Ride Along

(High-Level Design)

Team Specs

Jesus Cerda (Lead)

Jason Barber

Giovanni Contreras

Rainier Marlone Getuaban

Vi Nguyen

November 22, 2023

Changelog v0.0 **3**
Overview **4**
Objectives 4
Project Scope 4
Purpose **4**
Decision Tree **4**
High Level Design Architecture **6**
Product Architecture Overview 6
Cross-cutting concerns 6
Logging 6
Error Handling 7
Security 8
UI Layer 8
Web service/framework Layer 8
Manager Layer 8
Service Layer 8
Data Access Layer 8
Data Store Layer 9
Work Cited **10**

Changelog v0.3

Version Number	Date of Revision	Summary of Revisions
v0.0	November 08, 2023	● Initial creation of high level design document
V0.1	November 17, 2023	● Revision due to Client feedback ○ Added works cited page
v0.2	November 22. 2023	● Consolidated cross cutting concerns into Layers ● Updated diagram ● Added advantages and limitations

Overview

Objectives

- The goal of our product is to create a web application that users can use to keep track of the value of their vehicle and maintain their vehicle's value as high as possible. The value of their vehicle is determined by the vehicle market along with how well their vehicle is maintained and any modifications done to their vehicle.
- Users can register within our app, allowing them to create, edit, and transfer vehicle profiles.

Project Scope

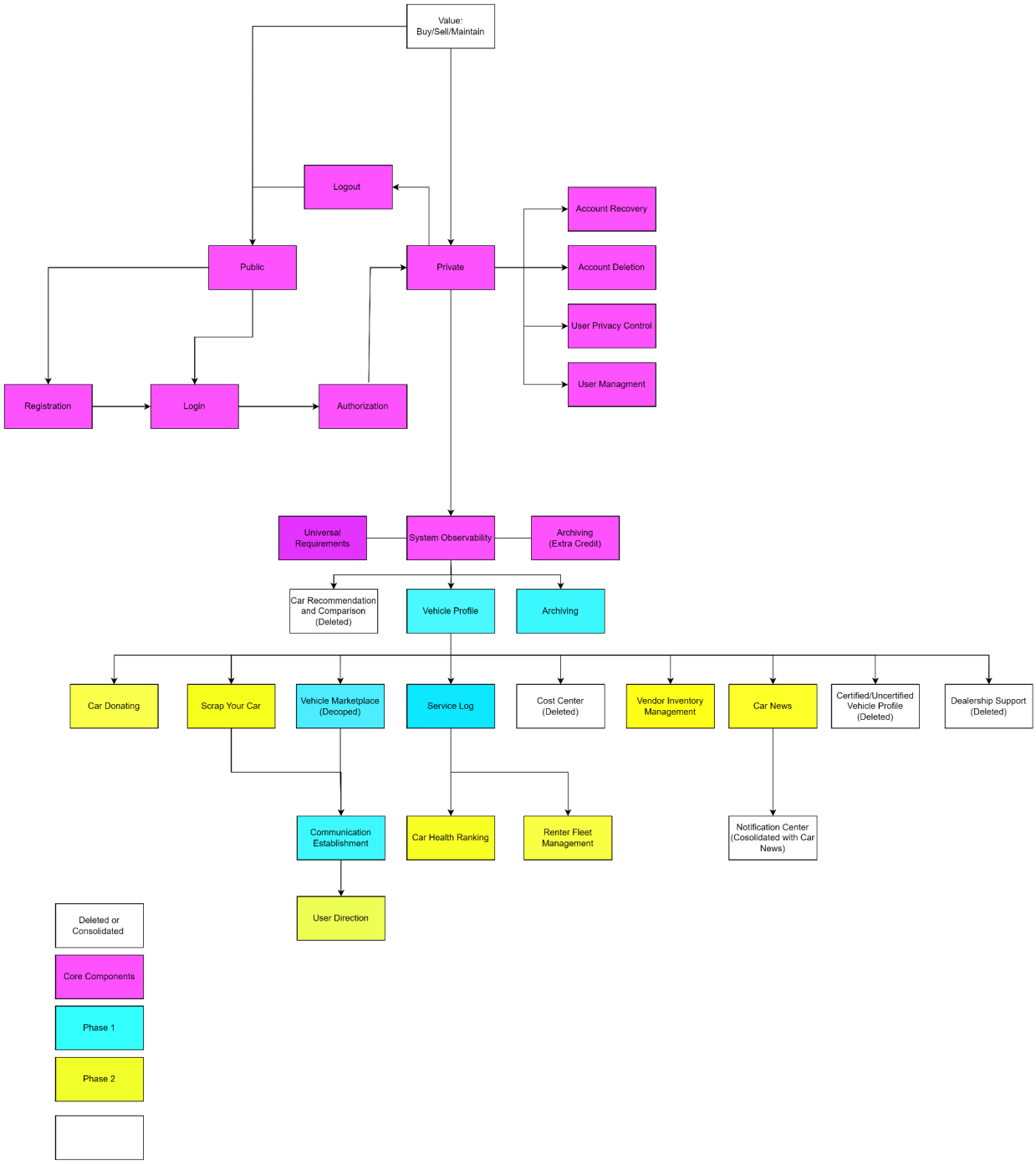
Ride Along will be a SPA web application compatible with the latest desktop Chrome version 118.0.5993.88 for Windows, macOS, and Linux, android Chrome version 118.0.5993.80, and iOS Chrome Version 118.0.5993.92 as of October 18, 2023. In addition, we will support standard display resolution of (1920 x 1080) and tablet and phone display resolutions. The application will only be available within Los Angeles and Orange County, using USD as currency and the imperial system for measurement, as of release the application will support Pacific Standard Time (UTC−08:00) and Pacific Daylight Time (UTC−07:00) while the default language will be American English (en-us). The application only supports vehicles newer than 1992 due to the limitations of some APIs. Users will have to be 18 or older in order to sign up for our application.

