Progress Report of WPI Cars Project

Group 7: Ashish Gurung, Dingda Zhou, Ke Shao, Weixiao Huang

**Abstract**

The overall purpose of the project is to provide an online platform for the WPI community to conveniently sell and buy their used vehicles without involving a third party/mediator that provide their services for an additional charge and commission. Currently, there aren’t any platforms that facilitate these services, which has forced the people in the WPI community to send out an email to fellow peers and hope that someone comes forward to make a purchase when they intend to sell their vehicles. We developed an online platform to address this problem. The platform is hosted on Amazon Web Service (AWS) and can be accessed by any interested party that wishes to take part in the services of the platform. We have designed a database and set it up on the AWS that is being hosted using Java in the backend. On the frontend we have implemented the pages to view the cars uploaded by other members of the community, a personalized tracker for each user to keep track of vehicles they are interested in or on the lookout for. At the time of writing this report, we feel that we have managed to complete about 60% of the project. The code for the project is hosted on GitHub.

**Key Word**

RDBMS, AWS, GitHub, Java, Javascript, AWS-S3, Lambda function

1. **OVERVIEW**

The primary objective of this project is the completion of our proposed novel system, “WPI Used Cars”, for WPI faculty and students to buy and sell used cars amongst fellow members of the community. The aim of the project is to address the high demand of members in the WPI community to purchase and sell vehicles amongst themselves. To be more specific, we found that Many WPI students buy and sell used cars to other students and faculty however in contrast to auto dealers, making deals amongst people on campus saves money and time. Currently, WPI students send out an email regarding their intent to sell their vehicle with the relevant information on public email groups and hope someone will come forward with the intent to purchase. This is a very inconvenient way of going about with the purchase and sells of one’s vehicle. Therefore, we developed and implemented a database system in the project. Our proposed system ‘WPI Used Cars’, an RDBMS application, is a web service and provides a platform for WPI students and faculty to trade used cars. With our backgrounds and experience, we believe we are able to make the most of our experience by developing a web-based application with a strong database system. We believe that our system would facilitate the people in the WPI community towards a smoother purchasing or selling experience. Also, we love cars.

1. **BACKGROUND MATERIAL**

The background materials we referred to include literature papers, URLs, tutorials, development tools, software environment, system architecture, and programming language libraries.

The literature papers are:

Hu, N., Chen, Z. H., Wu, X. C., & Luo, S. Y. (2004). Research of Vehicle Classification Method in Used Car Price Evaluation [J]. *Journal of Shanghai University of Engineering Science*, *3*.

Shishido, H. (2007). Online used car information search method, program, and device. *U.S. Patent No. 7,184,974*. Washington, DC: U.S. Patent and Trademark Office.

Rose, D. E., & Levinson, D. (2004, May). Understanding user goals in web search. In *Proceedings of the 13th international conference on World Wide Web* (pp. 13-19). ACM.

Madhavan, J., Halevy, A., Cohen, S., Dong, X. L., Jeffery, S. R., Ko, D., & Yu, C. (2006). Structured data meets the web: A few observations.

Spool, J. M., Scanlon, T., Snyder, C., Schroeder, W., & DeAngelo, T. (1999). *Web site usability: a designer's guide*. Morgan Kaufmann.

Gavazza, A., Lizzeri, A., & Roketskiy, N. (2014). A quantitative analysis of the used-car market. *American Economic Review*, *104*(11), 3668-3700.

Belgiawan, P. F., Schmöcker, J. D., Abou-Zeid, M., Walker, J., & Fujii, S. (2017). Modeling social norms: Case study of students’ car purchase intentions. *Travel Behaviour and Society*, *7*, 12-25.

From these papers, we learnt the basic ideas of the used car market and its necessary information that should be provided to customers; the customers or users’ potential demands when using our web services; their searching expectations when browsing our web service; as well as for college students, what factors are likely to be considered when purchasing or selling used cars.

The tutorials and URLs we reviewed:

<https://github.com/kenil0811/Library-management-system> is an example of a DBMS design for the library.

<https://github.com/Swapnil52/DBMS-Project> is an interesting example of designing and establishing McDonald’s online ordering and rating system. This can be a reference of buyers requests about contacting buyers in our web service.

<https://pelotondb.io/about/> is an interesting concept of “self-driving” database management system. What is needed for a truly “self-driving” database management system is a new architecture that is designed for autonomous operation. All aspects of the system are controlled by an integrated planning component that not only optimizes the system for the current workload, but also predicts future workload trends so that the system can prepare itself accordingly. The complexity of managing these systems has surpassed the abilities of human experts.

