CSCE 606 Software Engineering
Final Project Report

Jeff Chang, Shao-En Chen, Chun-Jung Chien,
Yi-Min Chou, Jia-Hung Wu, Chu-Ya Yang

I. The main customer need and how the application meets it

In our first meeting, the customer asked us to create a system that students can request for a vehicle and get a ride near campus. Since the service is mainly for students with disabilities and mobility disadvantages, the safety issue is another thing we should focus on. For the management purpose, administrators in the monitor center should be able to log in to the system and check the information of vehicles and drivers. We also need a notification system that if an accident happens or the driver found any issue that may jeopardize the passengers and vehicle, the monitor center should get informed.

We provide a web application for the managing system that is easy to use. When the administrator logs in, he can see information of vehicles and drivers and whether they are currently out on duty. On the index page, he can see where every dispatched vehicle is on a map. When a request from students is sent to our application, we automatically match an available vehicle and a driver, and dispatch them to pick up the students. Their starting locations and destinations are sent to Google Map API to get a predicted efficient route, and the route and the locations are shown on the map in our app so the administrators know exactly where they are.

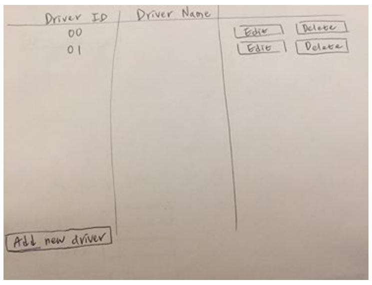
II. Description of all user stories (including revised/refactored stories in the case of legacy projects). For each story, explain how many points you gave it, explain the implementation status, including those that did not get implemented. Discuss changes to each story as they went.

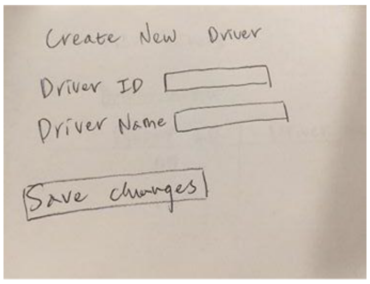
- As an administrator, I would like to perform CRUD operations on database table including drivers, vehicles, and administrators, so that I can manage the database easily.
    - Points: 1
    - Implement Status: Finished
    - Implemented using Rails scaffolds and several custom controller methods
- As an administrator, I would like to see the information of vehicles on duty on the main page, so that I can monitor their behaviors.
    - Points: 8
    - Implement Status: Finished
    - Done in two parts: Google Map info windows on the maps, and tables of status of vehicles, drivers and passengers
- As a TAMUber backend engineer, I need an algorithm to decide which vehicle and driver should be dispatched, so that I can respond to the user's requests.

    - Points: 4
    - Implement Status: Finished
    - Basically, the first-available match can be improved but not what we should focus on for a prototype
- As an administrator, I would like to see the positions of vehicles on duty on google map, so that I can monitor their behaviors.
    - Points: 2
    - Implement Status: Finished
    - Implemented using gem Gmaps4rails and map markers
- As an administrator, I would like to see the driving routes of the start point to the end point, so that I can see whether the cars are in the right position.
    - Points: 4
    - Implement Status: Finished
- As an administrator, I would like to see alert information if a driver sent alert information to the system, so that I would know they need help.
    - Points: 2
    - Implement Status: Finished
- As an administrator, I would like to see the information of the car when I move the mouse cursor to the car in monitor page, so that I can quickly get the information of that car.
    - Points: 1
    - Implement Status: Finished
- As an administrator, I would like to see the color of the car turn red on the monitor page when the car send alert information to the system, so that I would know they need help.
    - Points: 8
    - Implement Status: Finished
    - Can be further improved by integrating with what Safety Interface Team have done

III. Show lo-fi UI mockups/storyboards you created and then the corresponding screenshots, as needed to explain to stories.

- Table Lists of administrators, drivers, and vehicles (Figure 1 and Figure 4)
- Create new administrators, drivers, and vehicles (Figure 2, Figure 5 and Figure 6)
- Edit administrators, drivers, and vehicles information (Figure 3, Figure 5 and Figure 6)
- Lo-Fi UI Mockups/Storyboard:





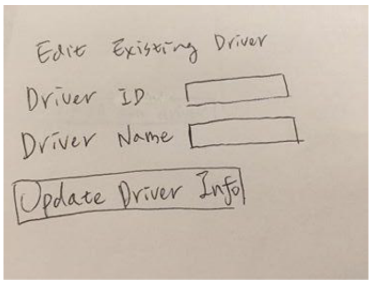
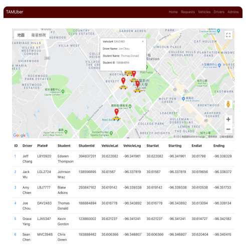


