CSC 470 Team Project

Car Rental System

Project Submission 2

Aniyah Little, Jamison Williams, Pamela Ngadjeu, Kalsoom Bibi, Angelena Lloyd

Table of Contents:

**Title Page****1**

**Contents Page****2**

**Architectural Design****3**

**CRS System in call and return Style****4-5**

**UML Communication Diagram(Staff makes reservation, Customer Returns car****5-8**

**Elaborated design Class diagram(Addition of attributes and methods for each class)****9**

**C**[**omponent-level design**](https://docs.google.com/document/d/1YeuRPwxL7w3g-bfmHrrvqV2G3OVXZYQhA_Xs9NKvY60/edit#heading=h.1rpsrn5t66sw) **10**

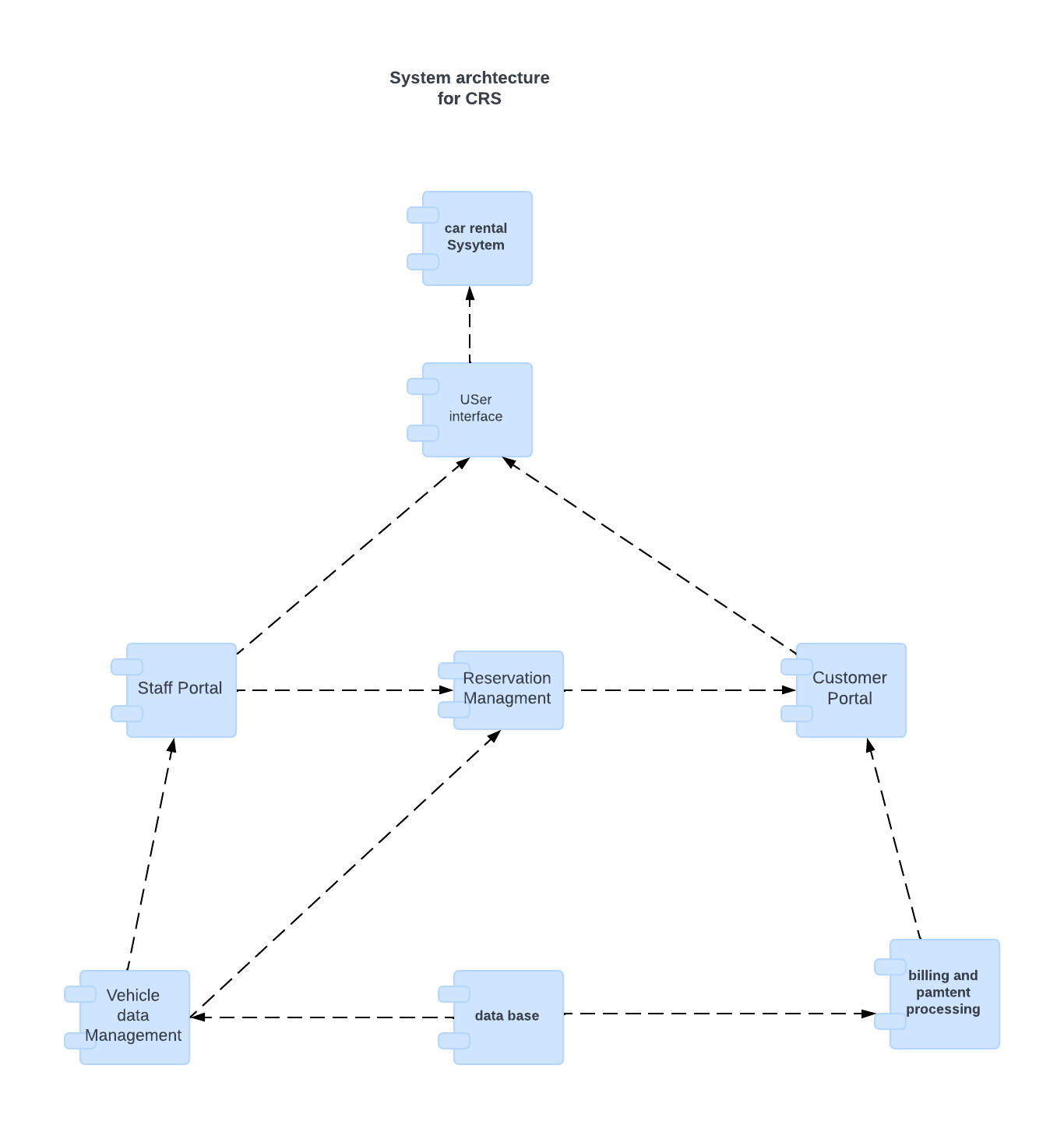
**UI Design For Each Use Case****11**

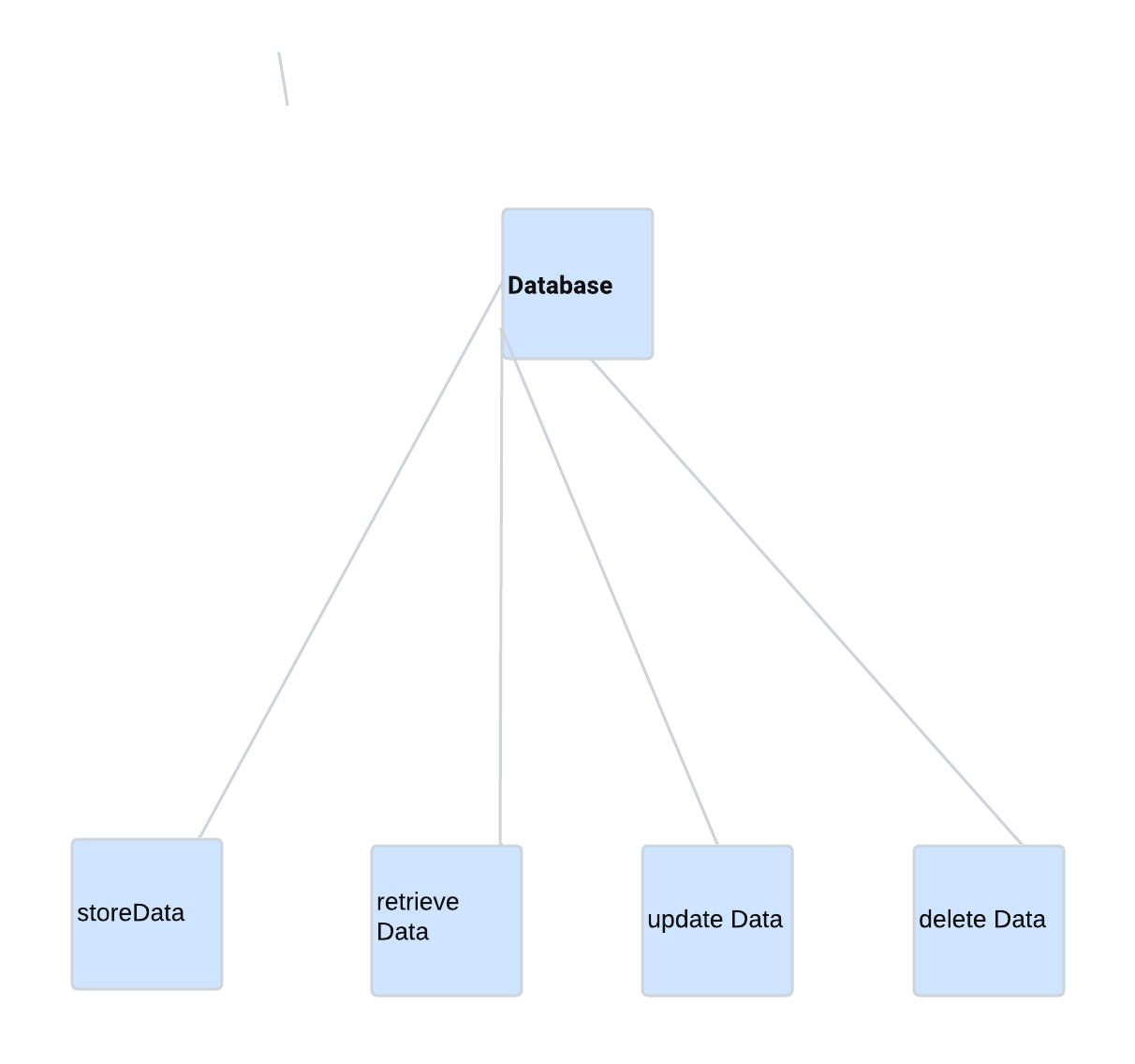
**Review Checklist****12**

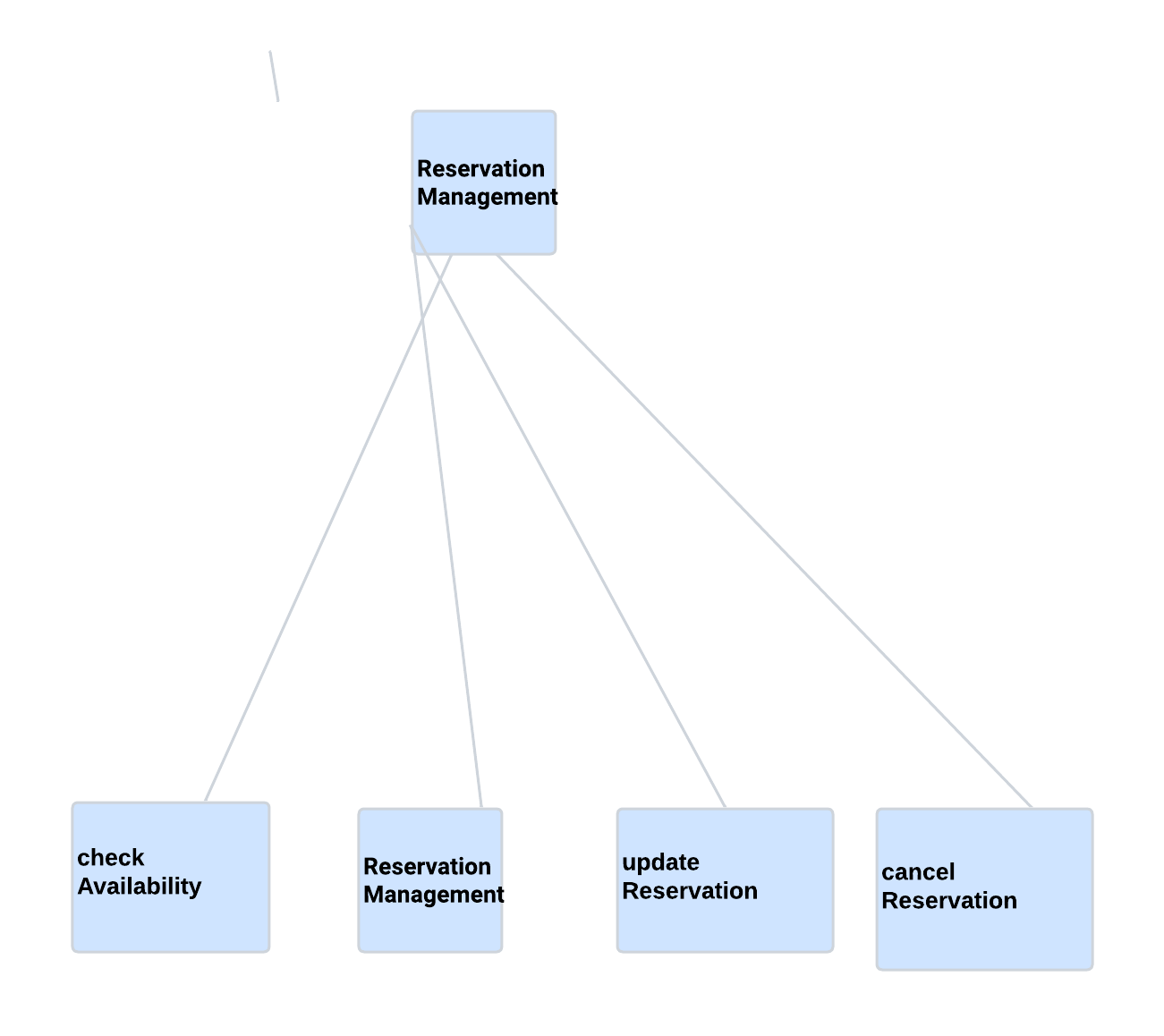
**Task Assignments****13**

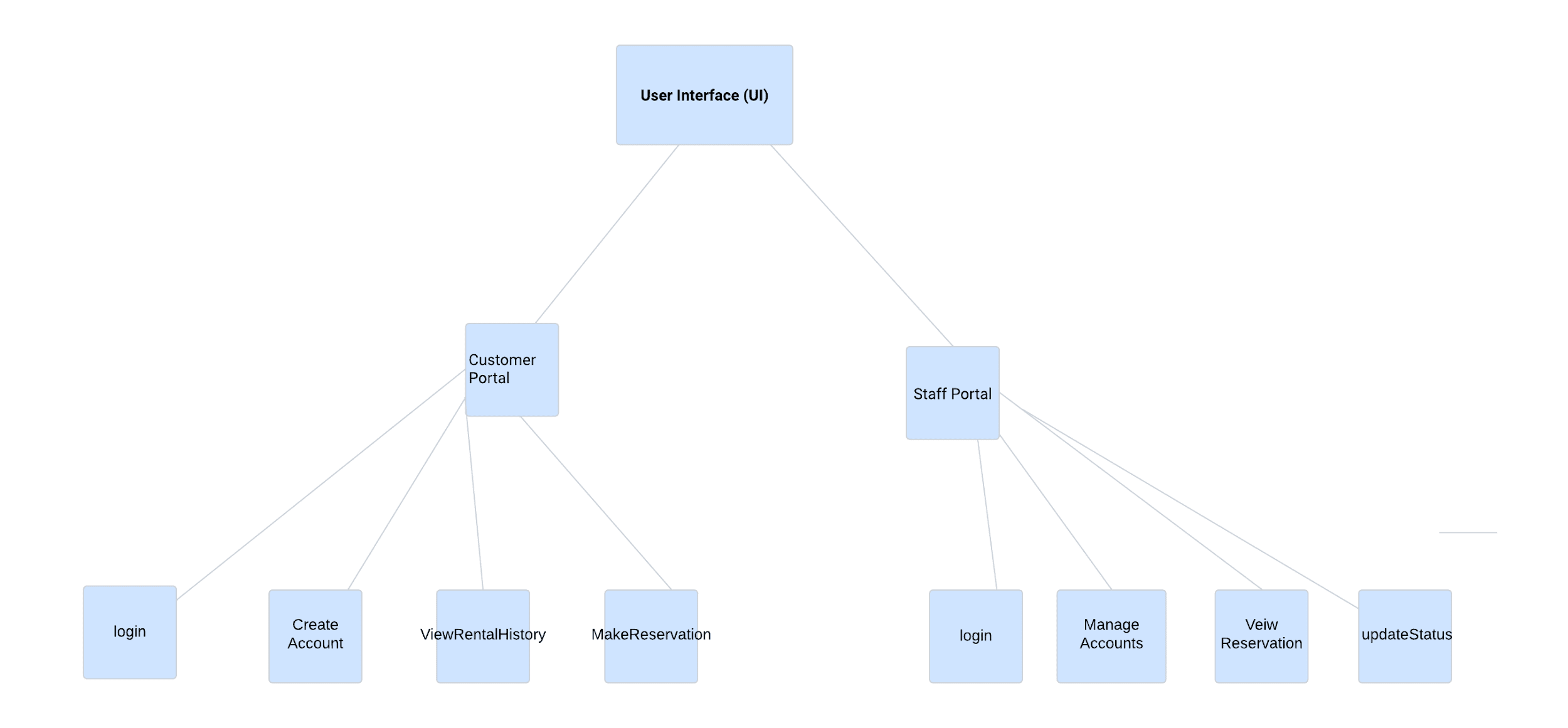
**Meeting Logs****14-15**

**Project requirements(Pasted from Canvas for reference)****16-17**

****

****

****

****

### 

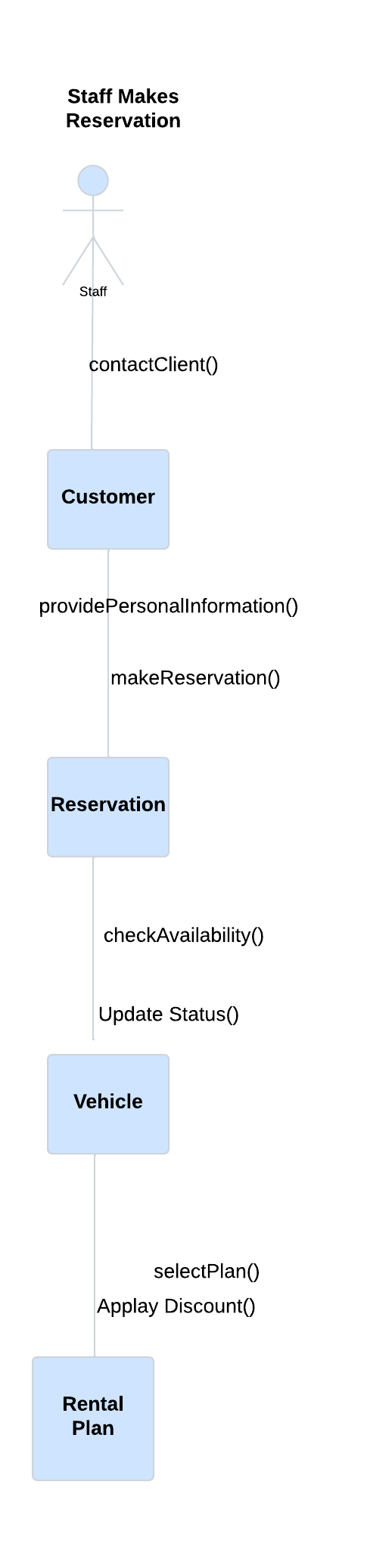
