

Final Report

Team

Brian Blockmon (brian.blockmon@sjsu.edu, 014984060)
Munkh-Erdene Khuderbaatar (munkh-erdene.khuderbaatar@sjsu.edu, id)
Jugraj Pandher (jugraj.pandher@sjsu.edu, 015791386)
Abel Seno (abel.seno@sjsu.edu, 014476345),
Jared Soliven (jared.soliven@sjsu.edu, 012080718)

Department of Computer Science, San Jose State University

CS 157A: Introduction to Database Management Systems

Dr. Ramin Moazzeni

May 10, 2024

Goal and Description

Our goal was to create a web app that provides a seamless platform for users to buy and sell Toyota cars online while offering a user-friendly experience and eliminating the complexities of traditional dealership systems. Our web app allows users to post cars for sale on the internet and also to view cars that are for sale. It allows users to enter specifications that make it easy to search for cars in specific areas. Our app focuses on Toyota car sales, and is populated initially with Toyota cars for sale. It is inspired by the Craigslist website and allows users to directly buy from or sell to the owners of cars, and thus, does not support the dealership system. The reason for focusing on Toyota cars is because a general car web app is large, and we needed to scale it down accordingly.

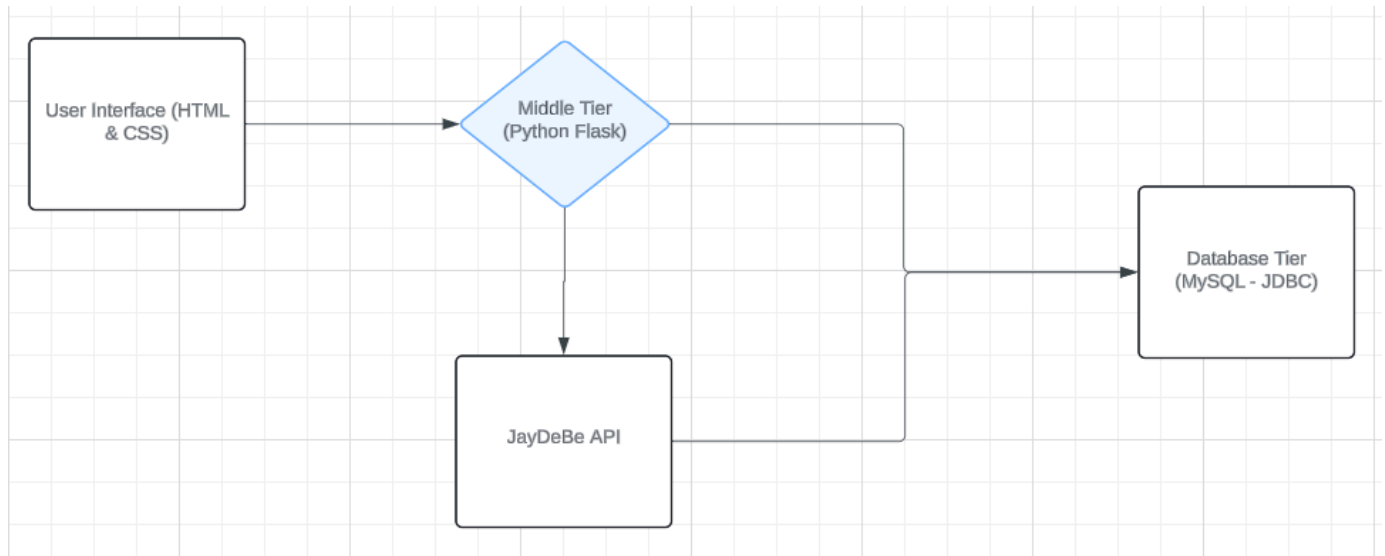
Application/Functional Requirements

1. Search Functionality:
 - a. Users must be able to search for specific Toyota cars based on various criteria including their current location, model type, body style, or manufacturer name.
 - b. They should also have the ability to view the posts of cars that match their search criteria.
2. Posting Cars for Sale:
 - a. Users must be able to post Toyota cars for sale on the platform, providing relevant details and images of the vehicle.
3. Bookmarked Posts:
 - a. Users should be able to bookmark posts of interest for later viewing or reference.
4. Login and Signup:
 - a. The platform should support user authentication, allowing users to create accounts (signup) and log in securely.
5. Account Deletion:
 - a. Users must have the option to delete their accounts if desired. Upon deletion, all related records associated with the user, including posted cars, bookmarks, purchases, and sales history, should be permanently removed from the database.
6. Viewing Past Transactions:
 - a. Users should be able to access their past purchases and sales history on the platform, providing a comprehensive overview of their activities.
7. Reporting Issues:
 - a. Users must have the ability to report any issues or concerns to the website administrator for resolution, ensuring a smooth and reliable user experience

