University of Bahrain
College of Information Technology
Department of Computer Science
ITSE302 Project

CAR RENTAL SYSTEM (CRS)

Section 01:
Mohamed Naeem Mohamed (202009117) – Team Leader
Yusuf Hasan Jaafar (202009393)
Mohammad Fares Saleh (202010424)
Ahmed Abdulsalam Ahmed (202010320)

Business Case
Requirements Description:

With the increased competition in the business industry, companies strive to implement new strategies to beat competition and stay ahead of competitors. Aside from changing business strategies, most companies implement digital solutions to stay competitive. This step towards digitalization has taken the business world by storm due to its outstanding results in how business function in today's world. Digitalization results in increased efficiency, increased productivity, better employee morale, increased transparency, faster decision making, and better customer experience. On the other hand, companies discovered that manual processing done by humans sometimes causes delay and results are prone to errors. However, achieving successful digitalization, good planning and implementation of new software systems is required.

During the last two years, most companies were forced to transfer most of their services online due to covid outbreak. Although the pandemic is recessing, companies continue transform their processes online since technology shape a huge part of our lives today.

The Car Rental System (CRS) is designed to help car rental companies in transferring services online to public, which enables the car rental company in keeping records regularly about the processes. CRS system was designed for customers to browse, search, and view all cars available for rental and make the booking online anywhere and at any time. The system will also allow staff in having separate access to make specific changes to the system like add or remove cars. In addition, there will be a database integrated with CRS where a database administrator will be responsible for managing customer, staff, and car records.
To make use of the system services, the customer is first required to create an account by signing and filling a form which consist of the first name, last name, email, and phone number. After filling the form, the customer is required to create a username and password which will gain the user access to the system.

The user will be able to use the search engine integrated in the system to search for any car by entering the first letters. The CRS will provide the user the preview of the car and if the car is available for rental at the chosen date and time. If the car was not available, the system will suggest similar options to help the customer in booking. After booking the car, the user has the freedom of choosing the preferred payment method to officially finish the process. The user will then have the option to view and print the receipt once payment is successfully done online. Furthermore, the CRS will keep track of the customer usage to enhance the user experience and make use of the data to improve the company services. The system will also have the option of generating reports, printing invoices for each successful transaction which will later help the company in the production of monthly and annual reports. Consequently, the system admin can view all information related to the car bookings, together with the customer details. Additionally, information about staff can also be managed by the admin.

Aside for the system functionality, the system's user interface is easy to understand which makes interaction and usage easy for all kinds of people and this helps in turning website visitors into real buyers since good user interface will strengthen the interaction between the user and the system.

Finally, the car rental system makes renting cars easier, faster and saves time and costs. In addition, it makes the car rental service easily accessible by the public.

Use Case Model:

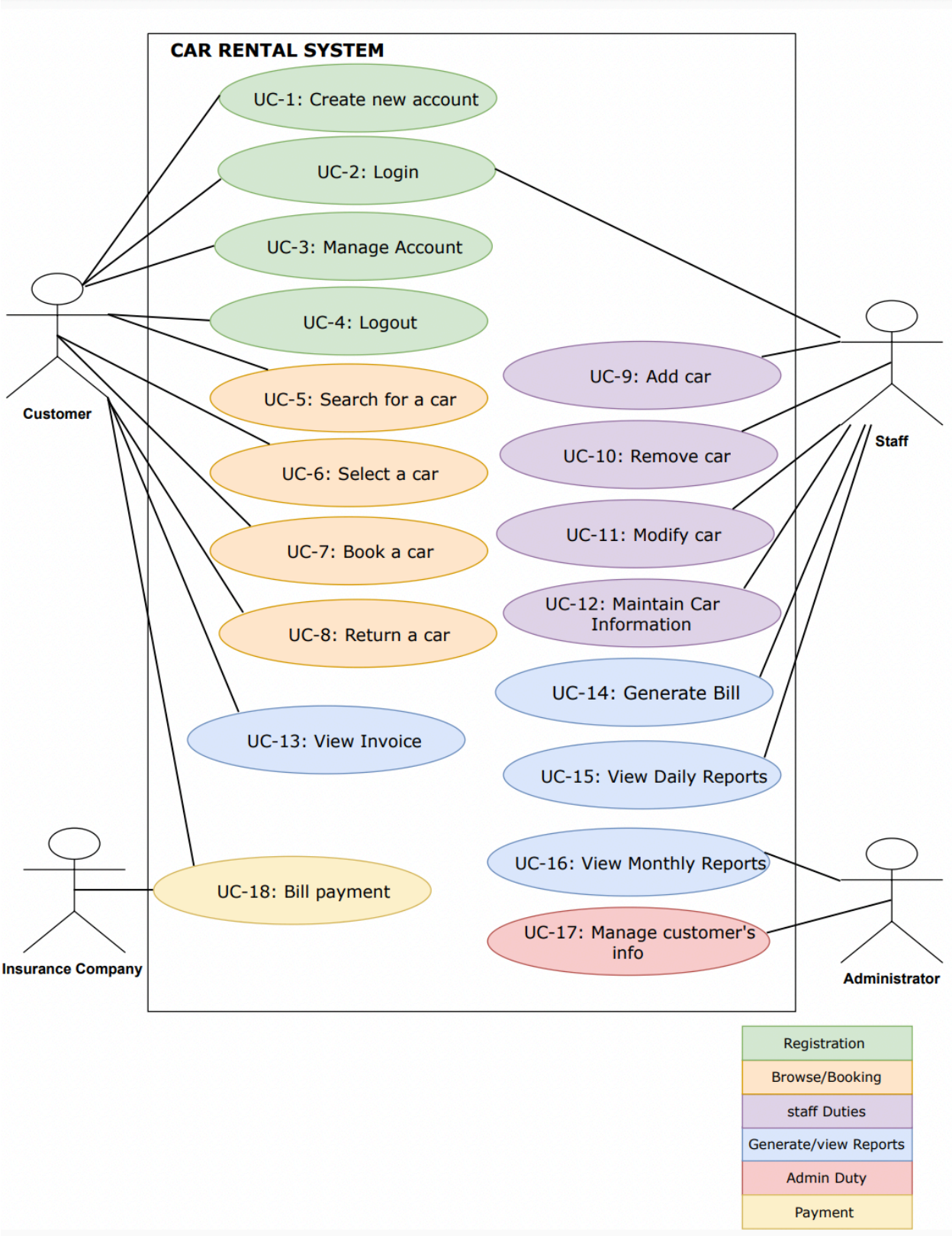

FIGURE 1.0 Use Case model for the Car Rental System (CRS)

Use Case Descriptions:

Use Case	Description
UC-1: Create new account	To make use of all the features and services of the car rental system, the customer is first required to create an account. The user should write the first name, last name, email, phone number, create unique username, and a strong password.
UC-2: Login	The user can immediately log-in after the creation of the account and gain access by using the username and password as a sign of authentication.
UC-3: Manage Account	The user can change his account info after logging into the system like changing the phone number or changing the username or even password.
UC-4: Logout	The user will be able to logout from the system after browsing the cars even if there is no booking made.
UC-5: Search for a Car	CRS has a built-in search engine which allows the user to search for the required car. The search engine has also a filter option based on type and price which makes the browsing much easier.
UC-6: Select a Car	The user can select any available car seen on the website and proceed to booking page.
UC-7: Book a Car	The user can book the car been added to cart after reading the whole agreement document, selecting the collection date and time, and paying 50% or full amount. In addition, the user will specify the rental period by choosing between 10 days and 3 months.
UC-8: Return a Car	The customer can return the car after finishing the rental period and the system will notice the customer in advance. The system will automatically count extra amount if the customer didn't follow the chosen return date.
UC-9: Add Car	The staff can add new cars to the system especially after studying the customer's demand and the most wanted cars.

UC-10: Remove Car	The staff can remove any car from the system based on certain conditions like accidents or defect problems.
UC-11: Modify Car	The staff can modify that car to be rented based on the customer's selection and preferences. In addition, staff can also change the pricing when necessary and apply special discounts in season period.
UC-12: Maintain Car Information	The staff will be able to maintain and update the car information if any updates required.
UC-13: View Invoice	The customer will be able to view the final invoice which includes all the information related to the transaction such as the necessary user details and the rented car details.
UC-14: Generate Bill	The staff generates the bill related to the transaction that was conducted and that bill is stored so that it can be used in the compilation of future company reports. The bill contains all the critical information needed for the generation of the future reports.
UC-15: View daily reports	The staff can have an overview of the daily reports which includes the cars rented, cars returned and by which customer with their details, and amount of bill that were payed or missed by the customers including the total cumulative of each day.
UC-16: View monthly reports	The administrator can have an overview of the data of every month's transactions of bills payed and should be payed, together with the number of customers that rented and returned cars within the that particular month.
UC-17: Manage customer's info	The administrator can edit any customer's profile if there is a mistake or error. The admin can also ban users who fail in following the rules and regulations of the car rental company. Admin has also the ability in giving some customers extra benefits.
UC-18: Bill Payment	The customer will be able to pay the remaining amount of the bill after the rental period. The customer can also pay across an insurance company affiliated with the car rental company to insure the rented car in case of any problem and the customer can receive 20% discount for every booking.

System Requirements
Quality Attributes List:

ID	Quality Attribute	Scenario	Associated Use Case
QA-1	Performance	The CRS will perform any operation or reservation under normal operation in less than 3 seconds. For example, the user can reserve a car for rent and the system will finish the payment transaction in less than 3 seconds.	All
QA-2	Reliability	The system performs majority of its tasks correctly within a time frame. For example, the system performs at least 99% of the tasks assigned correctly within 6 hours of use.	All
QA-3	Security	All user movements will be tracked by the CRS and the system will keep track of all the attempts to access data and will compare all the request patterns to identify any unauthorized behavior to protect user's sensitive data.	UC 1-8
QA-4	Security	The user attempts to login to CRS but the entered password or username is wrong. CRS tries to verify the user by sending an email to identify the user before login again.	UC-2
QA-5	Testability	Testing of the CRS is normally done by a testing team where system integration is being tested during compiling time to ensure all system functions are running right concurrently and to ensure 75% of path coverage within 2 hours of testing.	All
QA-6	Usability	The system has a simple and easy to use interface and the user should take at most 5 minutes to use the system in an efficient way.	All

