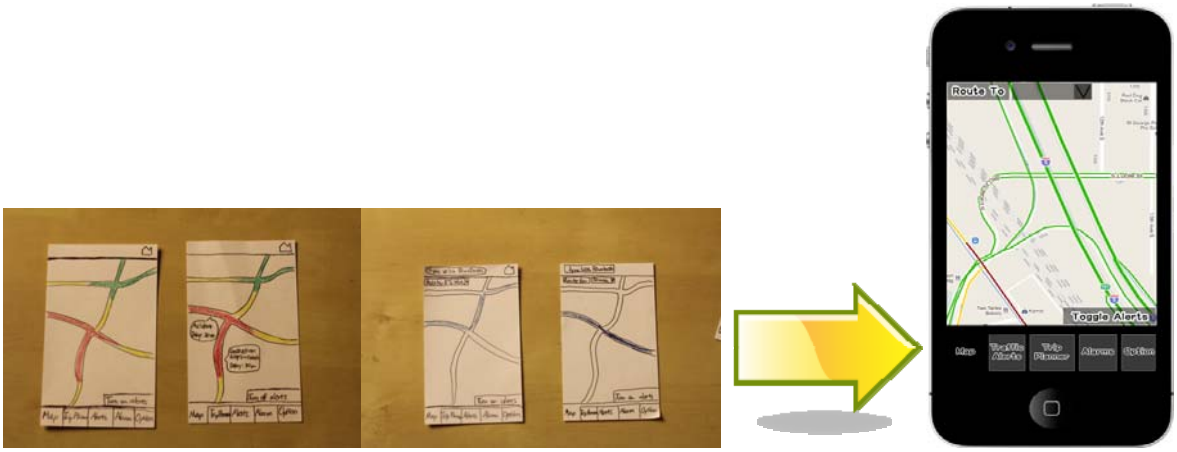

Figure 0 – Design Evolution

Prototype Overview

Overview of Implementation

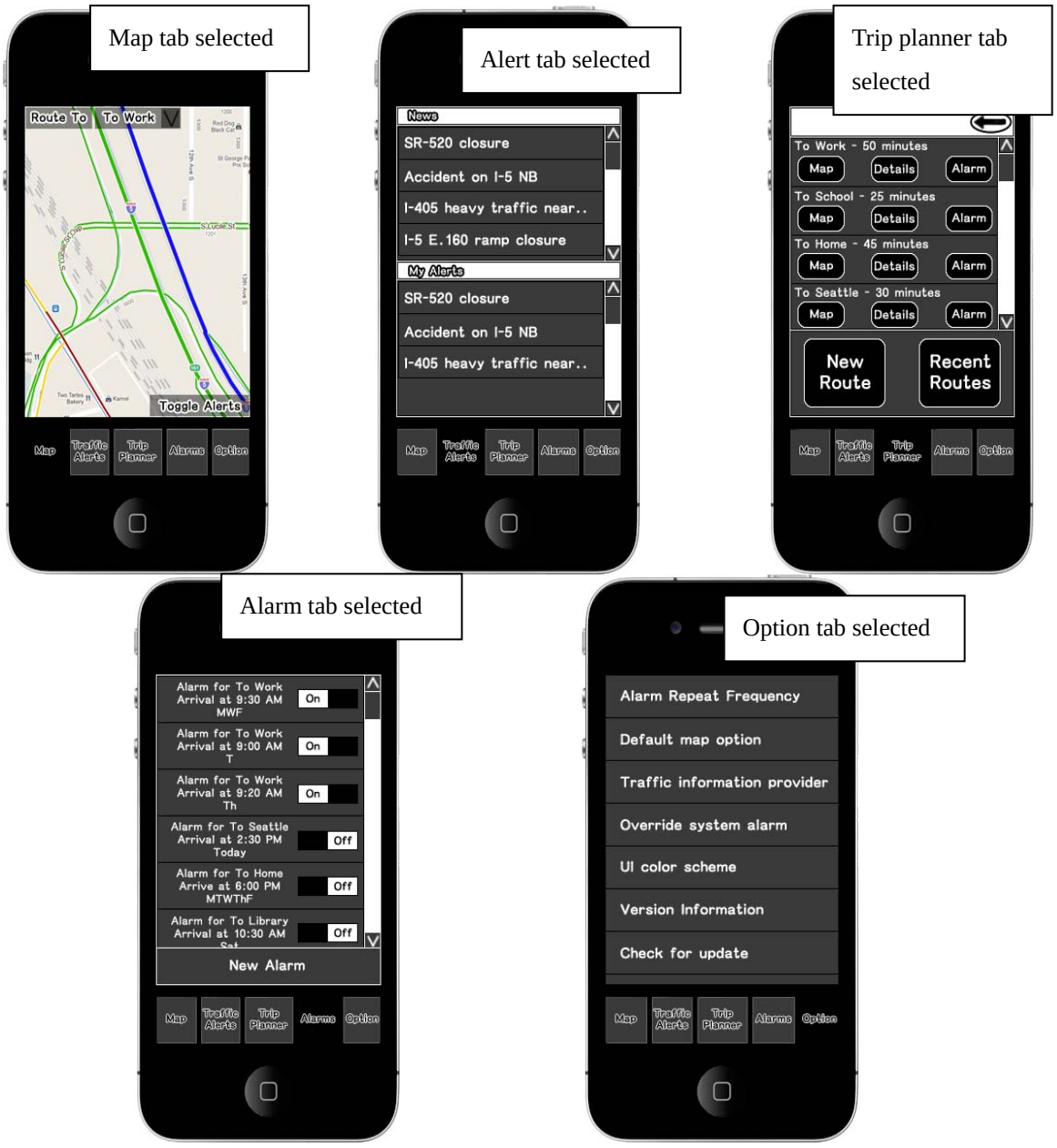

Figure 1 – Overview of Implementation

Our implementation consists of five different tabs: Map, Traffic Alerts, Trip Planner, Alarms and Option, as shown in **Figure 1**. The “Map” screen shows a view of a map with traffic information overlaid. “Alerts” screen is where the user can see news and alerts for their routes. “Trip Planner” has a list of saved routes, right above the “New Route” and “Recent Route” buttons on the screen. The “Alarms” screen displays a list

of saved alarms, with a switch to turn them on or off right next to each alarm. The Options tab consists of a couple of user settings.