<https://www.youtube.com/watch?v=OfVBXxwHmBA> An example of web service DBMS about baseball games and ticket order. We reviewed all the elements he prepared for both front end and back end.

The development tools we used are Amazon Web Services ([*AWS*](https://aws.amazon.com/), [*AWS lambda functions*](https://docs.aws.amazon.com/lambda/latest/dg/getting-started-create-function.html), [*AWS S3*](https://aws.amazon.com/s3/), ), Oracle Database ([*Oracle Documentation*](https://docs.oracle.com/en/database/)), Spring Boot ([*SpringBoot Documentation*](https://spring.io/docs)), and Docker ([*Docker Documentation*](https://docs.docker.com/)). We programmed in Windows and MacOS environment. The programming language and its libraries include [*JQuery*](https://api.jquery.com/), [*HTML, CSS, JS*](https://www.w3schools.com/), [*Handlebars*](https://www.w3schools.com/), [*AWS toolkit for eclipse*](https://docs.aws.amazon.com/toolkit-for-eclipse/v1/user-guide/lambda.html) and others.

1. **APPROACH**

**3.1 Basic Ideas**

We are using a traditional Model-View-Controller (MVC) structure for the development of our web application. The MVC is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components is built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development frameworks to create scalable and extensible projects. It is also widely considered a reliable and efficient design, which is easy to maintain and make additions to the functionality of the product in the future.

The Model component corresponds to all the data-related logic that we work with. It represents either the data that is being transferred between the View and Controller components or any other logic-related data. For example, we will retrieve car sellers and buyers information from the database, manipulate it and update it back to the database or use it to render data. The View component is used for all the UI logic of the application. For example, we will design all the UI components such as text boxes, dropdowns, and other functions that the final users (WPI faculty and students) interact with. Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. For example, we handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.

The specific tools and languages are listed in the below table:

|  |  |
| --- | --- |
| Tools and Languages |  |
| Front-end | JavaScript, jQuery, CSS, HTML, Handlebars[for templating] |
| Back-end | Java |
| Database | SQL[ Oracle SQL] |
| Source code management | Version control system [Github] |
| Backend web service | AWS |

The listed tasks and approaches attached to these tools are:

|  |  |
| --- | --- |
| Tasks | Approaches |
| Add Vehicle | The seller uploads all required information about his/her vehicle. The platform creates a new vehicle entry and enables the entry for searching. |
| List All Vehicles | List all of the vehicles in the database. |
| Find Vehicle by Keywords | The buyer enters keywords (make, body type or model) to find desired vehicles. The platform filters the search result for the buyer. |
| Vehicle History | The website will keep a tracking history of the vehicle if it has been added before to the website. |
| Filter Search Results by Keywords | The buyer chooses filter keywords (engine, milage or color). The platform returns a list of fewer vehicles. |
| Sort Search Results by Keywords | The buyer selects sort keywords to get results in a given order. |
| Contact Seller | The platform shows the contact information left by the seller to the buyer. |
| Review | After a successful transaction, the seller and buyer review and leave comments for each other. |

For the evaluation of the final deliverable version, we will conduct surveys among students who wish to buy/sell used cars. We will make comparisons between our platform and other methods of transaction. The test case will be added alongside the development process, which will ensure that our web application passes all the required test cases. A simultaneous pressure test will be added to make sure that the response time of the web application at an acceptable level.

**3.2 Design Diagrams and Figures**

The diagrams and figures of the user interface, software design, and database design:

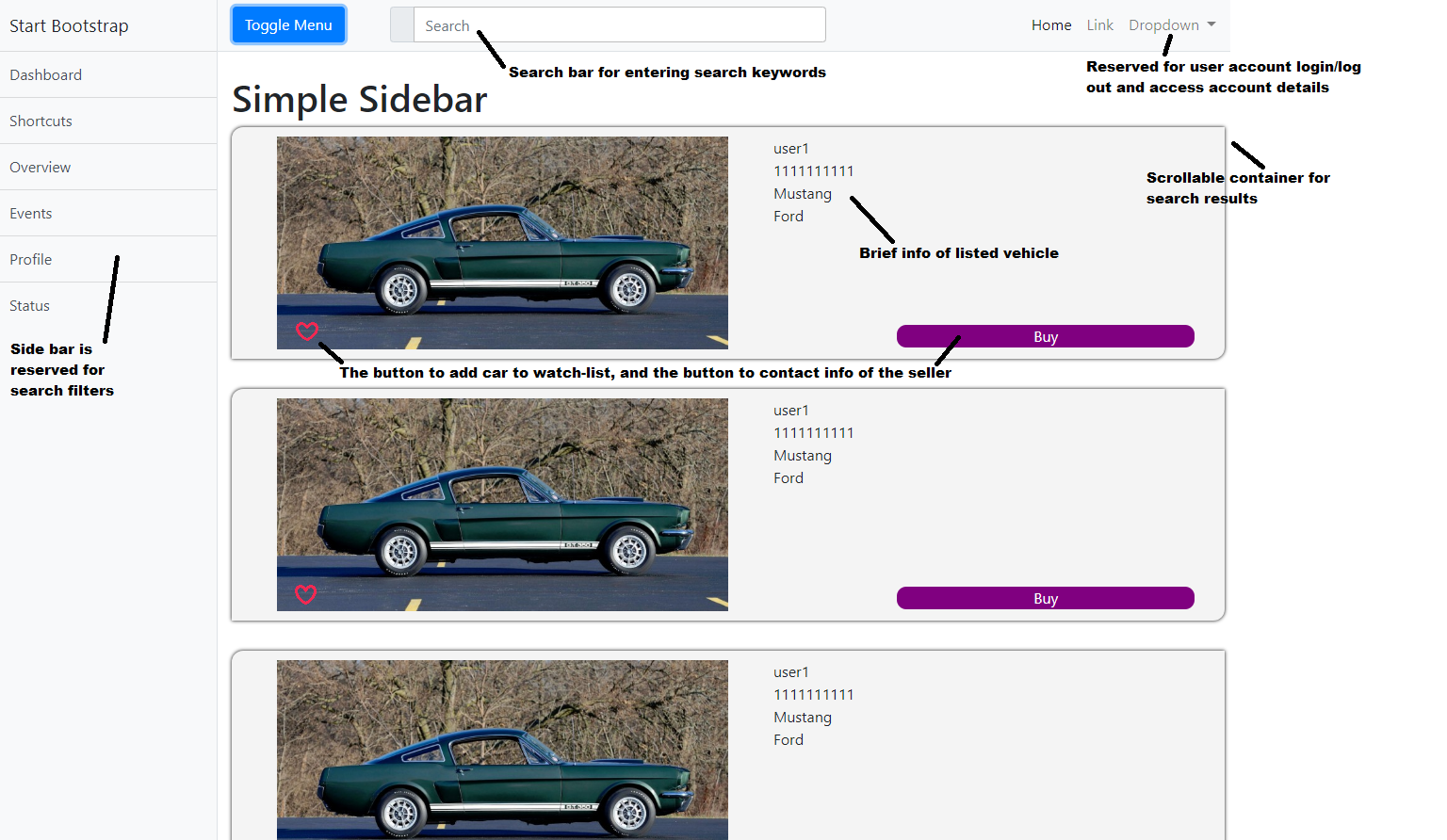


Fig.1 Index page UI design

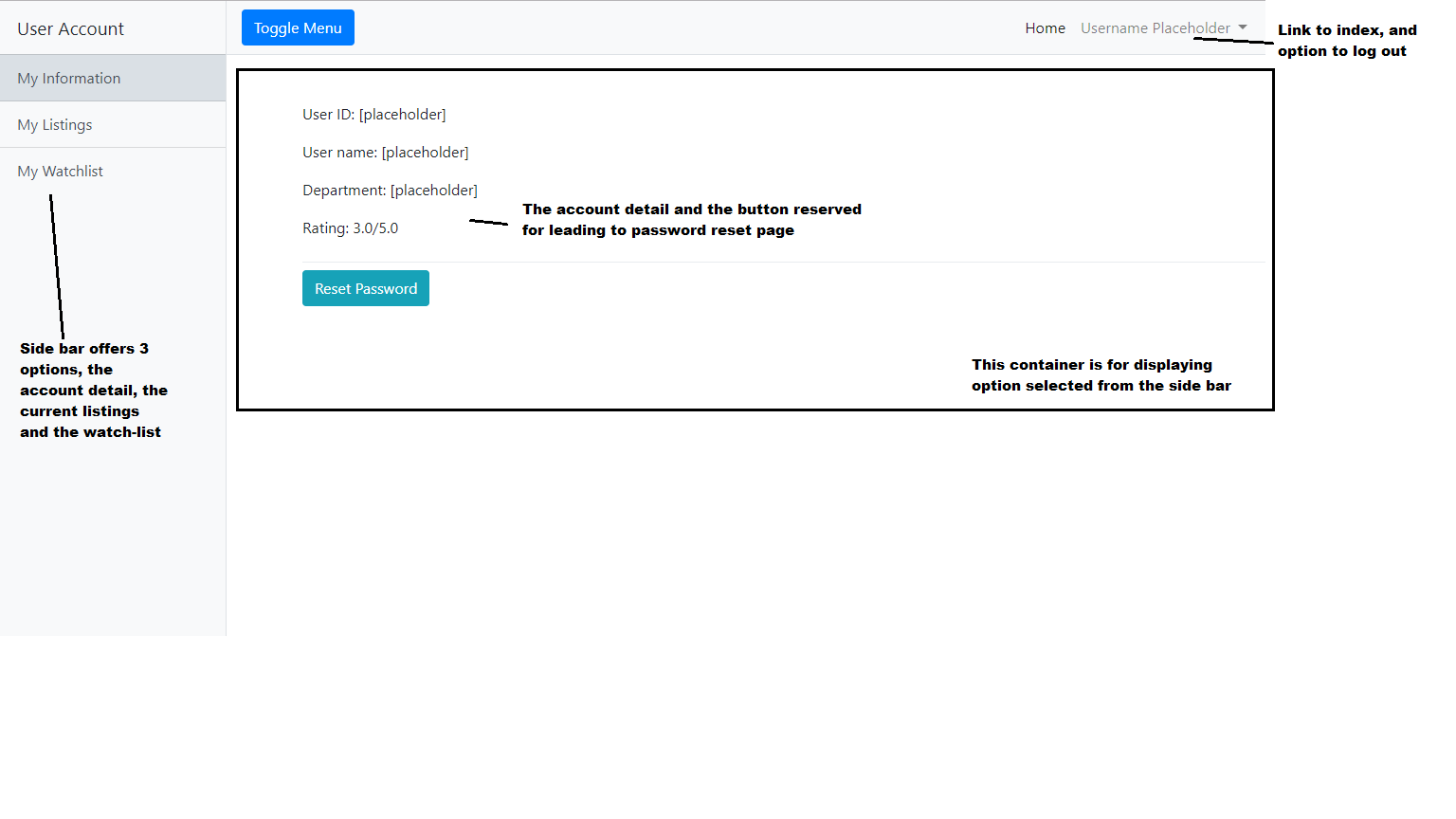


Fig.2 User page UI design (1 of 3)

