CarHub

Thanh Tran, Zohair Khan, Cory Keuning, Yonatan Tafesse, Yod Xiong

Group 4

Table of Contents

Table of Contents, cont.

01

Introduction

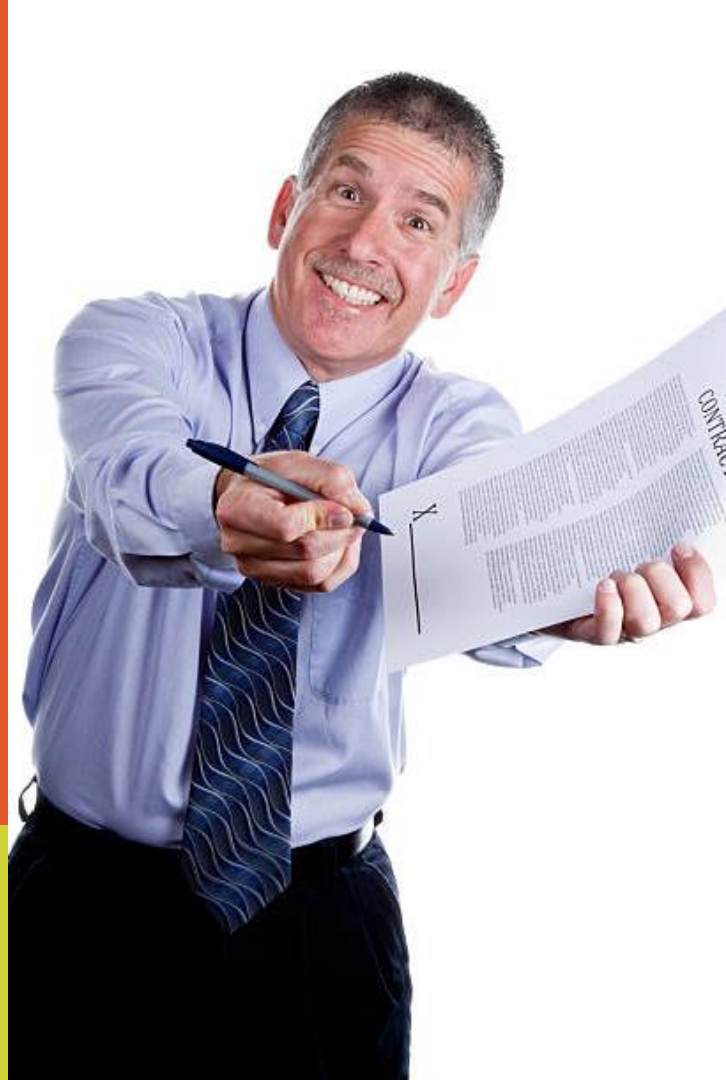
Cory Keuning

Introduction

Welcome to CarHub. Selling a vehicle can be daunting, and so can buying one! Carhub cuts out the used car lot grind and puts sellers and potential buyers in direct contact with each other. Sellers can create a profile with a username, password, and contact information. They can then login and create a listing with their vehicle's information and their contact info.

Buyers can browse or search listings until they find the perfect vehicle. The buyer can then simply call or email the seller and they can conduct the transaction.

No need to deal with this guy.



02

Feature Overview

Zohair Khan

Functional Features

Sign Up

Vehicle Sellers can create a profile on CarHub with a username, password, and contact information through the sign-up page. This information is saved to the database for login verification and future access.

Login

Users can also login with their username and password through the login page. This information is saved to a Singleton “session” which contains information about the logged in User.

Edit Profile Information

Vehicle Sellers can edit their profile information with updated contact information, username, and password.

Create Listing

Vehicle Sellers can create a listing for the vehicle that are selling with a picture and information about their vehicle through the create listing page. Their contact information is automatically included in the listing.

Modify Listing

Vehicle Sellers can modify the listings they have already made with updated information through the edit listings page.

View Listings

Vehicle buyers can view the listings from the base listings page, or they can click on a listing for a more detailed view of the listing.

Non-Functional Features

Data Persistence

This is achieved by storing all the CarHub data in a database the application interacts with.