Scenarios for 3 Tasks

Scenario 1

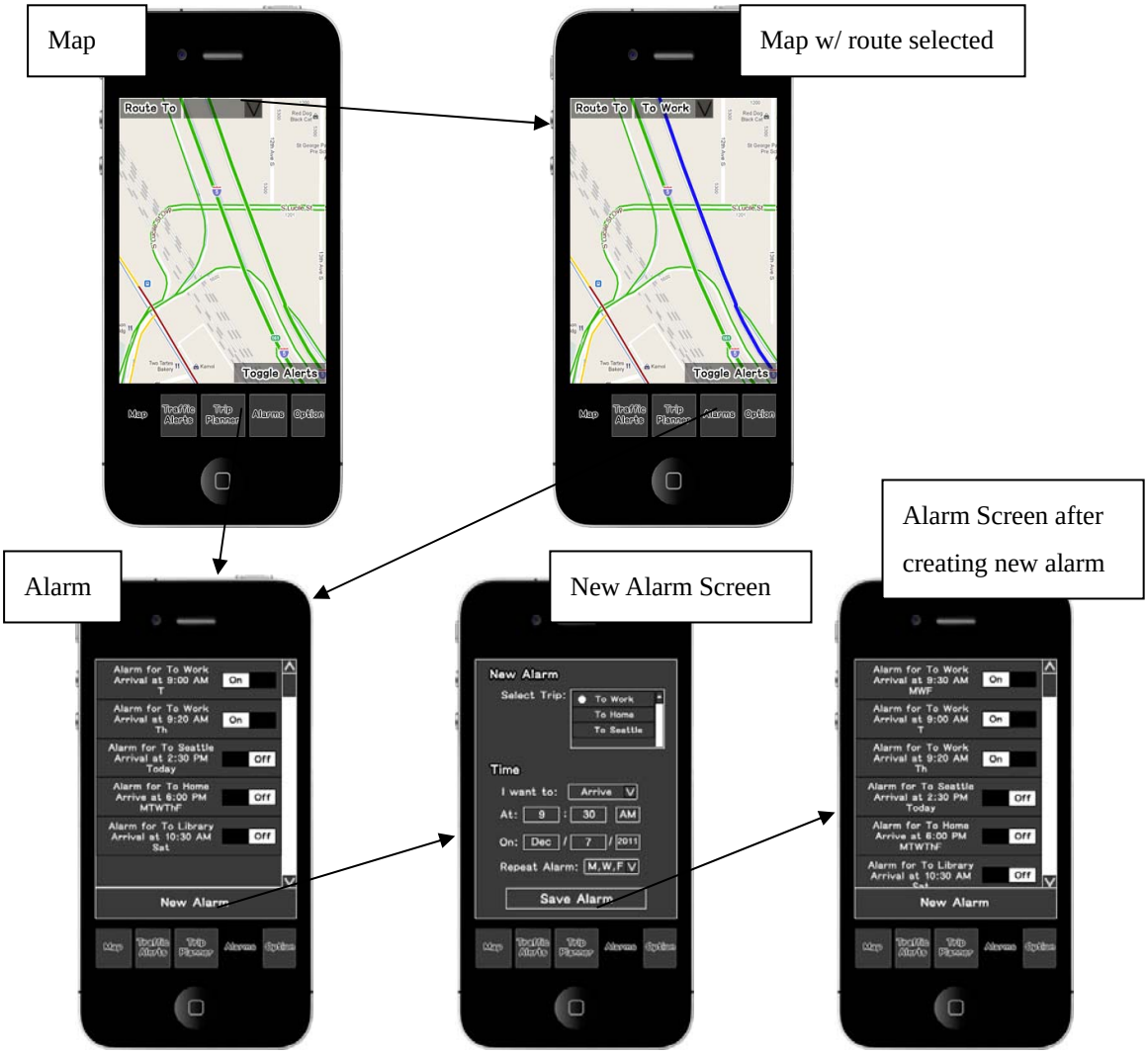

Figure S1 – Storyboard of Scenario 1

The application opens up with a map displayed on the screen, and the user first checks the traffic conditions of their commute route on the map. To do so, the user selects the route they are going to take from the drop-down list in the top-left corner of the screen. At this point, they should be able to see their route highlighted on the map with traffic congestion information. To set up alarms for this route, they then click the “Alarm” tab, and on that screen they click on “New Alarm” button to create a new alarm (assuming the alarm has not been created yet). On the “New Alarm” screen,

they can specify the route, their desired start/stop time and other options (our implementation does not support user input). After having done so, the user can then press “Save Alarm” and return to the “Alarm” screen, where they can see the alarm they just created appear on the list, and is turned on. **Figure S1** shows the storyboard of this scenario.