### **Justifications for Architectural Decisions**

For the Rental Car System, each top-level component manages specific business logic to enhance system maintainability and scalability. By isolating the user interface and core functional areas, such as billing, reservation, and inventory management, the architecture maintains clean and organized interactions. A centralized database ensures data consistency across the system, allowing all components to access and update records efficiently. This modular structure accommodates growth, making it straightforward to incorporate additional features as business needs evolve.

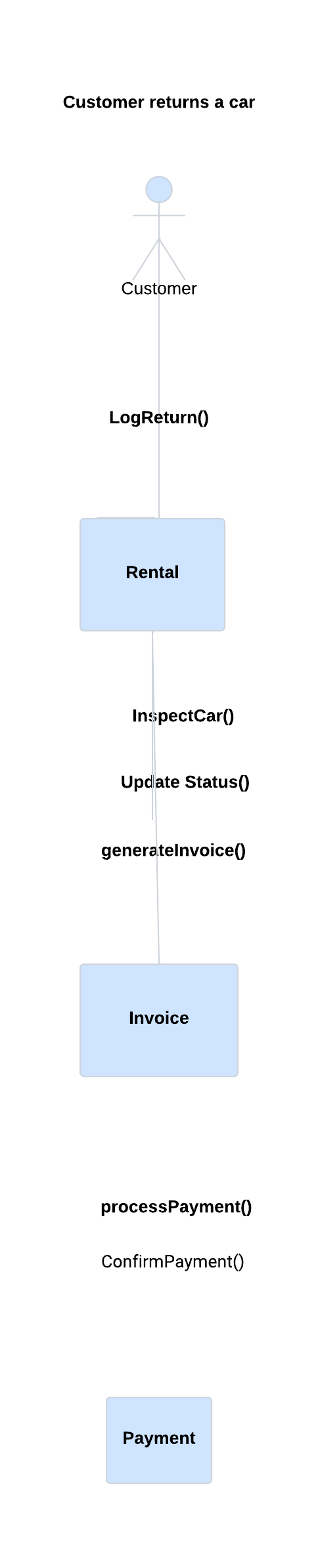
The user interface serves both staff and customer portals, providing an accessible, user-friendly interface for managing reservations, returns, and billing. Reservation Management handles all reservation processes, from creation to cancellation, ensuring efficient booking operations that integrate seamlessly with other system modules, including inventory and billing. Inventory and Vehicle Management oversee all vehicle-related data, including availability and updates to the rental fleet, allowing real-time tracking of vehicle status and supporting accurate availability checks.

The Customer Portal displays essential customer information, including rental history, which enhances user experience by offering straightforward access to account details. Billing and Payment Processing oversees invoicing, payments, and other financial transactions, processing payments securely and ensuring the accurate recording of financial data. Similarly the staff portal in the Rental Car System allows Staff to manage various operational tasks within the system. Database Management provides centralized storage for all system information, covering vehicles, customers, and payments, thus ensuring data integrity and supporting efficient data retrieval across the system.