Simplicity

This is achieved by designing the user interface in a simple manner. There in not too much happening with each screen and the purpose of each button is clear to the user.

Maintainability

The modular design of the application makes maintenance easier. One can make changes in the classes instead of having to deal with the UI interacting directly with the databasse.

Availability

The application, if released into the real world, would be available to user 24/7.

03 Software Architecture and Design

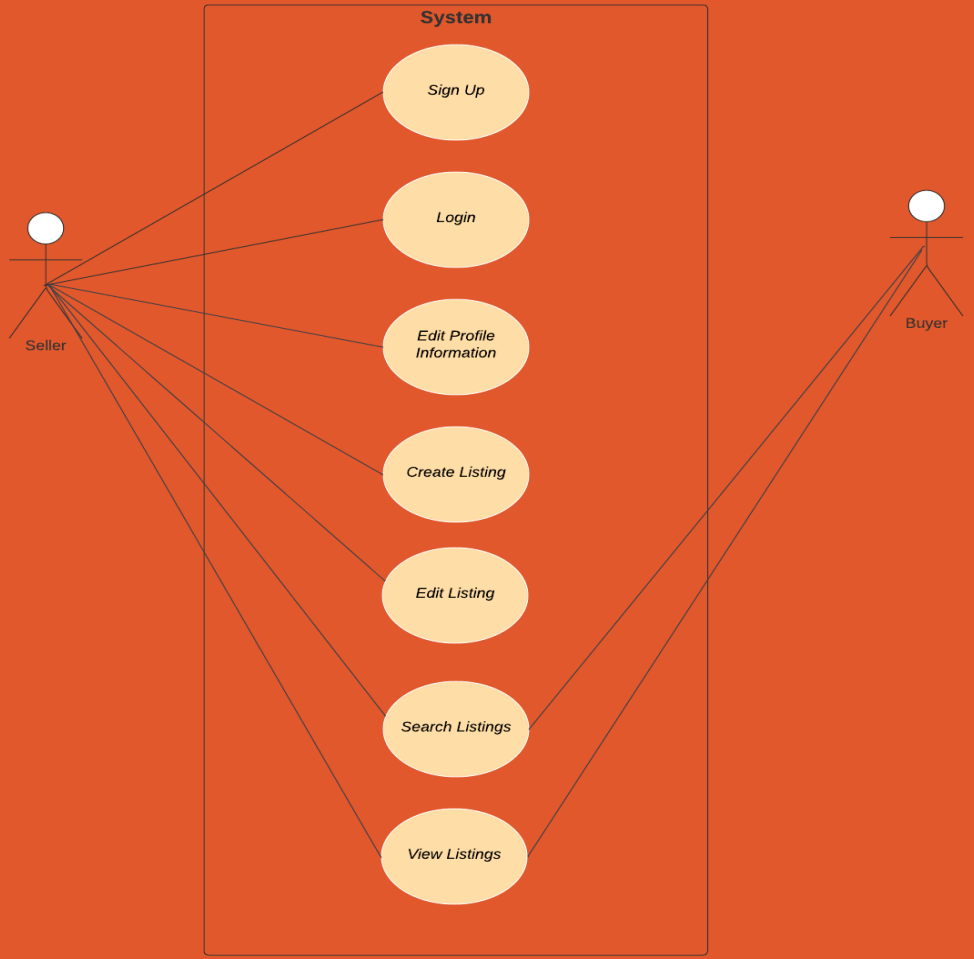
Thanh Tran

Description