Scenario 2

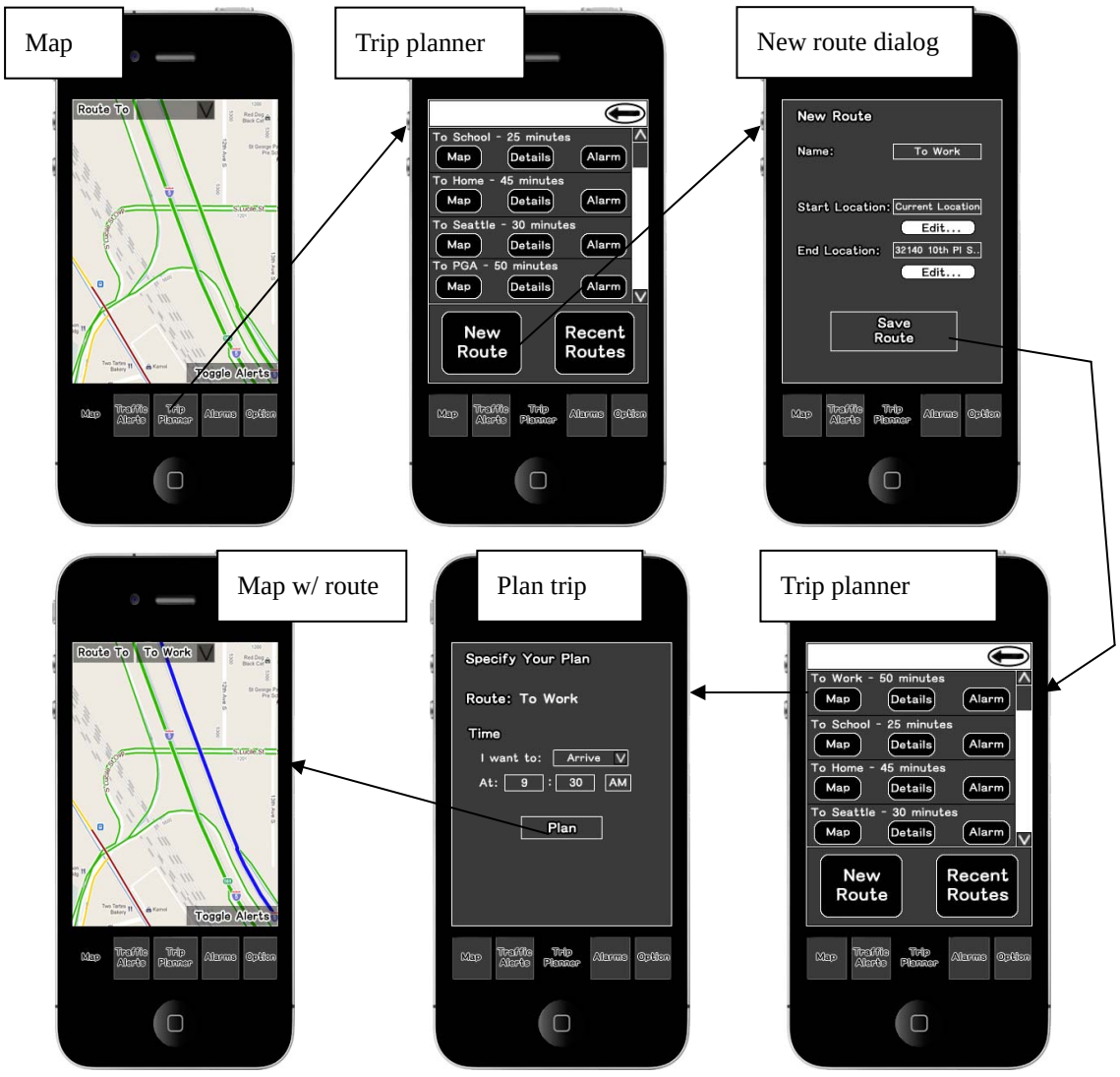

Figure S2 – Storyboard of Scenario 2

The user can again first check the traffic conditions on the map. They then press the “Trip Planner” tab and click “New Route” on the trip planner screen. On the screen that they go to, they can create a new route by providing their start and end locations. After they have created the route, they can now plan a trip with that certain route. There is an estimated time of travel for each route under current traffic conditions. The user can view the map, see more details, or set alarms for each route by clicking

the corresponding buttons on the screen for each route (not implemented).
Now the user can plan a trip by clicking on any saved route and specifying the time and other information on the “Specify plan” screen that pops up. After they specify all the information and hit the “Plan” button on the screen, the application returns to the map with a suggested route, and when the car starts moving, the application turns into a navigation system, and voice instructions are automatically given to the driver (not implemented). The storyboard of this scenario is shown in **Figure S2**.