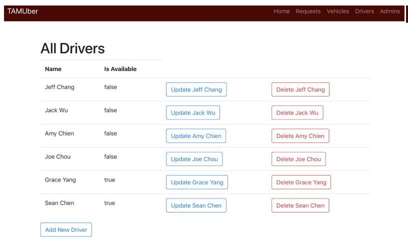
Figure 1 Figure 2 Figure 3

- Screen Shot:



Vehicle#: DAV2483
Driver Name: Joe Chou
Student Name: Thomas Donald
Student ID: 188884894

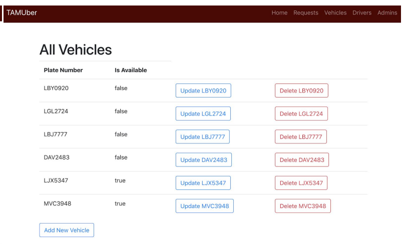
ID	Driver	Plate#	Student	StudentId	VehicleLat	VehicleLng	Startlat	Startlng	Endlat	Endlng
1	Jeff Chang	LBY0920	Edseen Thompson	394837201	30.622082	-96.341961	30.622082	-96.341961	30.61798	-96.338329
2	Jack Wu	LGL2724	Johnson Mraz	138556895	30.61567	-96.337819	30.61567	-96.337819	30.619656	-96.338372
3	Amy Chien	LBJ7777	Blake Adkins	293847102	30.619142	-96.339338	30.619142	-96.339338	30.610538	-96.351733
4	Joe Chou	DAV2483	Thomas Donald	188884894	30.616778	-96.343892	30.616778	-96.343892	30.613094	-96.339134
5	Grace Yang	LJX5347	Kevin Gordon	123860002	30.621237	-96.341241	30.621237	-96.341241	30.614727	-96.342182
6	Sean Chen	MVC3948	Chris Green	193888462	30.606366	-96.346807	30.606366	-96.346807	30.620404	-96.340415



All Drivers

Name	Is Available		
Jeff Chang	false	Update Jeff Chang	Delete Jeff Chang
Jack Wu	false	Update Jack Wu	Delete Jack Wu
Amy Chien	false	Update Amy Chien	Delete Amy Chien
Joe Chou	false	Update Joe Chou	Delete Joe Chou
Grace Yang	true	Update Grace Yang	Delete Grace Yang
Sean Chen	true	Update Sean Chen	Delete Sean Chen

Add New Driver



All Vehicles

Plate Number	Is Available		
LBY0920	false	Update LBY0920	Delete LBY0920
LGL2724	false	Update LGL2724	Delete LGL2724
LBJ7777	false	Update LBJ7777	Delete LBJ7777
DAV2483	false	Update DAV2483	Delete DAV2483
LJX5347	true	Update LJX5347	Delete LJX5347
MVC3948	true	Update MVC3948	Delete MVC3948

Add New Vehicle

Figure 4 Figure 5 Figure 6

IV. List who held each team role, e.g. Scrum Master, Product Owner.

- Scrum Master: Chun-Jung Chien
- Product Owner: Shao-En Chen
- Team Members: Yi-Min Chou, Jeff Chang, Chu-Ya Yang, Jia-Hung Wu

V. For each scrum iteration, summarize what was accomplished and points completed.

- Iteration 1
  - We can add vehicles to the database, and we can edit or delete them.
  - 4 points were completed.
- Iteration 2
  - We add a management system, and we can sign up for or log in to the management system, and change passwords. Besides, we can add drivers to the database, and we can edit or delete them.
  - 16 points are completed.
- Iteration 3
  - We can dispatch an available vehicle after receiving a request. The information is shown on maps. We can see the position of on-duty vehicles on the map, and we can see more information if we click the car on the map.
  - 16 points are completed.
- Iteration 4

    - We add front-end user interface design to our system. We can correctly update our vehicle locations and display them on the map. We can update request status after the vehicles arrive at their destinations.
    - 17 points are completed.

VI. List of customer meeting dates, and description of what happened at the meetings, e.g. what software/stories did you demo.