QA-7	Portability	The system can be accessed easily on different web browsers like Chrome, Firefox, and Microsoft Edge. CRS can also operate on IOS and Android mobile devices.	All
QA-8	Safety	CRS avoids data corruption or loss whenever the user or staff inputs an invalid value while interacting with the system.	All
QA-9	Modifiability	The user can modify and change their personal information at any time and the updated data will be saved automatically in less than 2 seconds in the database with no side effects.	UC-3
QA-10	Modifiability	The staff can add, modify, remove cars, and maintain the car's information when necessary in less than 5 seconds with no side effects.	UC 9-12
QA-11	Performance	CRS collects performance data from a time server during peak times within 3 minutes and process all user commands to prevent data loss according to CON-6.	All

- **Priority Matrix:**

Business Importance/ Technical Risk	**L**	**M**	**H**
L	10	9	11
M	5	1	4
H	7	6	2,3,8

Constraints:

Some constraints on the car rental system (CRS) and its implementation were collected as shown below:

ID	Constraint
CON-1	The CRS should support Windows, Mac, and Linux operating systems and can be accessed through different web browsers like Chrome, Microsoft Edge, and Firefox.
CON-2	The system should handle a minimum of 500 users simultaneously.
CON-3	CRS data needs to be backed up every one hour on a database server.
CON-4	The whole project should be done in 8 months while achieving all system requirements and objectives.
CON-5	CRS must process user commands in under 5 seconds.

Concerns:

Given that this is a greenfield development, only these architectural concerns are identified initially.

ID	Concern
CRN-1	Team members have background knowledge in Java technologies, Spring, Java Web Start, and the Java language.
CRN-2	Establishing an overall initial structure.
CRN-3	Work allocation to members of the maintenance team.
CRN-4	All stakeholders need to be involved during design phase.

The Design Process

ADD Step 1: Review Inputs

The first step of the ADD method is all about reviewing the inputs and identifying which of the requirements discussed in phase 1 & 2 will be considered as drivers to the system. The inputs are summarized below:

Category	Details
Design purpose	This is a greenfield system in a mature domain. The purpose is to establish a detailed design that is sufficient to support the system's construction.
Primary Functional Requirements	From the use cases presented in phase 1, the primary ones were determined to be: **UC-2:** Because it acts as a security measure to prevent unauthorized access to the system. **UC-7:** Because the main aspect of CRS is within this functionality. **UC-14:** Because bill could be used for customer reference and payment. It can also be used as a historical document for the management.

Quality attributes scenarios

Scenario ID	Quality Attribute	Importance to the Customer	Difficulty of Implementation According to the Architect
QA-1	Performance	High	Low
QA-2	Reliability	High	High
QA-3	Security	High	Medium
QA-4	Testability	Low	Medium
QA-5	Usability	Medium	Low
QA-6	Portability	Medium	Medium
QA-7	Safety	High	High
QA-8	Modifiability	Low	Medium

From the list above, only QA-1, QA-3, and QA-6 are selected as drivers.

Constraints	All constraints listed in phase 2 are included as drivers.
Architectural Concerns	All concerns listed in phase 2 are included as drivers.

Iteration 1: Establishing an Overall System Structure

Step 2: Establish Iteration Goal by Selecting Drivers

This is the first iteration throughout the design of a greenfield system, so the goal of the iteration is to achieve the overall system structure to fulfil CRN-2.
Even though a general architectural concern drives this iteration, the architect must take into consideration all the drivers that could affect the general system's structure, therefore the architect must be aware of the following:

- QA-1: Performance
- QA-3: Security
- QA-6: Portability
- CON-1: The CRS should support Windows, Mac, and Linux operating systems and can be accessed through different web browsers like Chrome, Microsoft Edge, and Firefox.
- CON-3: The system supports single-user software (which means only one user can use the system in one device at a time).
- CON-4: CRS data needs to be backed up every one hour on a database server.
- CRN-1: Team members have background knowledge in JAVA programming language.
- CRN-4: All stakeholders need to be involved during design phase.

