LocoMotive: Final Report

COS 333 Final Project – Spring 2016
Brian On, Trey Todnem, Adam Berman
May 10, 2016

1. Overview

For each of us, this was the first time we had developed and implemented a project of this scale. Even though we initially knew what we wanted our application to do, we weren't prepared for how proud we would feel to see our application come together at the end. We hope that through opening the application up to the general public on the Android stores, we can gather valuable feedback/input and improve it even further.

Our group was composed of two seniors and one sophomore. Although two of us will be graduated this year, we all hope to stay involved with maintenance of this application (assuming that people want to use it). Although as a team we were somewhat haphazardly formed at the last minute, we found that we were a cohesive group that enjoyed working on this project.

2. Milestones

2.1 The Founding of the LocoMotive Team

The three of us initially had trouble finding group members. We ended up getting so close to the deadline that Professor Moretti had to help us find one another. There were actually four of us at the beginning – we met immediately once we had a preliminary grouping, since the deadline to come up with a project was the following day. We were in Frist right around the time of Late Meal and the noise was deafening, but we couldn't have been quieter - we spent hours trying to come up with a project that night, but we just could not come up with anything we all wanted to do. Near the end, we tried to narrow down topics that we would be interested in pursuing, but as soon as we had thought we made some progress, one of our group members, the individual who soon chose to work with another group, said that she was not interested, so we had to come up with something else. Tired and frustrated, we decided to each come up with five new ideas and convene in the early afternoon before we had to meet with Professor Moretti to discuss our non-existent project.

Our next meeting started with bad news: our fourth group member decided to just not come and not respond to our phone calls or texts, so we were left wondering what had happened to her. Apparently nothing had happened to her, but we did have good news: one of our team members, Brian, had been trying to figure out the details for how he was going to get home for spring break; in his frustration with the NJTransit website, he foresaw a future where no Princeton Student would have to use the poorly planned website again, and thusly, our project was born. We quickly sorted out some details and the had our project meeting. Professor Moretti seemed to like the idea, and so we were back on track (pun intended).

2.2 Week 1: Preparation and Planning Ahead

We spent the first week after our project meeting learning about some of the libraries and technologies we might use. We decided we wanted – first and foremost – a sleek interface that was quick to respond and easy to use, just for Princeton affiliates. We also wanted alarms to notify users of

when their train would be arriving, and GPS tracking, if possible. We met with our TA, Mingru, and he was very helpful; he helped us figure out some good goals for the next week, and gave us an idea of how to do some of the things we needed to do.

2.3 Week 2: The Prototype

For the Prototype (2nd week), we made it so it was possible to manually input data into a locally hosted Django Rest Framework, and then access it with the Ionic framework on the frontend and display the unformatted text. We did not have any scraping done yet, and we were still working on figuring out the details of the backend. In retrospect, getting this basic goal done by this week was very helpful in getting the rest of the project of to a good start. We were able to do all of the testing locally, so we did not have to worry about weird problems with the back-end, so this probably saved us a lot of time later.

2.4 Week 3: Building out the App

Because we had a basic working prototype, we were able to get a lot done the next week. Adam implemented the scraping of the NJTransit website for putting real data into the database. He initially had some trouble with this because there was no available static web address for scraping, but he soon learned how to use headless web-driving to access the page he wanted (the one with a non specific URL) using Selenium and PhantomJS. He then learned to use BeautifulSoup to retrieve and process the HTML to parse out the desired train data. Trey tried to work on doing the live scraping for the station data, but he found that it was impossible to do the scraping in javascript because of the same-origin policy. We worked on getting the database stored on an Amazon RDS instance, but in the end decided against it because we wanted to reduce latency. Brian also formatted the data for the UI in the front-end and added the cards that showed all the details of the train routes.

2.5 Week 4: Filling in the Details

Having accomplished a good amount in the previous week, we had less to do in the next week. Trey finally got the back-end running on an Amazon EC2 instance, rather than locally. The live scraping was finished, although somewhat sloppily (we made it more robust in the next week). Brian made it so the user could search for train schedules on more than the current day on the front-end. Adam wrote the automatic script to scrape data between locations for future days.