Scenario 3

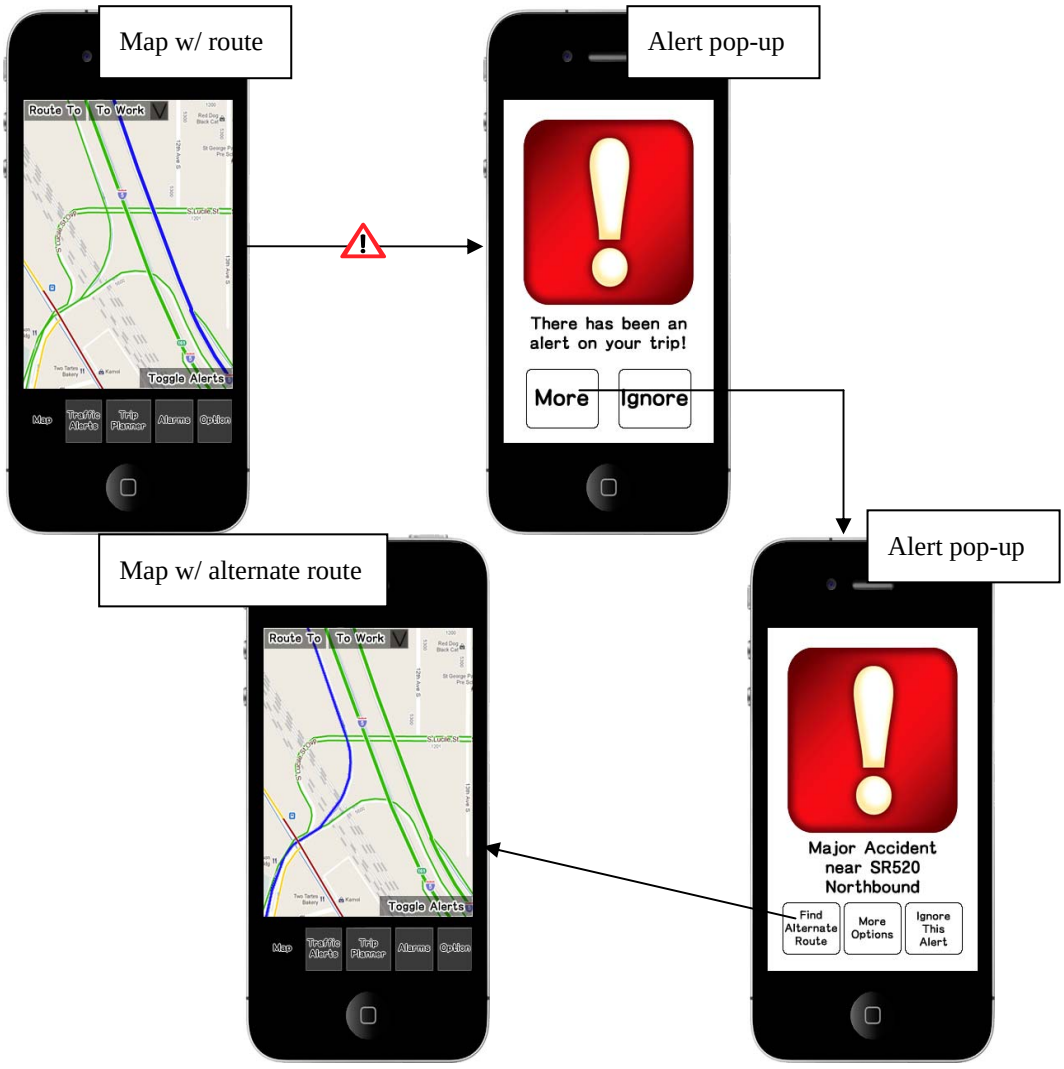

Figure S3 – Storyboard of Scenario 3

While driving, a screen with a big and red exclamation sign pops up indicating that there’s an alert on your current trip. The alert is given out on the screen as well as a digital voice (not implemented). The user is then asked whether or not to learn more details about the alert. After the user confirms, more information will show up on the

screen, with a few choices available to the user (to find an alternate route, to learn more options, and to ignore this alert). Then if the user chooses to find an alternate route, the application will try to find one and return to the map with a suggested route. The storyboard of this scenario is shown in **Figure S3**.

Tools Used

Since only one of our group members had experience in web-programming (not much), we decided not to build the prototype from scratch in HTML and JavaScript. First we started off with building the basic layout of the interface in Photoshop. After we created the images needed for the three scenarios, we started putting them into different HTML pages and linked them together properly. In addition, we used a little bit of JavaScript to help us do things such as refreshing a web page.

Creating the basic UI in Photoshop allowed us to make pretty layouts easier. If we were to do the same thing in HTML and CSS, it would definitely take much longer time for us. In addition, using HTML to link the images we created in Photoshop allowed us to have different people working on different parts of the prototype at the same time. For instance, while one of our members was creating images for the second scenario, another one was able to work on the HTML for the first scenario, without stepping on each other's feet.

One of the team members was moderately familiar with Photoshop. However, the creation of UI turned out to be very time consuming for the lack of accurate positioning tools in Photoshop. Since the screen was quite small, even a single pixel mismatch stood out very distinctively. Placing UI elements in fixed places, and keeping the layout same across every screen was very challenging, especially because it was done by hands, without any programmatic help.

Unimplemented Features

There are a lot of features that were left out of our prototype. Because of the short amount of time, we only implemented the essential parts for completing the three representative tasks. Since we were only to build a fixed-path, task-oriented prototype, we decided not to incorporate any real data to make the implementation simpler. Therefore, our prototype does not support user input, and the maps shown on the screens are just fixed size images, so the user will not be able to drag or resize the map to see more information. The navigation feature was also left out for the same

reason.

In addition, because we created the prototype using a number of images, only the buttons associated with the scenario are functional. In other words, all the buttons should be visible, but we only made a button or tab functional if it was needed to complete the tasks. This tradeoff was made because we felt that
Finally, the prototype does not support voice interaction. Thus, in order to complete the third task using our prototype, we set up the scripts so that the user would need to interact with the interface directly by clicking the buttons on the screen.