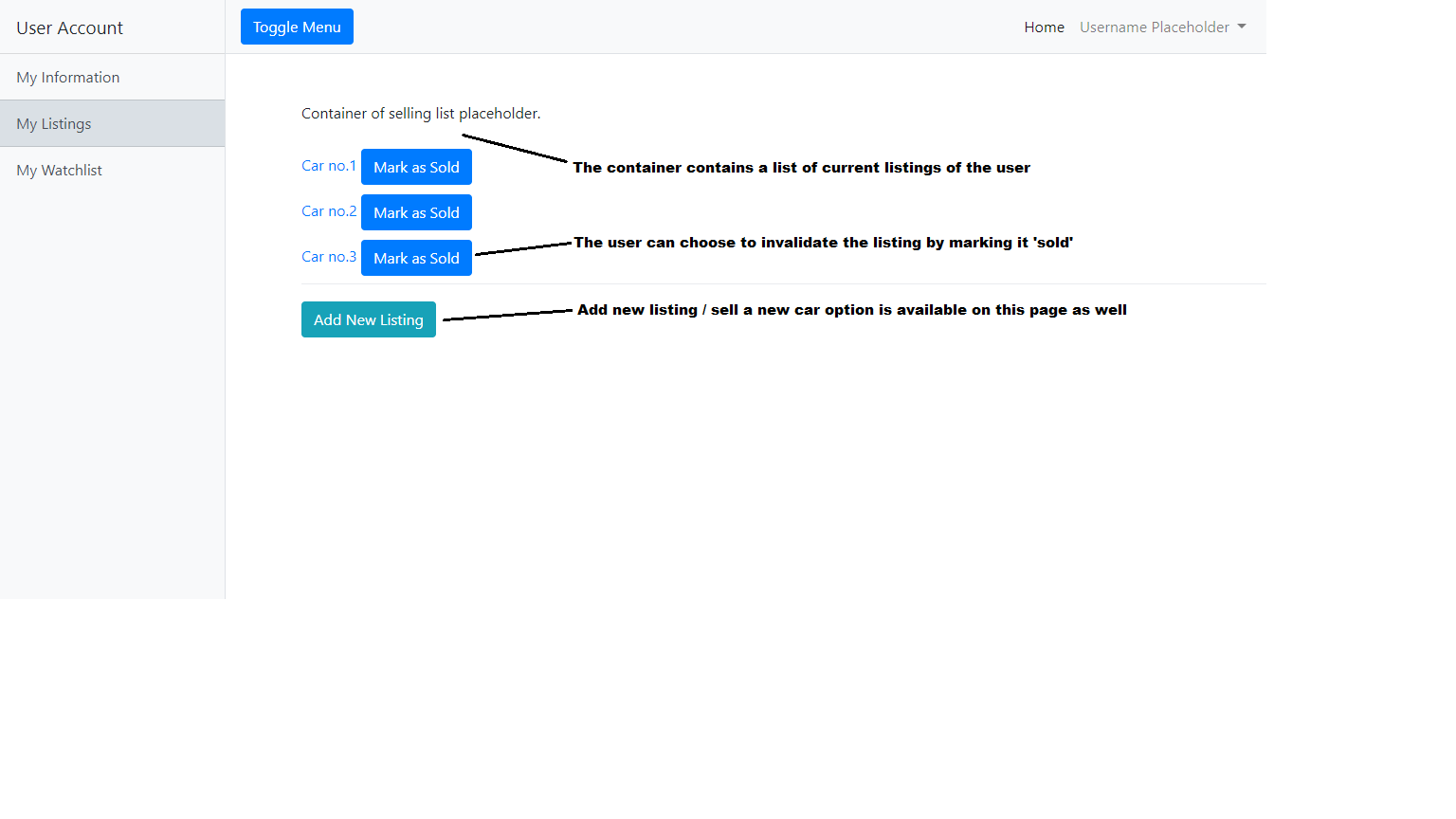


Fig.3 User page UI design (2 of 3)

