Status Summary:

Project Name - Car Rental System
Team Members - Aayushman Bhat, Saksham Sahai Srivastava

Work Done and Patterns:

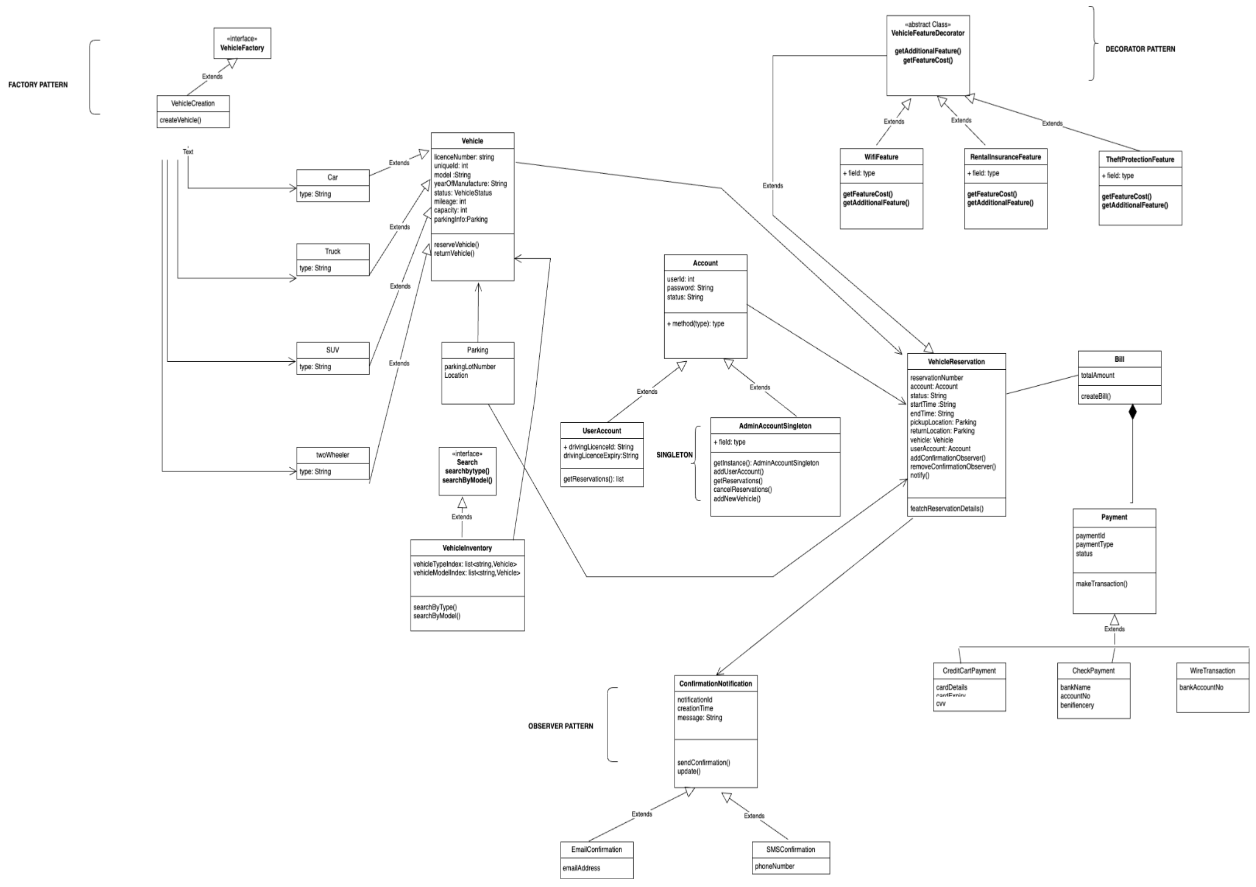
As of now, the Vehicle Reservation System has been expanded to incorporate the Factory Pattern for creating vehicles and the Observer Pattern to send reservation notifications. Below is an overview of the overall work done:
Vehicle Reservation System with Factory and Observer Patterns
1. Factory Pattern Implementation:
   - Introduced a Factory Pattern for creating instances of Vehicle objects.
   - Created a VehicleFactory class responsible for creating vehicles based on specified parameters.
   - The Vehicle class includes attributes such as license plate, type, make, and model.
   - The factory pattern enhances flexibility and scalability by centralizing vehicle creation logic.
2. Vehicle Search and Reservation:
   - Users can search for vehicles based on type, view search results, and proceed to reservation.
   - The reservation process involves inputting details such as license number, start time, and end time.
   - The system generates a unique reservation number for each reservation.
3. Observer Pattern Implementation:
   - Implemented the Observer Pattern to facilitate reservation notifications.
   - Introduced an Observer interface with an update method.
   - Created a ReservationObserver class that implements the Observer interface.
   - The ReservationObserver class is responsible for handling reservation notifications.
   - The Vehicle class now maintains a list of observers, and the reservation process notifies them.
   -
4. GUI Enhancements:
   - Extended the existing GUI to incorporate adding new Vehicles and Seaching the vehicle using Vehicle Type for reservation . Utilized the Factory Pattern to create instances of Vehicle objects in response to user input.
   - Implemented the Observer Pattern to notify observers (e.g., reservation observers) of the reservation process.

Changes and Issues encountered:

Database Integration:

- While planning the overall design in the initial stages, we thought of using some kind of caching to store the data but as we proceeded the need for a well defined database schema was necessary.
- The system may benefit from integrating a database for persistent data storage. So we had to change from a redis based database to a RDBMS based database.

Class Diagram:



Plan for next iteration:

- Decorator Pattern for Special Features:
    - Implement the Decorator Pattern to add special features (WiFi, Theft Insurance, Premium Rental Insurance) to vehicles dynamically during the reservation process.

- User Account and Admin Account Creation and authentication:
    - Develop user authentication with a User class for regular users and an Admin class implemented as a Singleton for administrative tasks. Include user registration and login functionalities.
- Payment Class Hierarchy:
    - Create a class hierarchy for payments with options like credit card, PayPal, etc. Allow users to select a payment option during the reservation process.
- Integrated GUI:
    - Combine all GUI screens into a unified interface.
    - Implement an admin control panel accessible with admin credentials.
    - Dynamically adjust the GUI based on user roles (admin or user).
    - Modify the reservation process to include options for special features, payment methods, and user authentication.