2.6 Week 5: Alpha Version

The next week was the Alpha (5th week). We were supposed to be done adding features by the end of this week. While we did have all of the features we really wanted in, we did not yet have the visualization implemented in Google Maps. We let some of our friends try out our application and adjusted the UI. This was very helpful because they had some insights that were very clear and seemingly obvious but that we had somehow seemed to overlook. Brian also had the live scraping displayed on the UI. From here on, Brian ended up being the bottle neck because the back-end took much less time than the front-end and the front end was all that was left. Brian also changed the UI so that the search options were all on the same page, rather than consecutive pages, and he set an initial background image.

2.7 Week 6: Beta Version

Because Trey and Adam were done with the back-end, they transitioned to helping Brian implemented some of the front-end work for the Beta (6^{th} week). Brian gave Trey some specifications, and then Trey implemented the visualization with Google Maps and Brian was able to insert it into front-end without much manipulation. Adam created functions for Brian that were for making sure that all the data was correct. Brian refactored the code by breaking it into separate services (files).

2.8 The Final Week: Completion

For the final week, before the presentation, Brian got the auto-updating notifications working correctly, and then made some changes in the UI. Adam and Trey worked on making sure there were no bugs on the front-end. Then we all worked on the presentation and practiced. After the presentation, Trey got the application loaded on the Google Play store, which required synthesizing icons, developing feature graphics, and writing product descriptions. There were also a few things that changed when putting the application on the store, rather than running it with full permission from the computer, so those changes needed to be ironed out.

3. Making Decisions about Features

Because we had become a group so late, we did not have that much time to come up with features that we wanted to implement. We initially wanted to implement alarms rather than notifications, but because we had not had time to look up how hard this would be, we did not know that it was not possible to implement alarms in Ionic that would work on iPhone.

In general, when we first met, we were not sure what was expected of our application for the class. Our TA, Mingru, was very helpful because he was able to tell us which features we should make sure we implemented before others. For example, we had wanted to implement GPS tracking, but Mingru told us to focus on everything else first and leave that as a reach goal. We did implement the GPS tracking, after we finished everything else, but we probably would not have been able to get everything done if we had tried to work on everything at once. Prioritizing what needed to get done and then doing it was what got us through this project.

We somewhat arbitrarily chose locations that were supported by our application. However, even though we did not foresee that we would be able to accommodate users that wanted other locations, our applications ends up alllowing users many more locations through our live tracking feature. The user can select New York Penn Station, for example, from Princeton Junction, and the Live Tracking will show all of the stops on the way to New York Penn Station. This means that all of the intermediate stops are covered by our application, albeit maybe not in the best way possible.

Our choice of locations was determined by the stops that a small sample of our friends seemed to think would be best. We did not do any actual polling of Princeton students and faculty to see if there were more stops that Princeton affiliates would prefer to see on the list natively. Our application also does not support Amtrack trains at this time, although many of the routes are available on the website, so it would not be too difficult to add this functionality.

4. Distributing Work

During our project presentation, we were asked what we would have done differently could we

have started the whole project over again. We had some trouble coming up with an answer to this question because our process of getting this project completed was a process of successively figuring out where we were and where we needed to go next. It's not always easy to tell how well other paths could have worked with that kind of tunnel vision, but there was one thing for sure: as Brian answered, it would have been easier if we had more people working on the front-end earlier. As we found near the end of the project, Brian was the bottleneck for implementing new code into the front-end because Adam and Trey had not initially taken the time to learn how to work with the Ionic framework.

However, in terms of distributing work, we still believe that we did a pretty good job. Part of the reason that this was not very difficult was that we met frequently enough that each of us would know if the others were having an unbalanced amount of work to do. Naturally, we would start helping the person that was having the most trouble until the problem was solved, and then we worked on our own things until another problem came up. In this way, we had a way of balancing how much work everyone had to do, without really having to spend time trying to figure out how much time everything would take beforehand. However, sometimes people work better on their own schedules and in their own space, so it was neccessary to balance this efficiency of working alone with the benefits of working together and being on the same page.

5. Selecting Good Libraries and Technologies

Mostly luckily, because Brian had prior knowledge of hybrid phone applications, we had good high level planning at the beginning in the form of making a system that would be easily compatible with cross-platform systems. We used the Ionic framework for creating our hybrid mobile applications, which we highly recommend as it made developing and testing our hybrid mobile application simple. Ionic comes with several clean CSS and JavaScript components specifically designed for phones like modals (we used this for showing the compete train data for a given card) which pop up, customized alerts, cards, etc. that allowed us to create a much more polished application than we would have been able to do by ourselves. In this sense, Ionic is like Bootstrap but specifically for hybrid applications (no web).