* UML Communication diagram for staff makes reservation



* UML Communication Diagram Customer returns a car

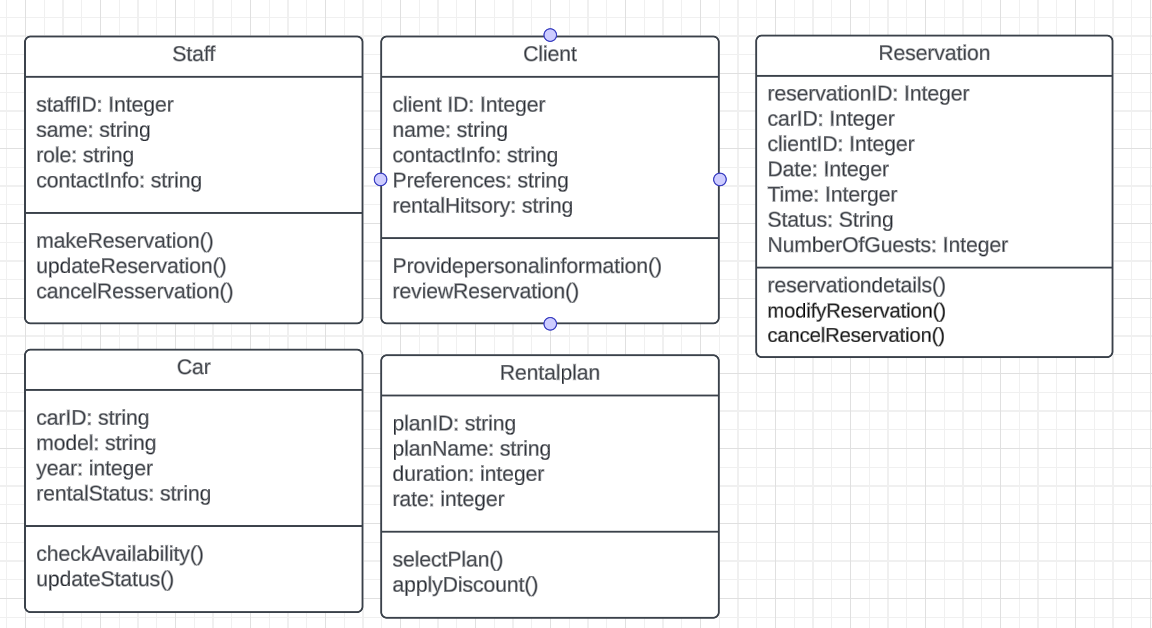


* Elaborated Class Diagram

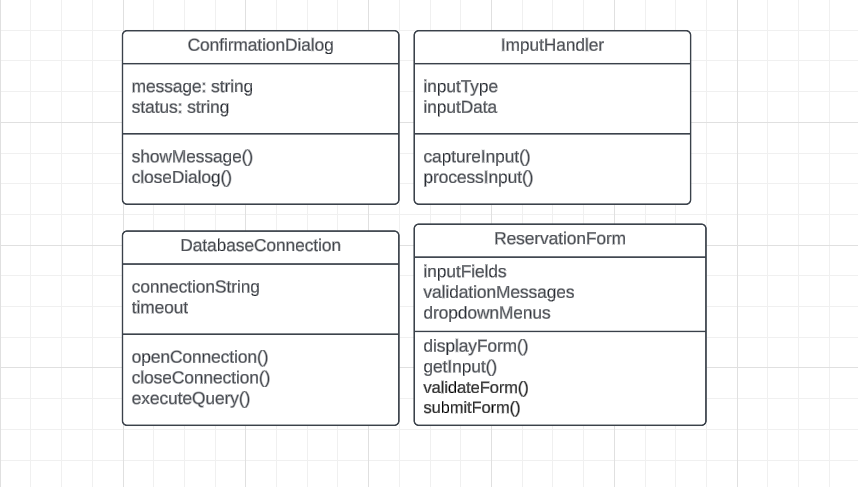


Use case1: Staff makes a reservation for a client.

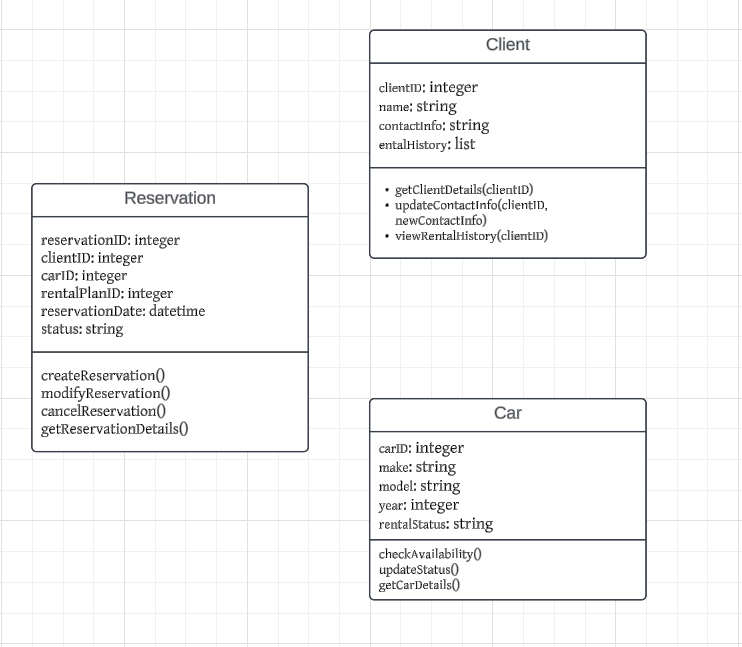
Step 1. Identify all design classes that correspond to the problem domain.



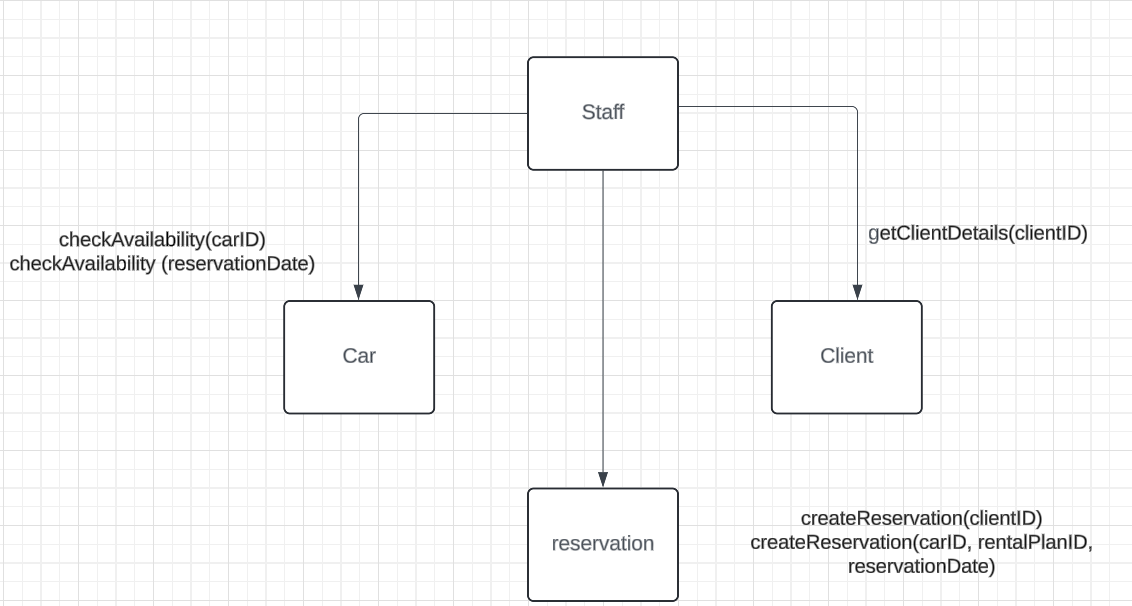
Step 2. Identify all design classes that correspond to the infrastructure domain.



Step 3.Elaborate all design classes that are not acquired as reusable components.