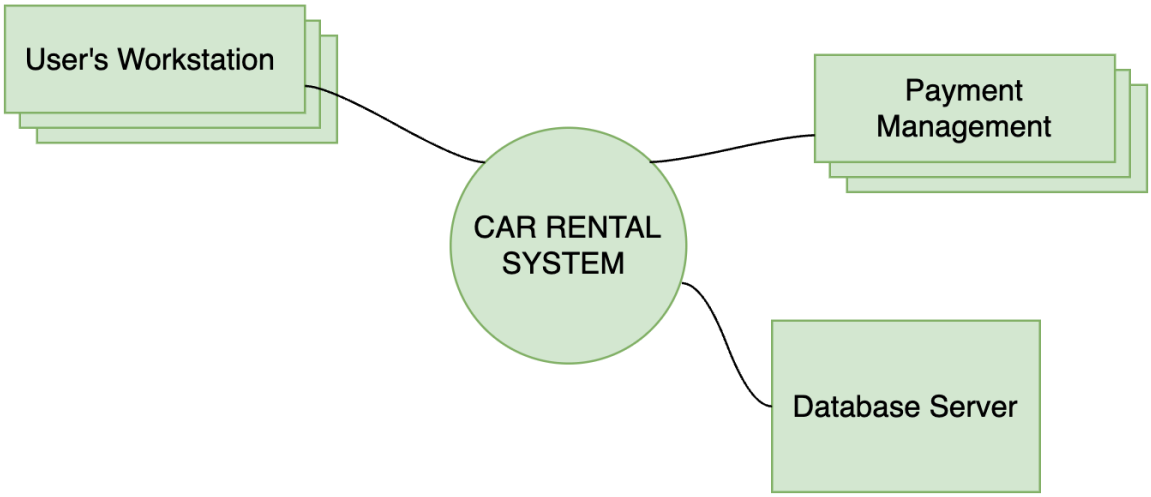

FIGURE 2.0 Context Diagram for the Car Rental System (CRS)

Step 3: Choose One or More Elements of the System to Refine

This system is a greenfield development effort, therefore, the element to refine is the entire car rental system. In this case, refinement is performed through decomposition.

Step 4: Choose One or More Design Concepts That Satisfy the Selected Drivers

In this first iteration, the rich internet application is selected as the design concept which will provide the overall structure of the application. The following table below summarizes the design decisions about choosing the design concepts and the alternatives.

Design Decisions And Location	Rationale
Logically structure the client part of the system using the Rich Internet Application reference architecture	This reference architecture is designed for creating applications with a rich user interface and sophisticated business logic that runs inside a browser. This reference architecture has also been selected since the developed application needs to have a rich user interface, a processing needs to be done on the client side, and we need to deploy and update the application easily without having to install it on the user's machine.

Discarded Alternatives:

Alternative	Reason for Discarding
Web Application	Although this reference architecture facilitates deployment and updating, it was discarded since it is difficult to provide a rich user interface.
Service Application	This reference architecture provides services to machines containing non-interactive layers and this contradicts with CON-6 where the system needs to process user commands in a certain time.
Mobile Application	This reference architecture is typically concerned toward development of applications on handheld devices. However, this alternative is discarded since this type of device was not considered for accessing the system.

Design Decisions And Location	Rationale
Logically structure the server part of the system using the Service Application Reference architecture	This reference architecture which is used to design a logical structure for the server part has no presentation layer so there is no user interface. And it is considered adequate to meet the requirements since it is used to expose services consumed by other applications.
Physically structure the application using the Three-tier deployment pattern	Because CRS needs to be accessed from a web browser as proposed in CON-2 and back-up should be done in an existing database server according to CON-4, a three-tier deployment will be suitable. It's obvious that some modifications need to be done to the web/app tier and database tier, but this will be later addressed in Iteration 3. Discarded alternatives include the other patterns with n!=3 and the two-tier deployment was rejected since an existing database server needs to be associated into the system and its very basic even in terms of security which is a high priority to the customer. In addition, all n>3 are discarded since additional tiers are unnecessary for now and they might add more cost, complexity, and lag.
Build the user Interface of the client Application using the Angular framework and other Java technologies	The standard framework to be utilized for building the rich internet application ensures portability (QA-7) and developers are already familiar according to CRN-1. Discarded alternatives include Laravel but the developers were not familiar with it.
Deploy the application using the Java Web Start technology	Since access to application is via a web browser according to CON-1. In addition, this technology is also available for each update. Applets are a discarded option since they need to be reloaded every time a web page is loaded, which increases bandwidth.

Step 5: Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces

The instantiated design decisions considered and made are summarized in the following table:

Design Decisions And Location	Rationale
Implement API in the server side of the Rich Internet Application	Since the code executes first on the server side before being sent to the browser, an API is recommended to make code execution much efficient and makes it a lot easier to create web pages that work on multiple browsers correctly which supports CON-1. In addition, it speeds up page loading which improves the user experience and supports CON-6.
Parts of the data source could be accessed in the data layer in the client part	The users can store, access, and modify their information and data in data source of the used device aside from the isolated storage.