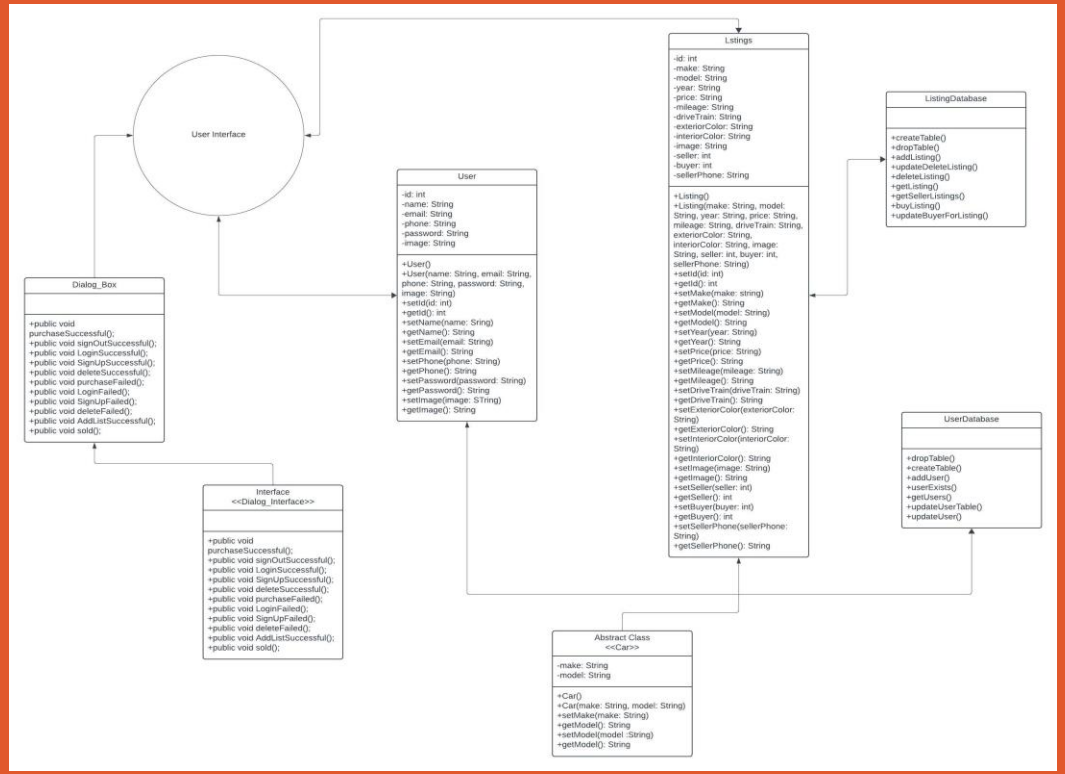
CarHub is implemented internally through the combination of JavaFX, Java classes, and a MySQL database using the model view controller architecture pattern. When a user interacts with the UI to create a profile, sign in, create a listing, or edit their information, the data is sent to and stored in the database.

When a user views the main listings page, searches the listings, filters the listings by keyword, the data is queried from the database, listing objects are created by the Java classes, and the objects are displayed on the UI.