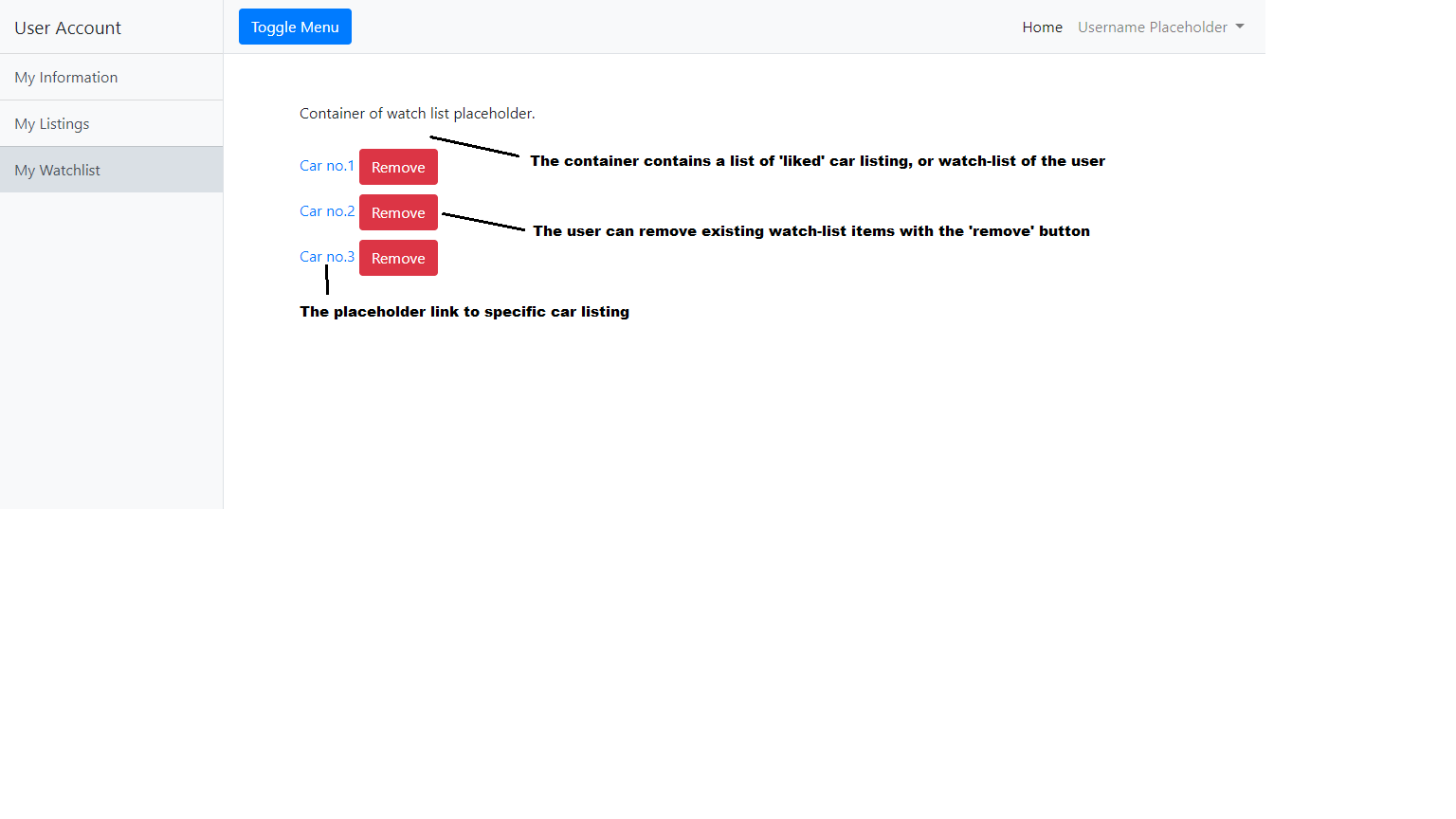


Fig.4 User page UI design (3 of 3)