Step 6: Sketch Views and Record Design Decisions

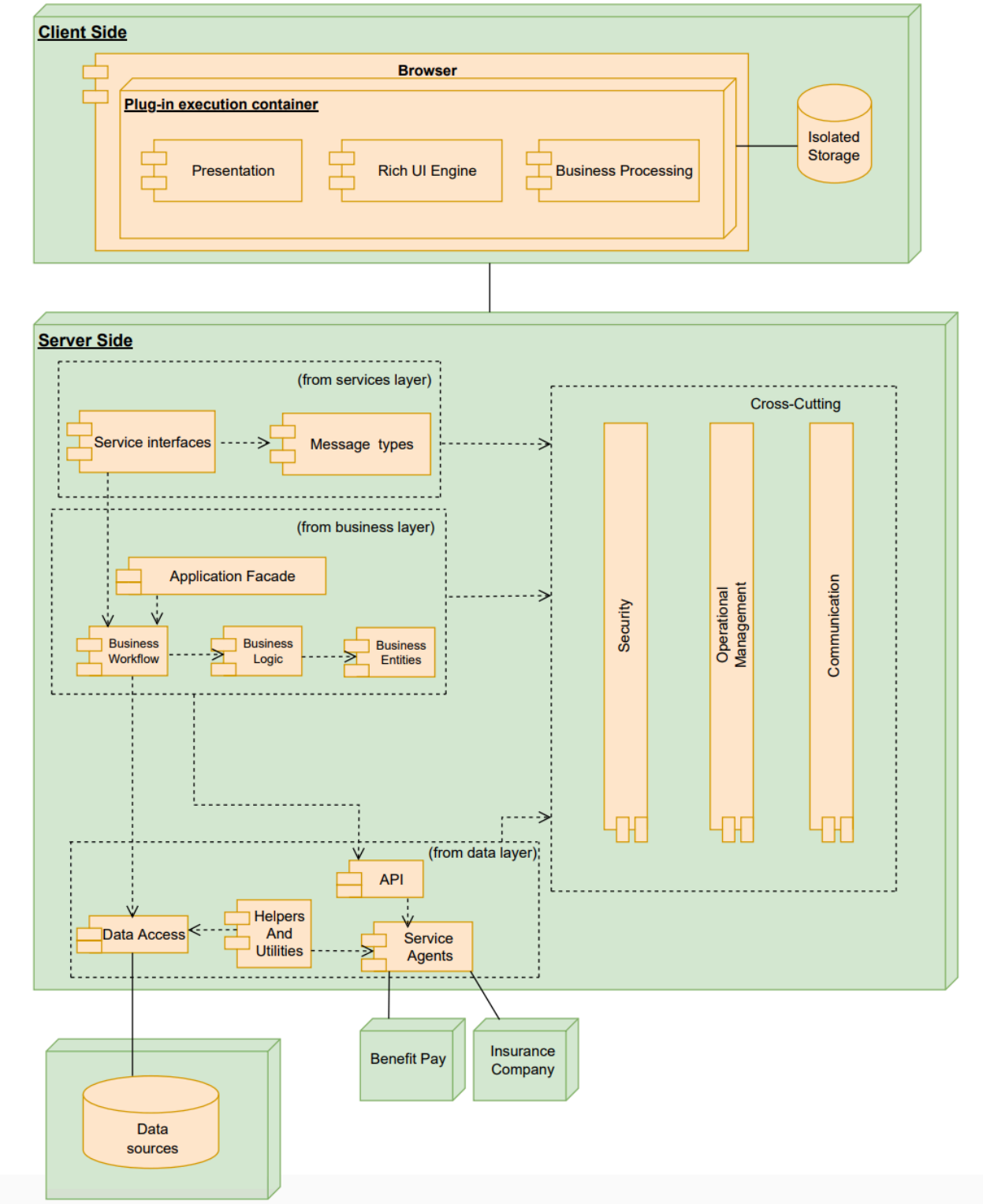

FIGURE 3.0 Modules obtained from the selected reference architecture

The sketch shown above was created using an online tool called draw.io and the table below represents each element with a short description of its responsibly.

Element	Responsibility
Presentation (CS)	Responsible for managing the user interaction.
Rich UI engine (CS)	Responsible for rendering user interface elements inside the plug-in execution container.
Business Processing (CS)	Responsible for handling business logic on the client side.
Service Interfaces (SS)	Responsible for exposing services which are consumed by components running on the browser.
Message Types (SS)	Responsible for handling the message types which are exchanged between the server and client part of the application.
Application Facade (SS)	Provides a simplified interface to the business logic components.
Business Workflow (SS)	Accountable for handling long-running business processes which may contains multiple use cases execution.
Business Logic (SS)	Responsible for retrieving and processing application data and applying business rules on the retrieved data.
Business Entities (SS)	These components represent the business domain entities and their associated business logic.

Data Access (SS)	These components encapsulate persistence mechanisms and provide operations used to retrieve and store information.
Helpers and Utilities (SS)	These components contain functionality common to other modules in the data layer but not specific to any of them.
Service Agents (SS)	These components abstract communication mechanisms which are used to transfer data to external services.
Cross cutting (SS) Security	These components include cross-cutting functionality which handles the security part of the system such as user authentication and authorization.
Cross cutting (SS) Operation Management	These components include cross-cutting functionality which handles the operation management part of the system such as exception management, validation, and logging.
Cross cutting (SS) Communication	These components include cross-cutting functionality which handles the communication aspect of the system across physical tiers and layers.
Data Sources	Location where data being used in the system is stored in and originates from.
API	Implemented in the instantiated reference architecture for better integration, automation, efficiency, and customization in the server side of the system.
Other System (Benefit Pay & Insurance Company	Other systems that are associated with the main system. These include Benefit Pay which is used for payments and insurance company where customers could make an insurance invoice from the system itself.