Purpose

The High Level Design document serves as an overview of the Ride Along web application architecture to provide a guide to when beginning to design our product. In addition, while also serving as an overview of the product's overall architecture, the document will describe abstractions and flow of control for cross cutting concerns such as logging, security, error handling, and data storage.

Decision Tree

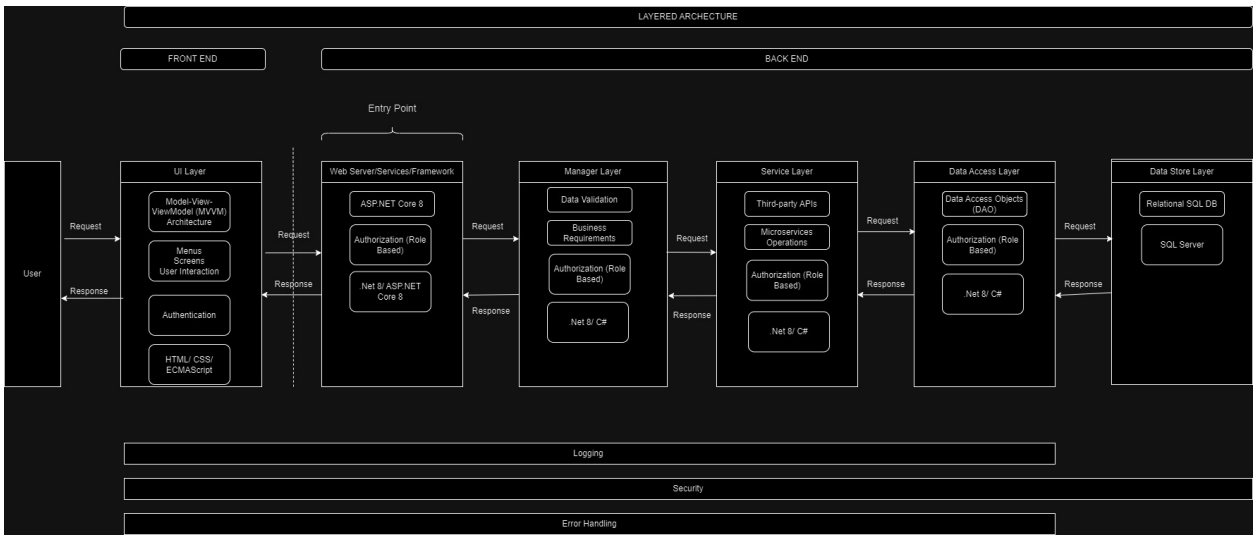
The project's decision tree shows the level of impact and dependencies that features have on each other. Color purple is the cross-cutting concern which will be the dependencies of all application-specific features, hence, these components will be delivered first if not at the same time as some of phase 1 features. Color blue are chosen phase 1 features, these features are dependent on the core components hence will be delivered after the completion of those components. Color yellow is chosen for phase 2 features, "Car Health Rating", "Rental/Fleet Management" and "User Directions" features are dependent on phase 1 features, hence the delivery of these phase 2 features if possible will take place after phase 1.