Step 3a. Specify message details when classes or components

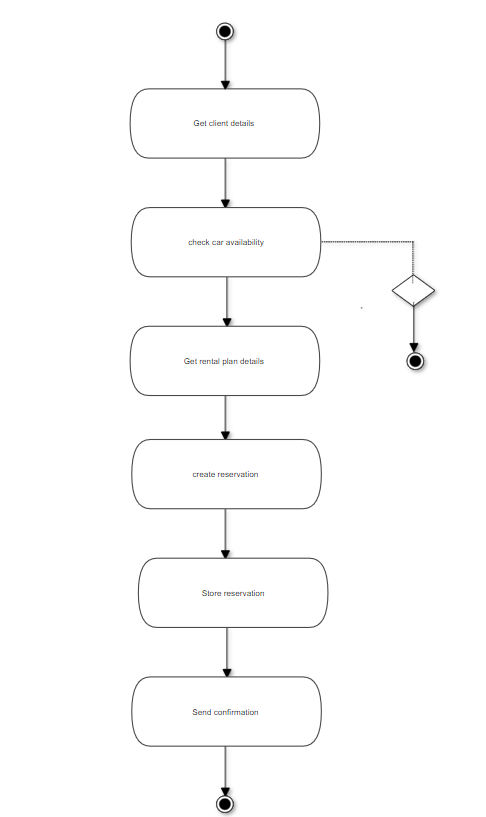


Step 3b. Identify appropriate interfaces for each component.

Step 3c. Elaborate attributes and define data types and data structures required to implement them.

| **Attribute** | **Class** | **Data Type/ Data Structure** |
| --- | --- | --- |
| staffID, name, role, contactInfo | Staff | Int, string, string, string |
| returnID, returnDate, condition | RentalReturn | Int, datetime, string |
| clientID, name, contactInfo, rentalHistory | Client | Int, string, string, list. |
| carID, model, rentalStatus, year | Car | int , string, string, int |
| invoiceID, amountDue, dueDate, status | Invoice | Int, float, datetime, string |
| paymentID, method, amount. | Payment | Int, string, float |
| reservationID, reservationDate  status | Reservation | Int, datetime, string. |

Step 3d. Describe processing flow within each operation in detail.



Step 4. Describe persistent data sources( databases and files) and identify the classes required to manage them.

Data Source:

ClientDatabaseConnection

Cardatabase

Reservationdatabase

Step 5. Develop and elaborate behavioral representations for a class or component.

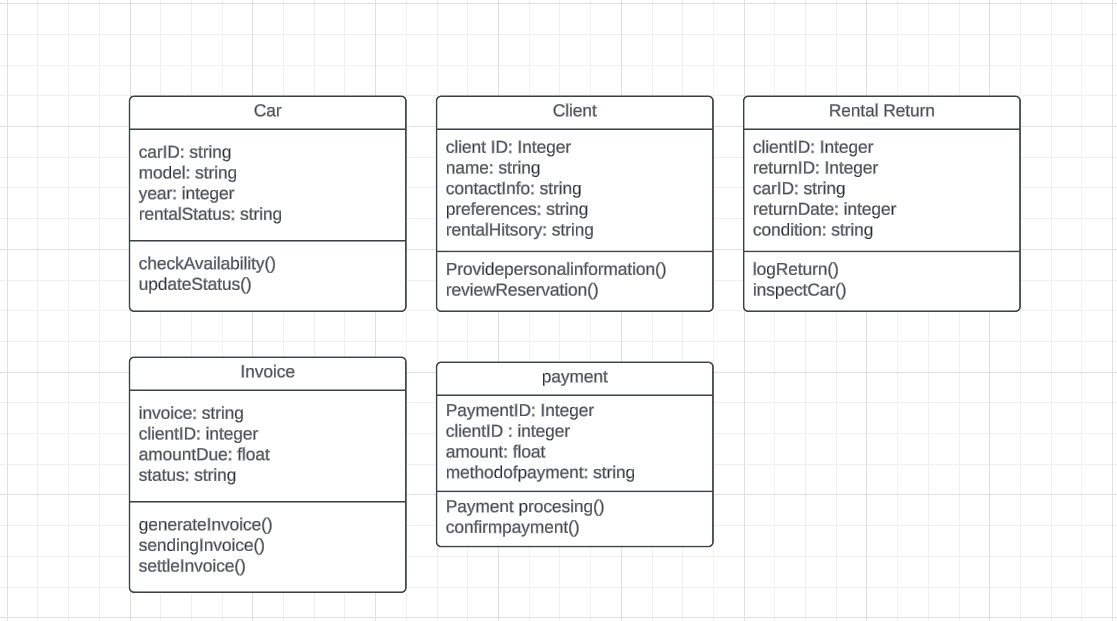
Step 5. Develop and elaborate behavioral representations for a class or component.

Step 6. Elaborate deployment diagrams to provide additional implementation details

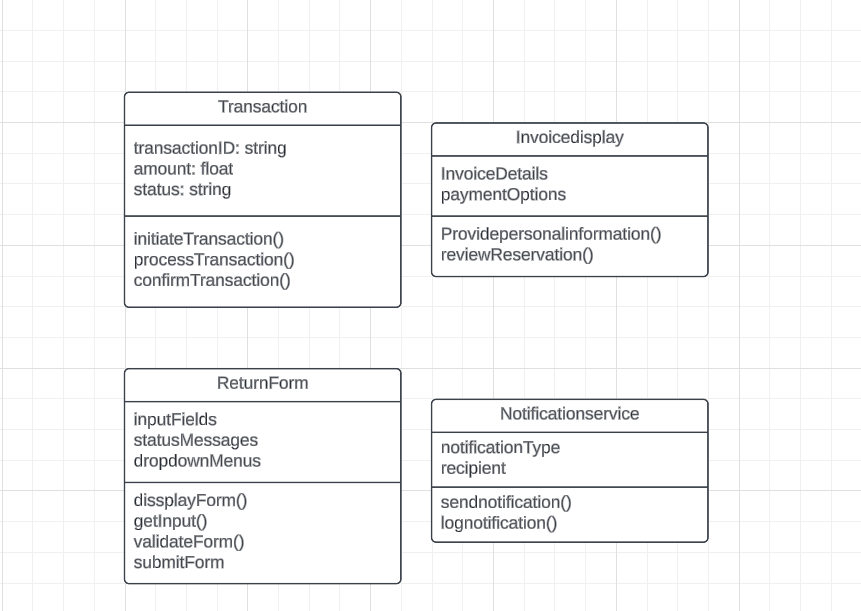
Step 7. Refactor every component-level representation and always consider

Use case2: A client returns the rental car

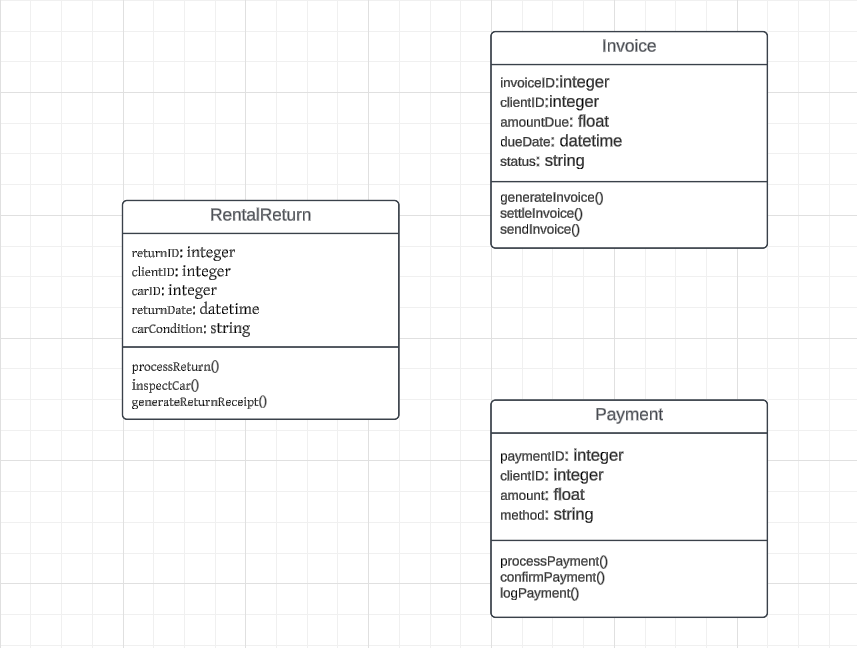
Step 1. Identify all design classes that correspond to the problem domain.