Deployment Diagram

The deployment diagram in Figure 4.0 sketches an allocation view which illustrates where components associated with the modules in the previous diagram

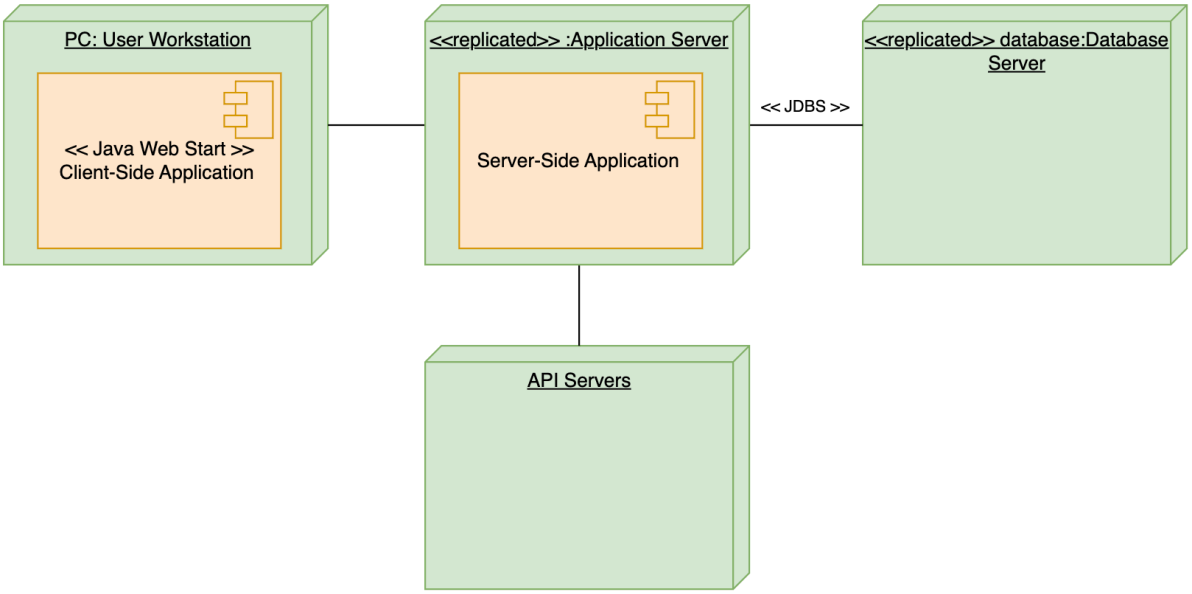

FIGURE 4.0 Initial Deployment diagram for the Car Rental System (CRS)

The responsibilities of the elements are summarized below:

Element	Responsibility
User Workstation	The server which hosts the client-side logic of the application.
Application Server	The server which hosts the server-side logic of the application and being connected with an **API** side server.
Database Server	The server which hosts the relational database.

In addition, information about relationships between some elements in the diagram is mentioned below:

Relationship	Description
Between Application server and database server	Communication with the database is done using the JDBC Protocol.

Step 7: Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

The following table summarizes the design process done in iteration 1 using the Kanban board technique.

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During the Iteration
	UC-2		The selected reference architecture will establish the module which will support this functionality.
	UC-7		The selected reference architecture will establish the module which will support this functionality.
	UC-14		The selected reference architecture will establish the module which will support this functionality.
QA-1			No relevant decisions were made.
QA-3			No relevant decisions were made.
	QA-6		The use of Java Web Start technology to deploy the system allowed access via the web browser to use the rich internet application. Since the rich internet application is being programmed in JAVA, this supported execution under multiple browsers.

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During the Iteration
		CON-1	The use of Java Web Start technology to deploy the system allowed access via the web browser to use the rich internet application. Since the rich internet application is being programmed in JAVA, this supported execution under multiple browsers.
		CON-2	Structuring the system using three tiers will allow multiple users to access the application server.
		CON-3	Since the system is structured physically using the 3-tier deployment pattern, there is a database which communicates with the application server using JDBC protocol.
	CON-4		Due to the project manager and timeline, the whole project is on time and track.
		CON-5	Due to implementation of API in the server side, efficiency is improved.
	CRN-1		The knowledge of the developers was taken into consideration up to this point.
		CRN-2	The selection of the reference architecture and deployment pattern.

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During the Iteration
CRN-3			No relevant decisions were made.
CRN-4			No relevant decisions were made.

Iteration 2: Identifying Structures to support Primary Functionality

This section will present the results of the activities that are carried out in each of the steps of ADD in the second iteration of the design process of the Car Rental System (CRS).
In this iteration, we will move from the general descriptions of functionality used in iteration 1 to a more detailed decisions which will later drive the implementation and therefore the formation of the development teams. This movement from the generic to the detailed specific description is intended and part of the ADD method. This is because we cannot design everything in advance, so we need to be thoughtful about the decisions we make through the design process to ensure design is made in a right and systematic way.
In the previous iteration, the goal was to establish an overall system structure and it has been met. The new goal for this second iteration is to reason about the implementation units which affect team formation, interfaces and the methods for the distribution and implementation of the development tasks.