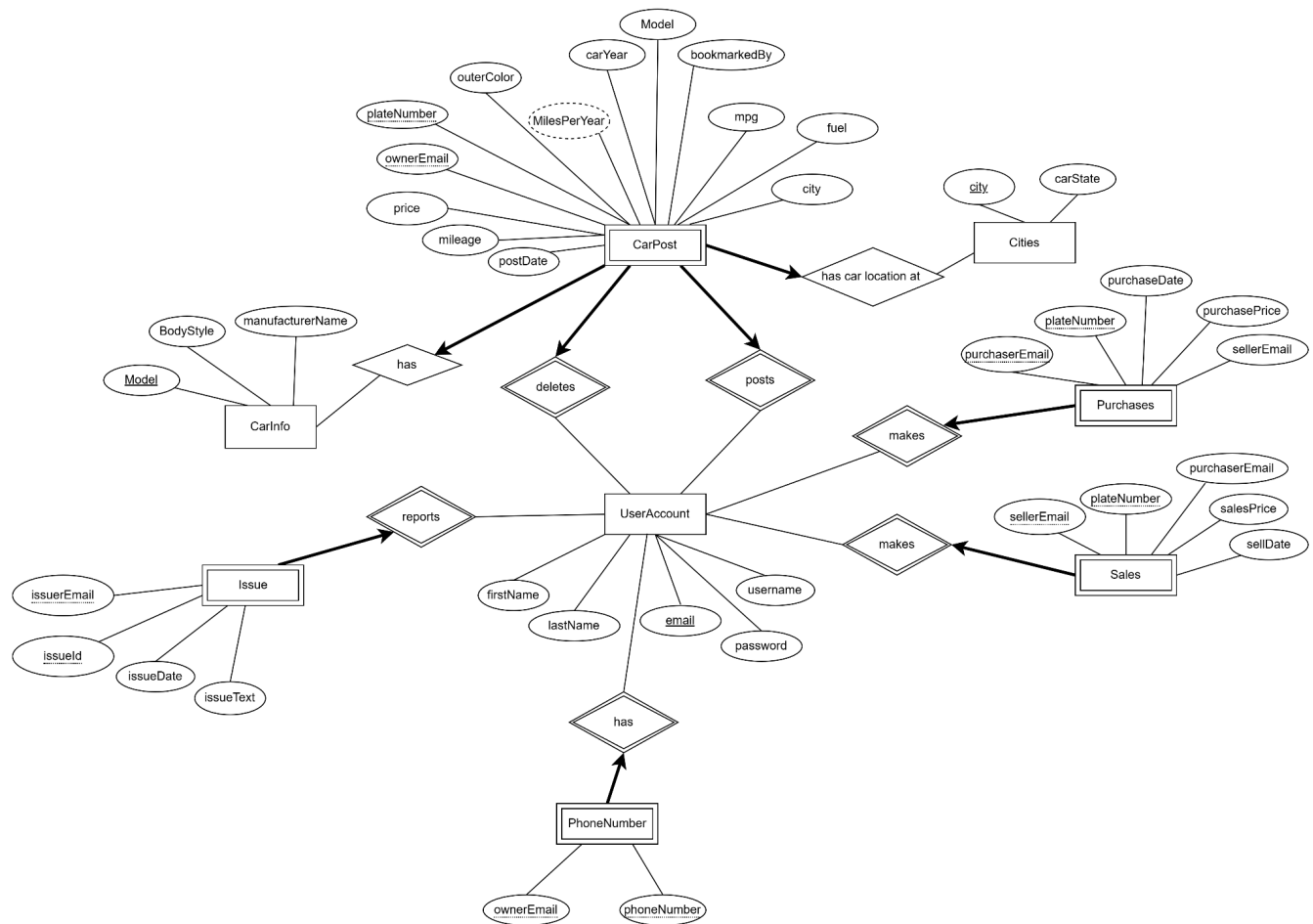
Architecture

Our web app's structure is straightforward and effective. We used HTML and CSS to create the user interface, making it visually appealing and easy to navigate. Python Flask handled the behind-the-scenes work, managing user requests and application logic. For storing and retrieving data, we turned to MySQL with the help of JayDeBeAPI, a Python library similar to JDBC.

Following our class instructions, we opted for a 2-tier architecture, skipping XML or JSON layers to keep things streamlined. One neat feature is how our user interface communicates directly with the database, speeding up access to information.



ER Data Model:



ER Design Considerations

BookmarkedPost Relation:

- Initially, we planned to have a BookmarkedPost relation, which would be a weak relation dependent on the UserAccount entity. However, we realized that it also depended on the CarPost relation, violating referential integrity. To rectify this, we opted to incorporate a "bookmarkedBy" attribute directly into the CarPost relation, ensuring both referential integrity and compliance with BCNF.

Sales Relation:

- The Sales relation functions similarly to the purchases relation, tracking general car information that has been sold.

Issue Relation:

- The Issue relation straightforwardly stores the issue text in String format along with the issuer email, providing a simple and intuitive solution.

Cities Relation:

- We maintain the Cities relation separately to adhere to BCNF, ensuring efficient data organization and management.

CarInfo Relation:

- CarInfo is kept separate from CarPost due to BCNF considerations. Since the car's body style and manufacturer names depend on the car model, they are included in the CarInfo relation.

PhoneNumber Relation:

- The PhoneNumber relation is kept distinct due to BCNF considerations. It's important to note that CarPost does not hold a phone number directly, as it depends solely on the owner's email. Additionally, users can have multiple phone numbers.

Important Note on "bookmarkedBy" Attribute:

The size of the "bookmarkedBy" attribute is set to VARCHAR(10000) to accommodate multiple user emails. While longtext could be used, it may impact query performance. Given our application's scale, we anticipate that 10,000 characters will be sufficient.

Assumptions:

Although it is true that a CarPost belongs to a UserAccount, we made the CarPost a strong entity. The reason is because CarPost is used almost all the time and making it a weak entity makes it very difficult for queries that access the CarPost relation. We made the ownerEmail to be Non-Null and added an ON DELETE CASCADE ForeignKey to alternatively keep the CarPost be dependent on the UserAccount.

A body style can belong to many cars.

A car has exactly one body style.

We will assume that car plate numbers are unique and use it as the PrimaryKey instead of VIN because it is more readable and shorter (makes queries efficient).

BodyStyle is a strong entity because it's a fact that a car model has a particular body type.

Referential Integrity Actions We Took: Majority of the entities such as bookmarked car posts, purchase history, sales history, and others are weak entities and are dependent on the existence of the UserAccount. This is because if a UserAccount is deleted, then all the other entities that belong to that UserAccount will also be deleted. This ensures consistency and no orphaned data is left.

Data Integrity Actions We Took: we set many constraints on UserAccount passwords, usernames, emails, and other attributes along with making all attributes NOT-NULL.

We normalized the CarPost relation into 3 separate relations: CarPost, CarInfo, and Cities. Whenever the same Model appears the same CarInfo follows and whenever the same zip code appears the same Cities follows, so we did a lossless decomposition of the CarPost relation such that all divided relations are in the BCNF form. For example: if the model is Camry, body style is Sedan, and manufacturer is Toyota. Another example: if the zipcode is 95112, the city is San Jose. More details are shown in the Normalization report.

Normalization process to the level of BCNF (PDF Report Below)

■ **CS157A_Toyota_Market_NormalizationReport (1).pdf**

Major Design Decisions

Our database does not include a car relation, as we are not a DMV website. In other words, the database does not need to track ownership information since our web app's primary function is to display available posts, not ownership status. Storing cars that a user owns but are not posted for sale on our website would be redundant and unnecessary, leading to inefficient storage. Therefore, we only retain posts currently available in our database to ensure data integrity.