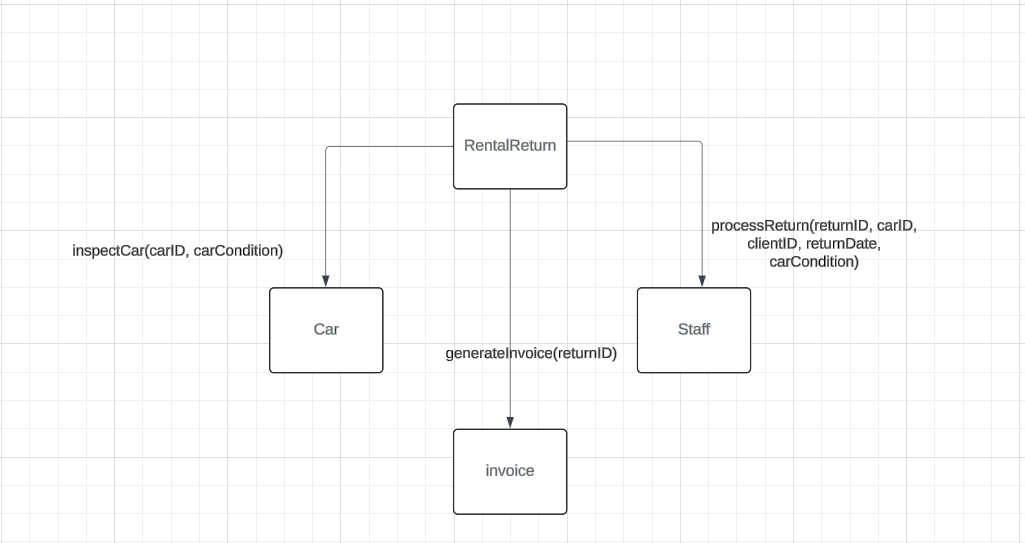
Step 2. Identify all design classes that correspond to the infrastructure domain.



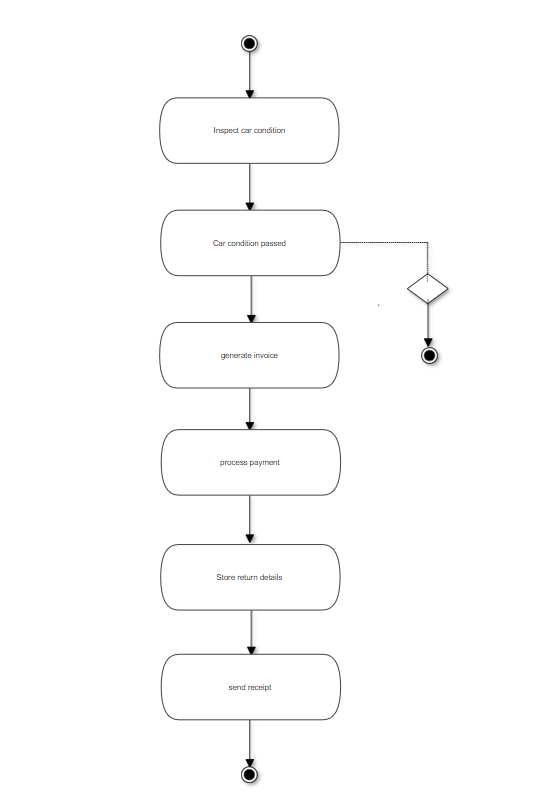
Step 3.Elaborate all design classes that are not acquired as reusable components.

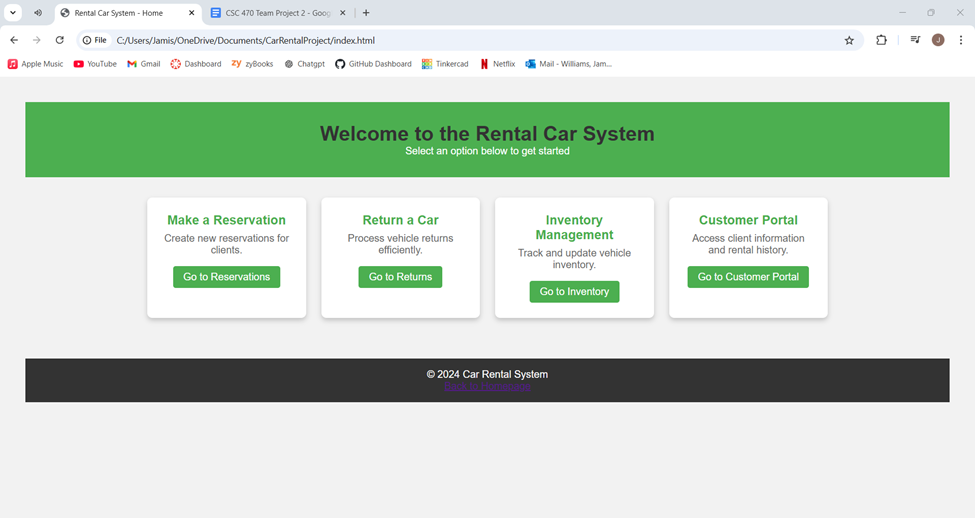


Step 3a. Specify message details when classes or components



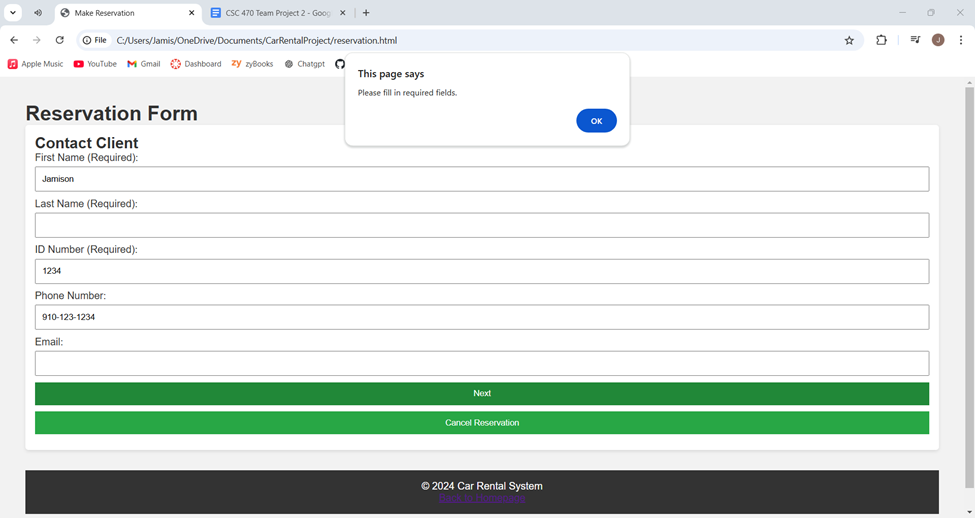
Step 3d. Describe processing flow within each operation in detail.