Value: Buy/Sell/Maintain
Logout
Account Recovery
Public
Private
Account Deletion
User Privacy Control
User Managment
Registration
Login
Authorization
Universal Requirements
System Observability
Archiving (Extra Credit)
Car Recommendation and Comparison (Deleted)
Vehicle Profile
Archiving
Car Donating
Scrap Your Car
Vehicle Marketplace (Decoped)
Service Log
Cost Center (Deleted)
Vendor Inventory Management
Car News
Certified/Uncertified Vehicle Profile (Deleted)
Dealership Support (Deleted)
Communication Establishment
Car Health Ranking
Renter Fleet Management
Notification Center (Cosolidated with Car News)
User Direction
Deleted or Consolidated
Core Components
Phase 1
Phase 2

High Level Design Architecture

Product Architecture Overview

This project will utilize a layered architecture with a total of 6 layers in total while for our Front - End of our product, we will be utilizing a MVVM design pattern.. The layers include: UI, Web Service/ Framework, Manager, Services, Data Access, and Data Store. Each layer will perform a specific role within our application. By taking on a layered architecture design, our application allows for easier testing, more flexibility, and allows us to implement security at every layer though on the downside allowing for potentially low scalability and performance.



Cross-cutting concerns

UI Layer

This is the front end of the application, from which users will be able to interact with different UI components corresponding to different functionalities of the application and display any results received from the application.

- **Security:** A user must be authenticated and authorized to interact with the UI layer.
    - If user isn't authorized to interact with certain feature, show a general message saying permission denied
    - If a user isn't authenticated, show a general message saying authentication failed
- **Logging:**
    - All successful and failed authentications will be logged with relevant information
    - All successful and failed authorizations will be logged with relevant information
- **Error Handling:**
    - Failed authentication
        - When a user attempts to login but does not have a valid or existing account, display a general message to re enter credentials or to create a account

Web service/framework Layer

This layer will be responsible for processing requests and delivering them to the manager layer as well as sending any HTTP response back to the front end.

- **Security:** To send any requests to the manager layer or responses to the UI layer, a user must be authorized and authenticated.
- **Logging:**
    - All HTTP response status codes will be logged
        - Codes can range from 100 - 599
    - All requests from the UI Layer that fail will be logged
    - All requests to Manager layer that fail will be logged
- **Error Handling:**
    - HTTP Fails
        - When the HTTP fails to send a code, a message will be generated saying to try again later

Manager Layer

This layer will be responsible for taking the request from the web framework layer and determining if it violates any business constraint that we have on any operations or user's input as that are stated in our Business Requirements Document (BRD).

- **Data Validation:** This layer will check the user's input according to any constraint and business rules that our product has.
- **Security:** Users will have to be authorized and authenticated to access services. If a user isn't authorized for a service a message will be generated saying detailing authorization error
- **Logging:**
    - All successful and failed requests to use a service will be logged
    - All failed communication requests to service layer will be logged
- **Error Handling:**
    - Violation of a Business Rule
        - When a user or the application violates a business rule, the application will flag the error and display a general message for the user. These violations will also be logged for the developer.
    - Failed Request to Service Layer
        - When a request fails to request a service, a message will be displayed saying that the service is unavailable

Service Layer

This layer contains our microservices required for our application, it will process validated data that is sent from the manager layer and processes the data or request received in a particular way.