Step 2: Establish Iteration Goal by Selecting Drivers

The main goal of iteration 2 is to address the general architectural concern of identifying structures to support primary functionality. These elements need to be identified not only for understating how the functionality is supported, but for also tackling CRN-3, which is allocating the work to the maintenance team members.
Aside from CRN-3, CR-4 will be addressed, and the primary use cases will also be considered.

- UC-2
- UC-7
- UC-14

Step 3: Choose One or More Elements of the System to Refine

The Modules located in the different layers of the identified two reference architectures during iteration 1 will be the elements to be refined. Consequently, the components associated with modules located in the different layers needs to have collaboration to support the system functionality.

Step 4: Choose One or More Design Concepts That Satisfy the Selected Drivers

Design Decisions And Location	Rationale and Assumptions
Create a domain model for the application	It is required to establish an initial domain for the system before beginning a functional decomposition. This domain should list all significant elements and their relationships. There are no worthwhile alternatives. At some point, a domain model needs to be developed; otherwise, an ad hoc architecture that is challenging to comprehend and manage would arise.
Identify domain objects that maps to functional requirements	A domain object is a self-contained building piece that needs to contain each unique functional component of the system. One way is to directly break down layers into modules without taking domain objects into account, however this raises the possibility of overlooking a need.
Decompose domain objects into general and specialized components	Although domain objects represent entire functional sets, these functional sets are backed by finer-grained elements found within the layers. In this pattern, the "components" are what we have referred to as modules. Module specialization is related to the layers in which they are found (e.g., UI modules). Decomposing the layers into modules to provide functionality is the only viable solution.
Use Angular framework and angular rest orm	A framework which is widely used to support application development is angular. an object to relational mapping (orm) framework that integrates well with angular is angular rest orm. jee was considered as an option for application development. because it was deemed to be more "useful" and the development team was already familiar with it, angular was ultimately chosen, leading to greater and earlier productivity. other orm frameworks were not taken into consideration because the development team were already familiar and satisfied with the functionality of angular rest orm.

Step 5: Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces

The instantiation design decisions made in this iteration are summarized in the following table:

Design Decisions And Location	Rationale
Create only an initial domain model	To expedite this design step, only a basic domain model is produced; the entities that take part in the core use cases must be identified and modeled.
Map the system use cases to the domain objects	By looking at the system's use cases, domain objects can be initially identified in any case. Domain objects are identified for each of the use cases to address CRN-3.
Decompose the domain objects across the layers to identify layer-specific modules with an explicit interface	This method makes sure that all the functionalities are supported by modules. Only the primary use cases will receive this assignment from the architect. By identifying the remaining modules, another team member can distribute the work among the team members. After the set of modules has been established, a new architectural concern is discovered: CRN-5: Majority of modules shall be unit tested This concern only applies to "a majority of modules," as it is challenging to separately test modules that handle user interface functionality.
Connect components associated with modules using Angular	Due to this framework's inversion of control methodology, various elements of it can be supported, and the modules can be unit tested (CRN-5).
Associate frameworks with a module in the data layer	The modules that make up the data layer have ORM mapping built in. These modules operate with the previously chosen Angular framework.

Step 6: Sketch Views and Record Design Decisions

In this step of Iteration 2, several diagrams are created based on the decisions made during step 5.