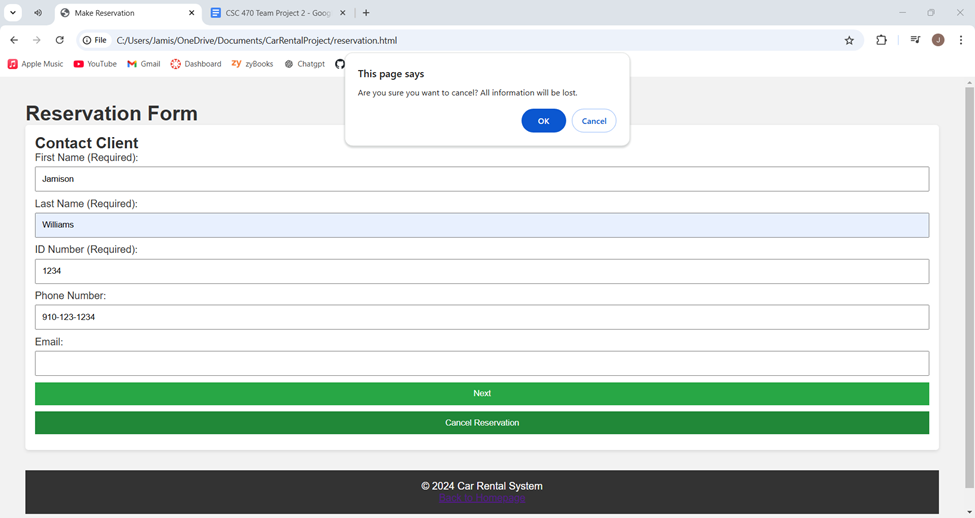


Index.html - this is the homepage for the CRS

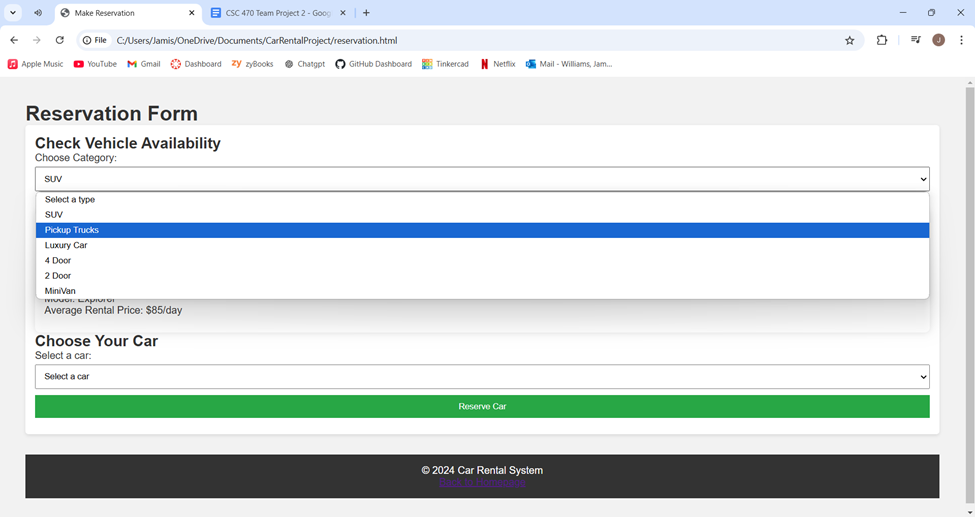
A client can choose between 4 main options: Make a reservation, return a car, inventory management, and the customer portal. A staff member can access the same page, doing the same functions



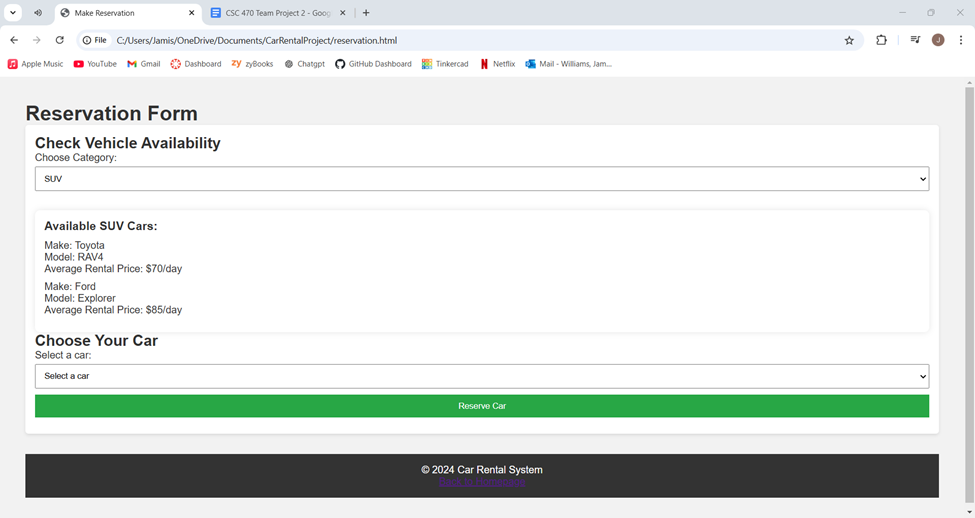
Reservation.html - this page is the form in order to rent a car, the client first must enter their information. If any line gets missed, this alert is throw (“Please fill in required fields”)