- **Security:** In order to access any services related to this layer or requesting any information from a third-party API, a user must be authorized and authenticated to perform that particular service.
- **APIs:** This layer will also be requesting data from third-party APIs if necessary, process the response and then send those information back to user in the UI layer
- **Data Validation:** Data coming back from third-party API will also be validated to ensure the accuracy of the information provided by our service
- **Logging:**

    - Any services that are fail to be used will be logged
    - Any service that is successful will be logged
- **Error Handling:**
    - Invalid API authentication
        - Application will try to connect to other available APIs if available, if all fail or none are available to user user will be displayed a message to return later.
    - Unable to access services
        - If a user is unable to use a service or the service fails, the user will be prompted to try again later

Data Access Layer

This layer will process any request from the user that needs to access information from our datastore. We will be utilizing Data Access Objects in order to interact with our database, and any request passed through this layer will be authenticated to confirm the sender's identity and any valid user session they have associated with it.

- **Security:** Any requests sent by a user will be checked to see if they are authenticated and if they are authorized to perform the action based on the service they are attempting such as reading or writing to a database.
- **Logging:**
    - Log all unauthorized attempts to read or write to data store
    - If the data storage is full, a log will be generated
- **Error Handling:**
    - Invalid Server Request
        - If a invalid server request occurs, the application will display the error code and log the error.

Data Store Layer

This is the storage for all the data we collected for the application, this layer will carry out any valid queries from the data access layer whether for modifying or just reading from the data store.

- **Security:** In order to access the data store, the DAO will have particular permissions to write or read from the data store.
- **Error Handling:**
    - Server timeout
        - If the server has a timeout, the user will be displayed a message to try the feature again at a later time. In addition, the error will be logged for the developer.
    - Invalid Database Access Token
        - The DAO won't be able to access the Database and be sent a error message accordingly
    - Empty Database search
        - If a search in the database returns nothing for a search, when multiple results are expected, the data store will return 0.

Tech Stack

HTML

- Advantages
    - HTML is supported by all web browsers and thus is universally compatible
    - Able to integrate with CSS and JavaScript
- Limitations
    - Relies on integration of CSS to handle presentation and Javascript for functionality of the web page
    -

CSS

- Advantages
    - Enhances presentation and content separation of HTML elements
    - Able to integrate with HTML
- Limitations
    - Issues with browser compatibility

JavaScript

- Advantages
    - More flexible since its a dynamically typed language
    - Able to integrate with HTML and JavaScript
- Limitations
    - Issues with browser compatibility
    - Has security vulnerabilities

.Net 8

- Advantages
    - Newest Version of .Net
    - Will have long term support
    - Includes security features such as Code Access Security (CAS)
- Limitations
    - Recently released thus some bugs may be present

ASP.Net Core 8

- Advantages
    - Newest ASP.Net Core version
    - Will have long term support
    - Robust security features
    - Cross platform migration
    - Allows for separation of concerns
- Limitations
    - Difficult documentation

        - Other frameworks like node.js are more easier to learn and use
            - More scalable

C#

- Advantages
    - Integrates well with .NET 8 and is fully supported by it
    - High scalability
    - Is object oriented promoting efficiency and flexibility\
    - Use .Net libraries
- Limitations
    - Slower code execution
    - Dependent on .Net platform

SQL Server 2019

- Advantages
    - Long term support
    - Advanced firewall and thus secure
    - Free cost
- Limitations
    - Works well with Window users only and if you are planning to use a non-Windows platform some features may not be supported

IIS 10+

- Advantages
    - Supports ASP.Net Core
    - Security includes many features such as SSL support
- Limitations
    - Designed for strictly Windows environments

Work Cited

- Layered architecture - cheriton school of computer science. (n.d.). https://cs.uwaterloo.ca/~m2nagapp/courses/CS446/1195/Arch_Design_Activity/Layered.pdf
- O'Reilly Media, Inc. (n.d.). *Software architecture patterns*. O'Reilly Online Learning. https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html