Use Case Diagram



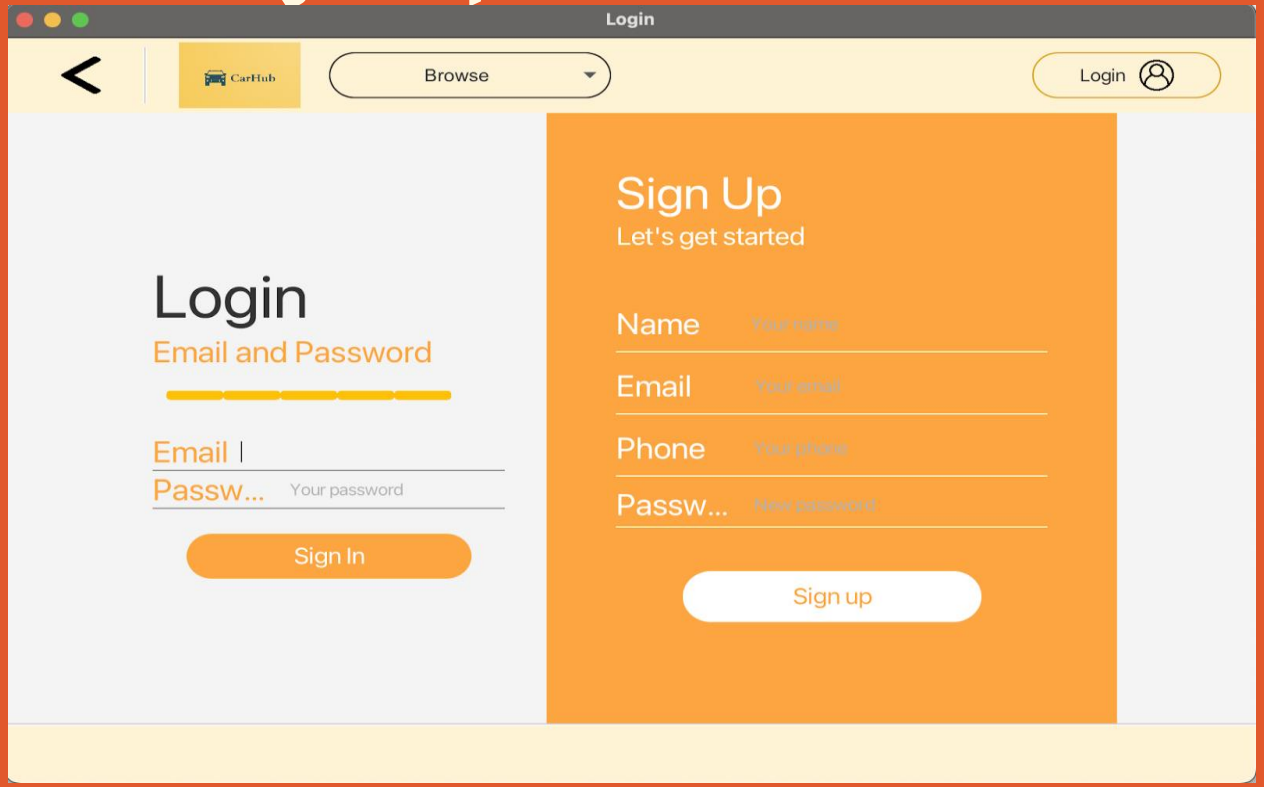
Class Diagram



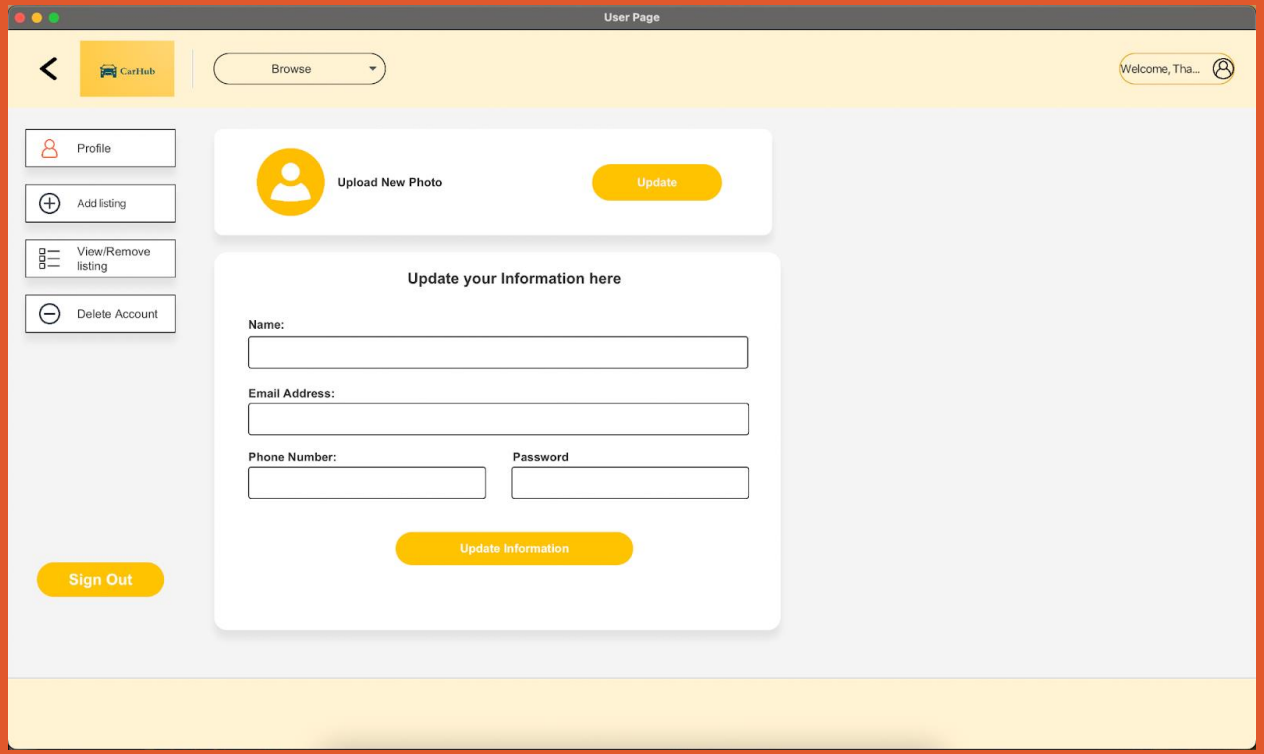
04 User Interface Design

Yonatan Tafesse

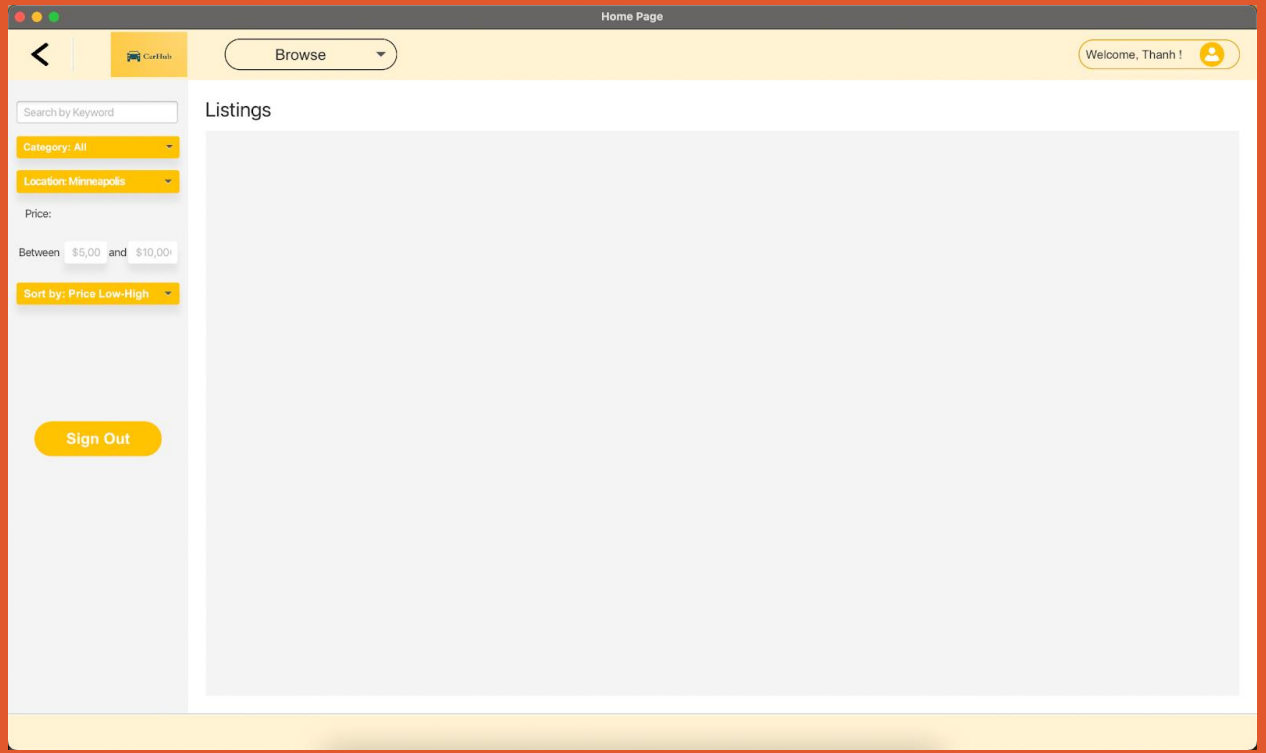
Login / Sign Up



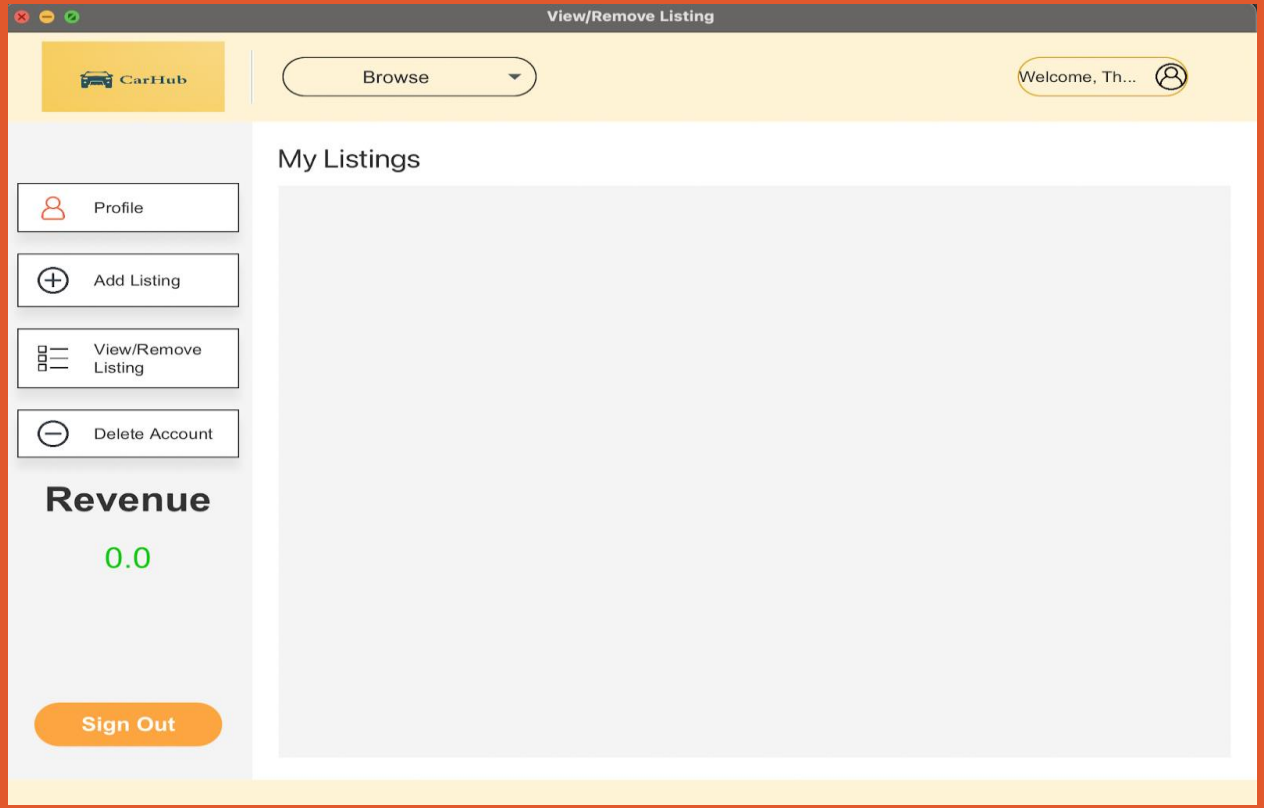
Update Info Page



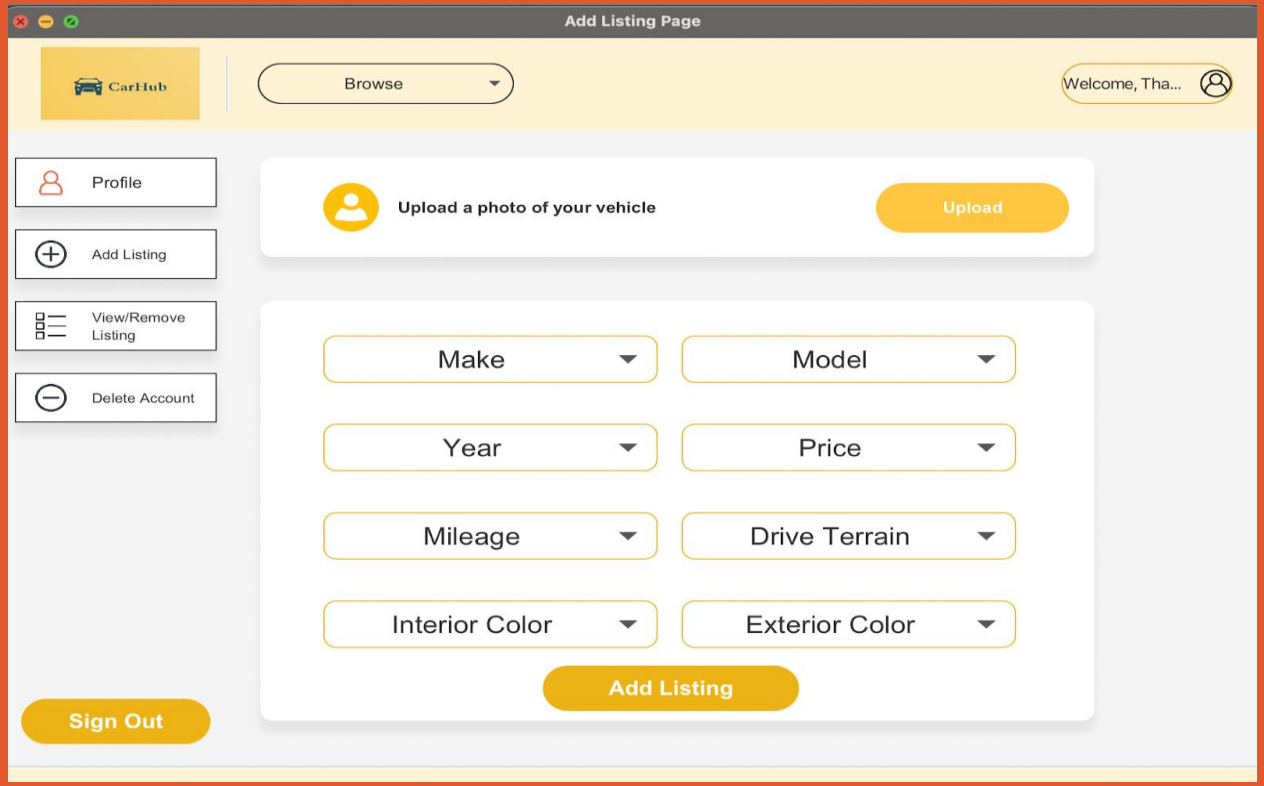
Listings Page



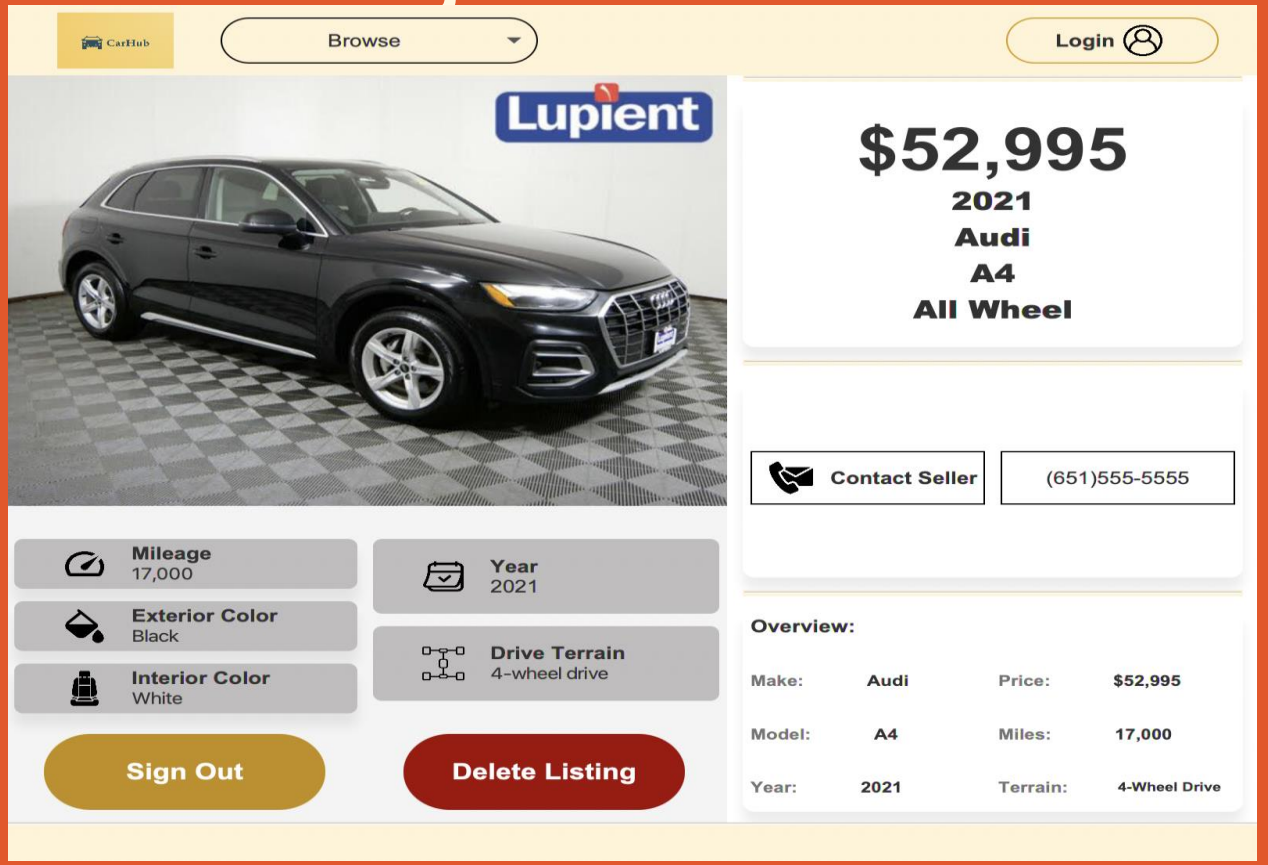
My Listings Page



Create Listings Page



Vehicle Profile Page



05 Development Process

Zohair Khan

Work Organization