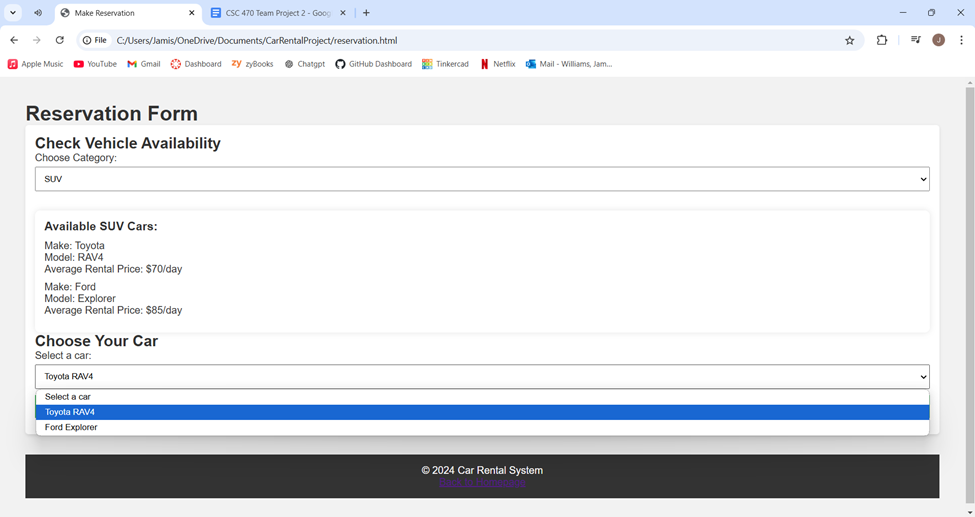


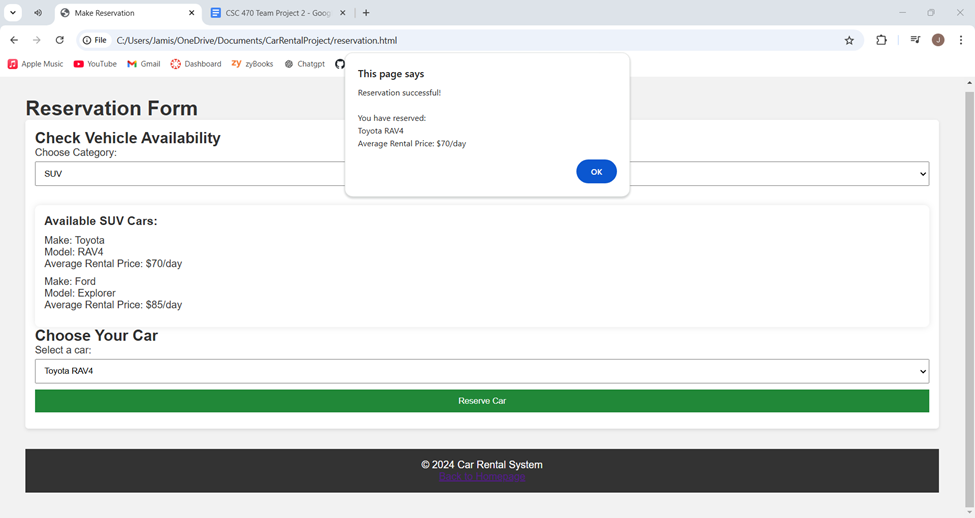
Reservation.html - there is a Cancel Reservation button along the bottom of the page that will clear the entire form



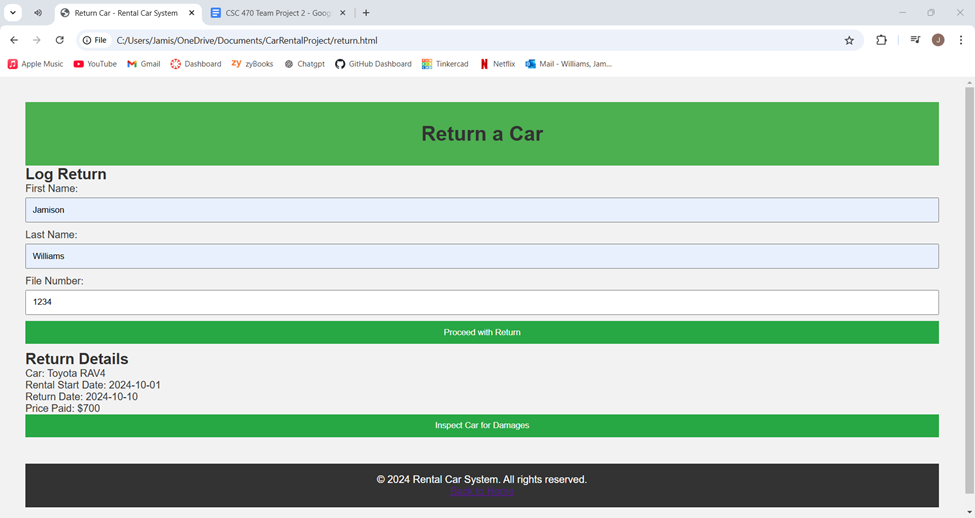
Reservation.html - this is the second part to the rental service, availability is shown and a vehicle should be chosen between the options available



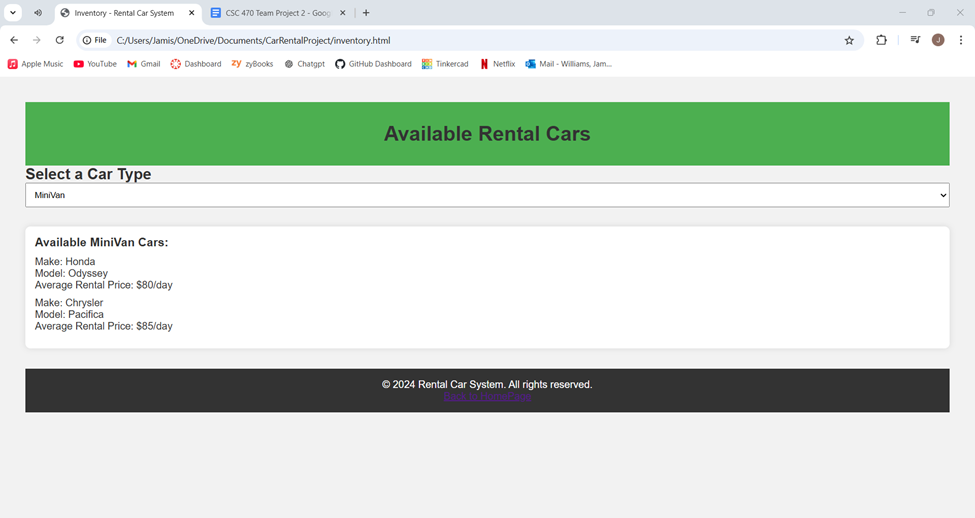




Reservation.html - the final stage the rental process, all selected information is displayed



Return.html - the return details is displayed once valid information is entered



Inventory.html - this page is to only view available cars

**Task Assignments:**

**Note**: All diagrams in this project were created using Lucidchart.

**Basic Requirements (10%):**

* Your project document should have a cover page with team member names and be followed by a page of content index in which each of items below clearly labeled and indexed\*
* At the end of the document, it also includes **a detailed work log\*** of teamwork, i.e., task assignments of each team member; team meetings and other activities, GitHub commit lists; etc.
* In each team meeting, especially when working on Part 5 below, assign one member to be the Facilitator (who reads questions), the second to be the Recorder (who writes down answers), the third to be the Moderator (who summarizes answers), the fourth and the fifth to be the Reviewers (who verify/test answers). Record them in the work log in your submission. The role assignment can be rotated. \*

**Project Deliverables (90%)**

1. Perform architectural design for the CRS system\* (25%) meeting logs(kalsoom)
   1. Develop an overall architectural structure for CRS with top-level components. Justify your decisions
   2. Describe the system in the call-and-return architectural style
   3. Derive two UML communication diagrams for the use cases of “Staff makes a reservation for a client” and “A client returns the rental car”, respectively
2. Develop elaborated design class diagrams\* (20%)(Aniyah)
   1. Add attributes for each class
   2. Add methods for each class
3. Conduct component-level design\* (20%)(Pamela)
   1. Go through the seven steps described in Section 11.3 and document major steps (your document should include at least activity diagrams and statecharts for the two use cases mentioned in 2.c above.
4. Perform user interface (UI) design\* (25%)(Jamison)
   1. For each of the use cases, produce a UI design (some modules may share the same UI)
   2. Use any programming language to develop prototypes for your UIs. Screenshot them in your submitted document

Meeting Logs:

Occurred Wednesday 10/18/24

**Start time:** 3:00 PM

**End time:** 3:45 PM

* Discussed project requirements
* Assigned tasks to team members
* Set future meeting times

Occurred at Monday 10/21/24

**Start time:** 7:00 PM

**End time:** 8:00 PM

* Asked some questions for clarification as a team
* Worked on tasks

Occurred at Tuesday9/24/24

**Start time:** 12:30 PM

**End time:**12:30 PM

* We discussed the feedback we received from Dr. Jin regarding our project. Based on the feedback,
* Need a lot of work done

In addition to these online meetings, we frequently used Teams chat to communicate project tasks and ask questions to our team.