This design choice eliminates the need for a car relation, which would otherwise require redundant storage. By focusing solely on storing posts and not car ownership information, our logic and design consistently uphold referential integrity and data integrity. Additionally, this approach avoids the illogical relationship between Car and CarPost, as Car stores car information while CarPost holds details such as car specifics and poster email. When a user account associated with a post is deleted, the corresponding car post is also removed, maintaining referential integrity.

Implementation Details:

We employed a variety of queries, including joins and nested queries, to handle user requests efficiently. Python Flask, combined with HTML, CSS, and JDBC, ensured the web app's speed and robustness. Indexing was applied to popular attributes to optimize query performance.

Demonstration

Login/Registration

157A Car Market

Login

Login

Signup

Register

Then, it will take you to the page of your bookmarked posts where you can filter depending which posts you're looking for (none so far)

Your bookmarked Cars

All Posts Your Posts **Bookmarks** Your Purchases Your Sales Your Car History Account Issues Logout

Color:

Minimum Miles:

Maximum Miles:

Minimum Price:

Maximum Price:

Energy Type:

Condition:

Zip Code:

City:

State:

Then, when you go to the all post tab you will see all the Toyota cars in the database where you can filter by color, year, miles, price, mpg, fuel type, and city. Also, you can buy or bookmark the car that you are looking for or interested in.

(Looking for a car that is electric, white, and year of 2020):

Viewing Car Posts - Hello user: kevin.durant@gmail.com!

[All Posts](#) [Your Posts](#) [Bookmarks](#) [Your Purchases](#) [Your Sales](#) [Your Car History](#) [Account](#) [Issues](#) [Logout](#)

Enter Car Name

Color: white

Year: 2020

Miles

Miles Min:

Miles Max:

Price

Price Min:

Price Max:

Energy Consumption Type: Electric

Mpg

Mpg Min:

Mpg Max:

City:

[Filter Posts](#)

owner0_98@gmail.com

Plate Number: PLNUM_0_98

Model: prius

Color: white

MPG: 19

Fuel: electric

Year: 2020

Mileage: 47726

City: Pembroke Pines

Price: 28635

Post Date: 2023-5-30

[Buy Car](#)

[Bookmark Car](#)

owner2_33@gmail.com

Plate Number: PLNUM_2_33

Model: corolla

Color: white

MPG: 20

Fuel: electric

Year: 2020

Mileage: 99855

City: El Paso

Price: 38860

Post Date: 2012-5-3

[Buy Car](#)

[Bookmark Car](#)

owner7_95@gmail.com

Plate Number: PLNUM_7_95

Model: fortuner

Color: white

MPG: 18

Fuel: electric

Year: 2020

Mileage: 149907

City: Oakland

Price: 31745

Post Date: 2012-8-18

[Buy Car](#)

[Bookmark Car](#)

After buying you will be redirected to the Your Purchases tab which shows all the cars you have bought.

Your purchase history

[All Posts](#) [Your Posts](#) [Bookmarks](#) [Your Purchases](#) [Your Sales](#) [Your Car History](#) [Account](#) [Issues](#) [Logout](#)

Enter Plate Number

Price

Price Min:

Price Max:

Date:

[Filter Posts](#)

Plate Number: PLNUM_0_98

Seller: owner0_98@gmail.com

Car sold on: 2024-05-10

Car sold for: \$28635

Or if you bookmark a post and click on the Bookmarks tab it will show all the cars you have bookmarked where you can buy or remove posts.