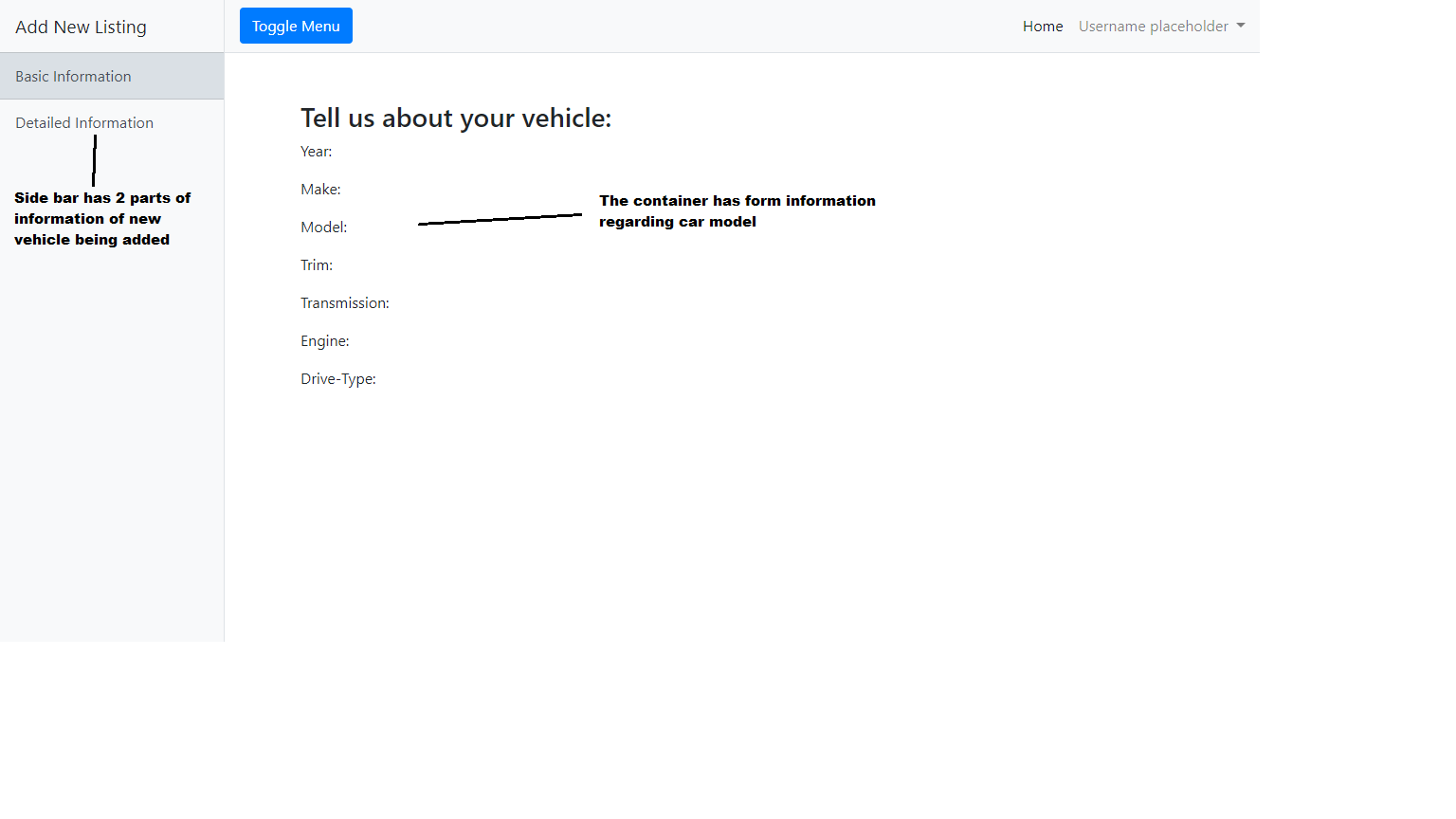


Fig.5 Addcar page UI design (1 of 2)