The Ionic framework was also great for putting the application on the Google Play Store. Ionic allows you to develop you applcation without worrying about the details of how you will implement the relevant Android or iPhone API. Almost anything that we could write in JavaScript was immediately compatible with iPhone and Android because the framework leverages Cordova to just wrap all of the services for you. We did not need to learn any of the Android or iPhone details because of this. It is a common problem for application developers to required hardware for implementing in a particular mobile operation system. For example, iPhone applications can generally only be developed on Apple computers, but one of our group members had a PC. It would have been a problem to not be able to develop for Apple because a lot of Princeton affliates have Apple mobile devices and we wanted to make sure that the application was available for everyone.

This was also our (Brian's) first time using AngularJS. We highly recommend it as a JavaScript framework; having had prior, minor experience working with raw JavaScript (of course, with jQuery) and working with the JavaScript library Backbone, AngularJS made many aspects of frontend coding much simpler. For example, two-way binding made it extremely easy to immediately show changes to data with minimal code. Furthermore, the various directives, which allow us to bind functions to buttons saved a lot writing boilerplate code that would reload the HTML template. One directive in particular that we found extremely useful was ngRepeat, which is essentially generates HTML elements for all items in a JavaScript list. We used this directive often, as when we had to display cards for a list of train schedules or a list of transfers within a schedule. This saved us from writing individual

HTML elements for each item or writing a needlessly complex function to handle the repetition.

Overall, we believe the successes we experienced in meeting our deadlines and accomplishing our overall objective to create an application was in no small part due to the quality of the tools we used. Learning how to use certain frameworks like AngularJS can be daunting at first, but are well worth the time and energy because they allow you to accomplish so much more, so much more easily. One specific tip is to use spring break to acclimate to the technologies your group will be using; by then you should have your project concept delineated and an idea of what languages/frameworks you will be using. That way, when you return from spring break, everyone is ready to jump into coding. Brian used spring break to learn about Django, Ionic and AngularJS, and it made the first few weeks of creating the prototype a lot easier.

6. Lessons Learned and Moving Forward

One of the key lessons we learned from building this application is the importance of allocating appropriate amounts of time for seemingly "small" aspects of the project. At the beginning of our development process, we did not spend much time fixing "small" aspects or isses of our program, such as determining whether live-scraped data should be interpreted as AM or PM (the website that we scrape from does not say, but uses 12-hour time). It turned out that these seemingly minor issues that we figured would take 10-30 minutes to resolve almost always took at least twice that long, if not three or four times as long. So many of these small issues built up that midway through the semester, we allocated much more time to solve these more minor problems as they arose. This was very helpful to us, because it enabled us to complete the whole app thoroughly, and on time, instead of working too long on implementing too many high-level features that were not essential to our application. This is how our app ended up pretty polished and detail-oriented. In future projects, we will definitely adopt this strategy to efficiently tie off all of the ends of a project before there are too many of them to handle.

Another important lesson we learned was the usefulness of modularity. The backend, which two of members (Trey and Adam) were responsible for, was finished up quickly. Nevertheless, modular coding allowed them to help Brian work on the frontend without having to learn AngularJS or too much of the Ionic framework; Brian was able to enlist Adam's help through simply requesting certain JavaScript functions that accomplish an individual task, which helped to free him up to work on other parts of the application. Through this process, we were also able to accomplish our reach goal of creating a visualization tool to show the user's progress along a given route. Trey worked with implementing this feature in Google Maps implementation in an isolated environment (a simple HTML file for testing), and Brian simply inserted the finished code into an AngularJS service and created a page to display it.

Trey also learned an important lesson. When he was learning how to set up the Amazon servers and relational databases, he had opened up a few test instances of each to figure out all of the details of how they work. Unfortunately, Trey neglected to terminate all of those extra instances, thinking they weren't running if they weren't being used, and then he was charged for all of that extra computation that he was not using at all. Next time, he will make sure that he knows exactly what the terms of use are for using external servers. This could have been a much larger problem if it weren't for the fact that most of these services are so cheap nowadays. Having a single instance open that is not covered by the free services is just a couple cents per hour, but obviously having a lot of them open for many hours can add up.