Your bookmarked Cars

[All Posts](#) [Your Posts](#) [Bookmarks](#) [Your Purchases](#) [Your Sales](#) [Your Car History](#) [Account](#) [Issues](#) [Logout](#)

Color:

Minimum Miles:

Maximum Miles:

Minimum Price:

Maximum Price:

Energy Type:

Condition:

Zip Code:

City:

State:

owner2_33@gmail.com
Plate Number: PLNUM_2_33
Model: corolla
Color: white
MPG: 20
Fuel: electric
Year: 2020
Mileage: 99855
City: El Paso
Price: 38860
Post Date: 2012-5-3

When you click on the Your Car History tab, it will show you all the cars that you've owned and currently owning

Your Car History

[All Posts](#) [Your Posts](#) [Bookmarks](#) [Your Purchases](#) [Your Sales](#) [Your Car History](#) [Account](#) [Issues](#) [Logout](#)

Minimum Price:

Maximum Price:

Cars you own

Plate Number:
PLNUM_0_98
Car Price: \$28635

Cars you've owned

And you can also report issues or make changes to the account.

Report an issue

Issue reported by: kevin.durant@gmail.com
Issue number: 2

Date

mm/dd/yyyy

Write issue details here

Send Issue

Go back to Index

Edit Profile

Current email: kevin.durant@gmail.com

Please make your password length ≥ 8

Enter a new password

Confirm new password

Save changes

Delete Account

Go back to Index

Now let's see the owner's car that kevin.durant@gmail.com bought from. As you can see in this user Your Sales page shows who bought the car they have posted.

Your sale history

All Posts Your Posts Bookmarks Your Purchases Your Sales Your Car History Account Issues Logout

Enter Plate Number Search

Minimum Price:
Min price...
Maximum Price:
Max price...
Filter Posts

Plate Number:
PLNUM_0_98
Car sold to: kevin.durant@gmail.com
Car sold on: 2024-05-10
Car sold for: \$28635

When the user go to Your Car History tab, they will see the cars they currently own and owned in the past.

Your Car History

All Posts Your Posts Bookmarks Your Purchases Your Sales Your Car History Account Issues Logout

Enter Plate Number Search

Minimum Price:
Min price...
Maximum Price:
Max price...
Filter Posts

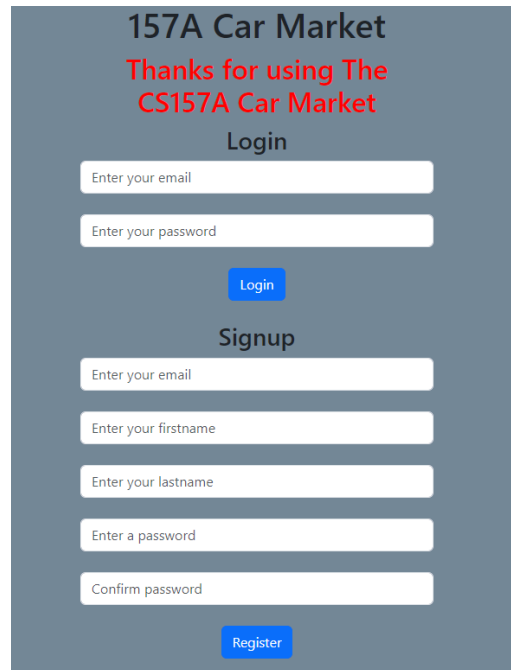
Cars you own

Plate Number:
PLNUM_0_26
Car Price: \$20391

Cars you've owned

Plate number:
PLNUM_0_14
Car sold to: kevin.durant@gmail.com
Car sold on: 2024-05-10
Car sold for: \$10665

It will take you back to the login/register page when you click log out.



The image shows a web form for '157A Car Market'. At the top, it says '157A Car Market' in bold black text, followed by 'Thanks for using The CS157A Car Market' in red text. Below this is a 'Login' section with two input fields: 'Enter your email' and 'Enter your password', and a blue 'Login' button. Underneath is a 'Signup' section with five input fields: 'Enter your email', 'Enter your firstname', 'Enter your lastname', 'Enter a password', and 'Confirm password', followed by a blue 'Register' button.

Conclusion:

As a result, we were able to make a website where you can buy and sell Toyota cars easily, kind of like Craigslist. We made sure everything stayed organized and searches ran smoothly so you could find what you wanted without any trouble. Along the way, we learned a bunch about how to organize data, keep things running smoothly, and make the site easy to use. This will definitely help us make even better websites in the future. We've got some ideas for making our site even better, like adding more ways to search for cars, including different car brands, letting users give feedback, and making sure the site can handle lots of people using it at once. These changes would make the site even more helpful and fun to use.