For the most part, group members naturally fell into different roles and took on tasks in which we felt comfortable tackling. We didn't development a rigid system of organization.

Sharing Code

We shared code through a mix of a GitHub repo and uploading files to our Discord channel.

Feature Creation

We came up with the features of our project together in class at the beginning of the semester. Our strategy was to come up with the minimum number of features, but enough to make a usable application. Basically, keep it simple.

Feature Testing and Completion

Our testing and feature completion strategy was simply to keep working on each feature until it worked.

06 Demo

Yonathan Tafesse

07

Summary

Cory Keuning

What went well?

The beginning of the project went very well. Our work organization strategy held up, team members completed tasks they felt comfortable with, and we scored very well on the first three milestones.

What didn't go well?

Our code sharing strategy was not good. We originally had a GitHub repo set up for us to work on the project together, but it was abandoned after milestone three. Members starting working on the project on their own with separate codebases. This made it near impossible to work together and easily delegate tasks.

What didn't go well? Cont.

Procrastination was another problem for our group. We never came up with an organized scheduling system, and with large chunks of time in between milestones, it was easy for us to put it off until the end. This led to scrambling before due dates and no time to fix issues with the project, especially with the different codebases.

What would we do differently?

Address the problems we had in the previous slides. Sticking to the same codebase would have helped us tremendously. Everyone could have stayed on the same page and delegating work would have been much easier. Developing a schedule would have given us much more time to complete tasks and fix issues when we ran into them.

We wish we knew.

Looking back, we should have taken more time in the beginning to come up with a simpler project to take on. It seems we may have bit off a little more than we could chew.

Lessons Learned

Planning! We learned how crucial planning is when it comes to software development. Better planning of our architecture and design, a schedule for completing tasks, and who is completing what task makes for a smoother development process.