- 10/5
    - The customer asked us to create a system that students can request for a vehicle and get a ride near campus. Our system should show where the vehicles are, who's the driver, and who's on the vehicle now. One main challenge our customers wanted us to think about was that the vehicles may not have access to the Internet when they are out on duty.
- 10/23
    - We had a more clear division of work with our customers and the other 2 groups that we were to focus on the managing system for the monitor center, and came up with an architecture that a central server (which we did not implement due to the limited schedule) would send requests to the vehicles and get data of them. The data included the geographic location, whether there were mechanical malfunctions of the vehicles, and information of the drivers and passengers. Our group and the other two groups working on this whole project each had our own system, and would access to data sending requests and pushing responses to the central server. Thus we ensured data integrity for the whole service.
- 11/20
    - We demo our TAMUber vehicle app. First, we can sign up for or log in to the management system, and we can also change the password or log out. Second, we can add vehicles and drivers to the database, and we can also edit or delete it. Last, we can receive a request, dispatch an available car and show the position of cars and more information on the map.

VII. Explain your BDD/TDD process, and any benefits/problems from it.

We use cucumber and rspec for BDD/TDD process. The benefits are that we can make sure what we are going to do at the beginning of the iteration and we can test our app to see if we do things right in the end of the iteration.

VIII. Discuss your configuration management approach. Did you need to do any spikes? How many branches and releases did you have?

In the development process, we have created several branches for each iteration (0-4). For each iteration, for each specific feature, we create a branch, such as new features, testing frameworks (used cucumber and RSpec), and UI development. We also have a general debug branch to fix issues to maintain correctness of the master branch. To verify the correctness of our application, we wrote Cucumber and RSpec scripts for function and feature testing. To validate, we had a demonstration to our customers to see if we had captured their needs and get feedback from them.

IX. Discuss any issues you had in the production release process to Heroku.

Since we use SQLite 3 for development and testing, and PostgreSQL for production, there are some syntax differences between the two and need some modification.

Another issue is that in Heroku's documentation, it says there is no asset pipeline for Heroku, and this leads to two problems. One is that since during development we need to fetch data from out controllers using Javascript, we need access to the paths in Ruby/Rails. We've installed a gem called "js-routes" that generates Javascript file that defines all Rails named routes as helpers to solve this problem. The other problem is that we can not access to images on Heroku, so we've put image files to the public directory.

For each iteration, we deploy our app to Heroku for demonstration.

X. Describe any issues you had using AWS Cloud9 and GitHub and other tools.

Most of us have been coding on local machines but AWS Cloud9. Since integrity and collaboration are some of the most critical issues for this project, we decided to create branches for different features, and merge them to the master branch when the functions are complete and correct. In our group when a feature is done, we create a pull request on Github. It is after another team member having completely checked the commits in the pull request that such pull request can be merged into our master branch. By adopting this policy we encounter few issues while working together.

XI. Describe the other tools/GEMs you used, such as CodeClimate, or SimpleCov, and their benefits.

Besides the initial gems that are installed when creating a Rails application, we use jquery-rails to perform an asynchronous HTTP request to controllers using jquery.ajax() feature. As mentioned above, we have js-routes for Rails named routes. For our user interface, we use Bootstrap for fast development and basic responsive web design. We access Google Map API through gem gmaps4rails to show locations of vehicles and put markers on the Google map. And for testing, we use Cucumber, RSpec and SimpleCov mentioned in class to check our function behaviors in early development and ensure correctness.

XII. Link to two-minute video interview with the customer discussing the software that was produced.

Based on what we have developed, we had a review meeting with customers on 11/20. We presented our features like users management, login page, drivers and vehicles management pages, and so on. We also demonstrated a scenario that when a user sends a request to our system, our solution real-time dynamically dispatched a driver and a school vehicle to pick her up. Once this dispatch mission has been delivered, it could be monitored in the monitor page with a pop-out box including information like user, drivers, vehicle, and start/end locations. Detailed information of the request could also be viewed at the bottom section of the page.

Upon our demonstration, our customers asked us to add a feature to let them see whether there is emergency happened to each dispatched vehicle. We back and forth discussed with our customers about how this feature would look like in the monitoring page, and we came up with a conclusion. To monitor the emergency status of every dispatched vehicle, we needed to receive alert status from the Driver Safety team. Then we could highlight the specific vehicle which is in emergency status with different colors on the map of the monitor page.

- https://vimeo.com/305357920

XIII. Link to the 2+ minute demo highlighting your app's main features.

- https://vimeo.com/305331117

XIV. Links to your Pivotal Tracker, Github repo, and Heroku deployment.

- Pivotal Tracker: https://www.pivotaltracker.com/n/projects/2205674
- Github Repo: https://github.com/ShaoEnChen/TAMUber
- Heroku: https://tamu-uber.herokuapp.com/