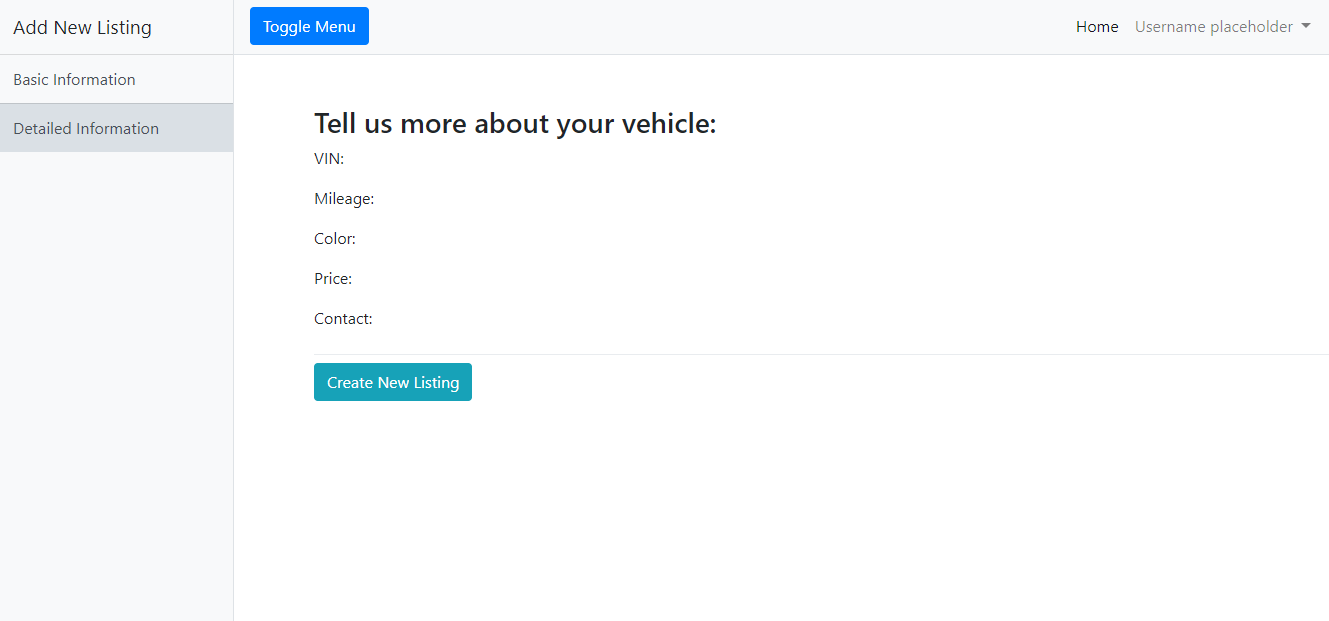


Fig.6 Addcar page UI design (2 of 2)

**3.3 Database and Software Implementations**

Current database design diagram:

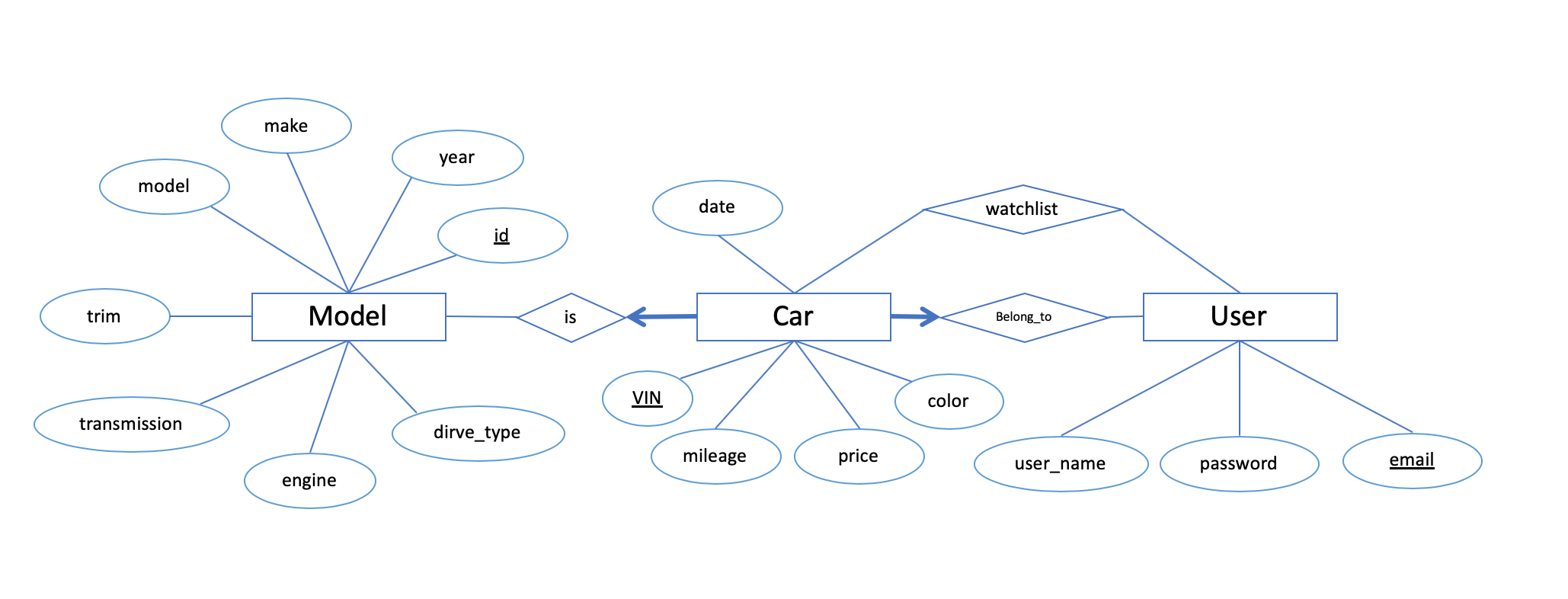


Fig.7 Database design diagram

Tools:

1. Eclipse:  
   We programmed the lambda functionalities for AWS using eclipse. One of the primary reason for selecting eclipse was the stability of an Eclipse IDE and the excellent support it receives from the community and Amazon for AWS related bugs.
2. Atom:  
   For coding the front end of the project we used Atom. It was just a matter of choice. Although it did complicate things when we had to debug the code and re-upload it to S3.
3. MySql Workbench:

MySql Workbench gives us an easy and straightforward way to check the content in the database. It has the ability to create and modify tables in the database using GUI operation, and automatically generate SQL command accordingly, which is very helpful.

1. GitHub:  
   As two people were primarily working on the back end of the project we decided to set up the java project for lambda functions, which allowed push/pull interaction between Eclipse and Github.
2. Bitbucket:  
   We did give bitbucket a consideration as it is private and it also has additional features to help with project management. However given the size of our team and the project we decided not to explore bitbucket as it would only further add to the list of things to keep track, which would hinder our progress.
3. Slack:  
   Initially, we decided to use slack to keep track of all our tasks, documentation and communications but we soon realized that we were getting easily distracted by its rich set of features and rather engaging interface. So we decided to simplify the whole situation and stick to Facebook messenger for communication and planning meetings and Google doc to track our progress.
4. Facebook Messenger:  
   We used messenger to communicate, set up meetings, events, and reminders and create pools to help with a design decision.
5. Google Document:  
   We used the google document to keep track of the progress of the application and ensure that all the use cases were met.

**3.4 Problems**

While implementing the lambda function on AWS, we realized that only rely on the database to validate the user input is not enough. Since we are using the MVC design, all the necessary API is opened to the public, including add new user, add a new vehicle, etc. The JSON request sending to the server could be viewed publicly, which is very easy to be used as a tool to attack our server and database. We are planning to add more input validation to the lambda function to eliminate the unnecessary connection to the database, to prevent unknown request traffic.

1. **LESSONS LEARNED**

Lessons we have learnt so far:

Front-end and back-end tools and techniques?

HTML with jQuery and Bootstrap.

AWS Lambda request handling.

MySQL database and JDBC connection.

Use of software and online platforms for developing?

Eclipse and IntelliJ for backend development.

AWS for deployment and hosting MySQL database.

Github for version control.

1. **CONCLUSIONS**

Our group currently have finished:

1. Design database
2. Setup AWS account for every team member
3. Create Database on AWS RDS
4. Create AWS Lambda Functions
5. Sync development environment for each team members
6. Frontend design

The skills we are practicing are:

1. AWS Lambda Function development

The new tools we are using are

1. AWS Lambda Function
2. AWS RDS
3. AWS Cloudwatch
4. MySQL Workbench

The techniques we are applying are:

1. MVC design
2. **PLAN AND SCHEDULE FOR THE REMAINDER OF PROJECT**

|  |  |
| --- | --- |
| Week 5 | Test and debug. Finish project progress report. |
| Week 6 | Backend development and frontend development. Filter vehicles, contact the seller. |
| Week 7 | Backend development and frontend development for the features in the wishlist. |
| Week 8 | Individual group meeting with the instructor. Adjust the development plan based on the feedback. |
| Week 9 | Test and debug. Finish project presentation slides. |
| Week 10 | Finish project final project report and product. |