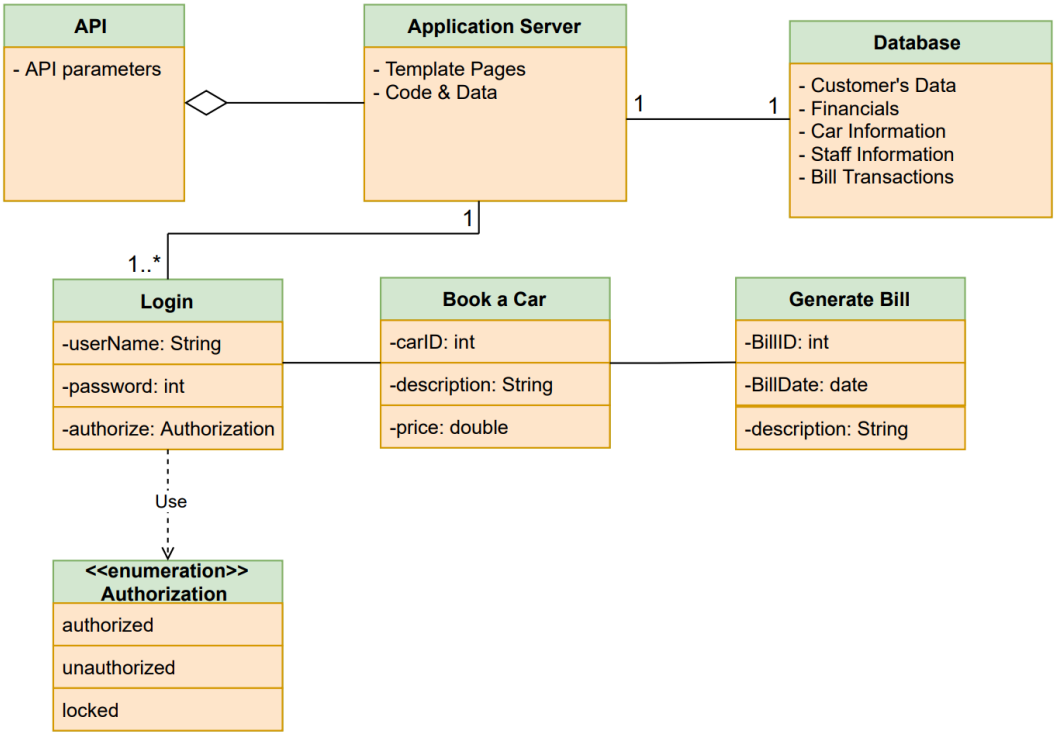

FIGURE 5.0 Initial domain model

As shown in the above figure, the initial domain model created covers the primary functionalities that are mentioned above in step 1 of the ADD process specifically in the review inputs. These primary use cases include UC-1: Login, UC-7: Book a Car, and UC-14: Generate Bill. Aside from the primary use cases, the important aspects of the deployment model which has direct connection with the primary use cases is also found in the domain model

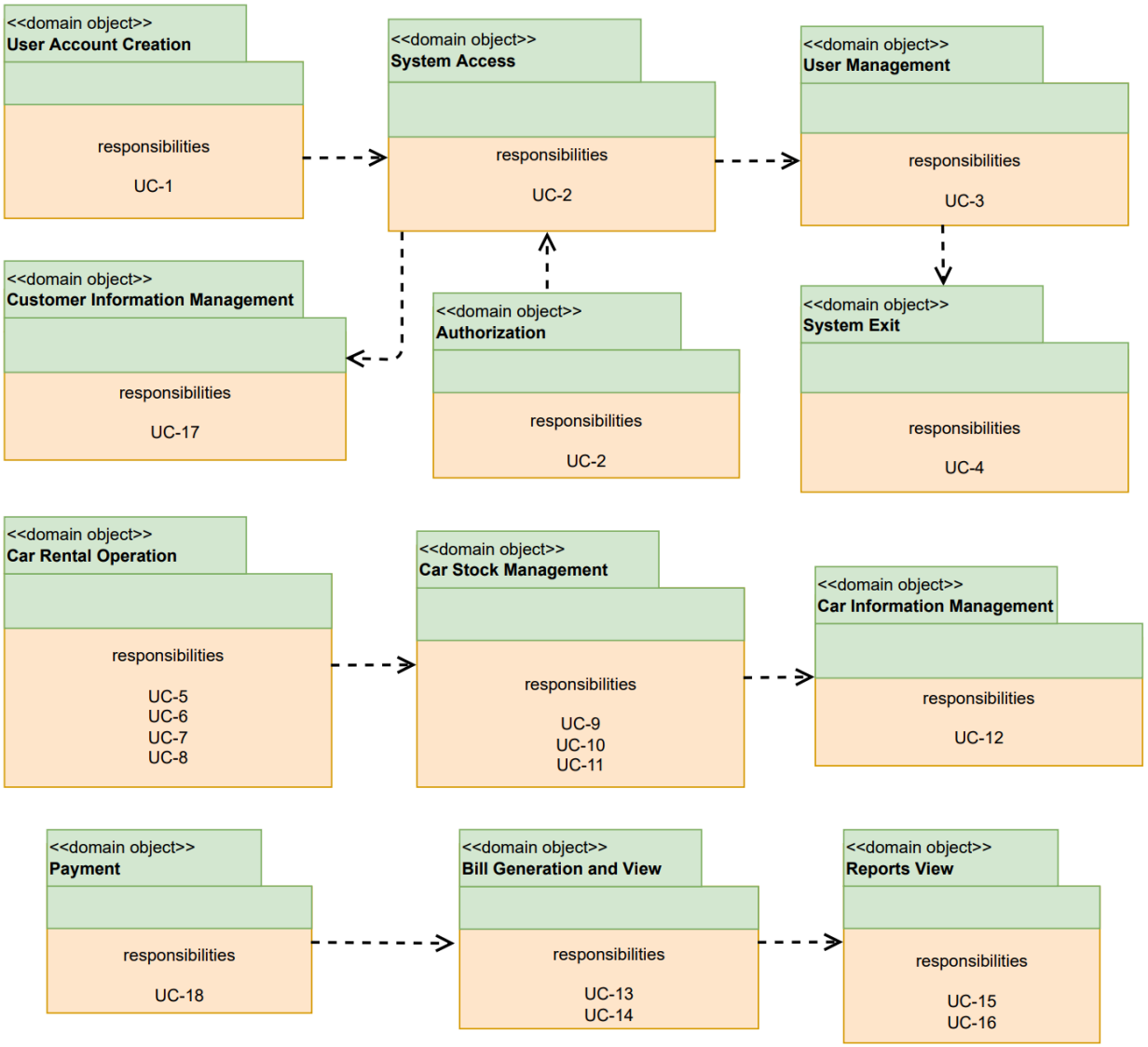

FIGURE 6.0 Domain Objects associated with the use case model

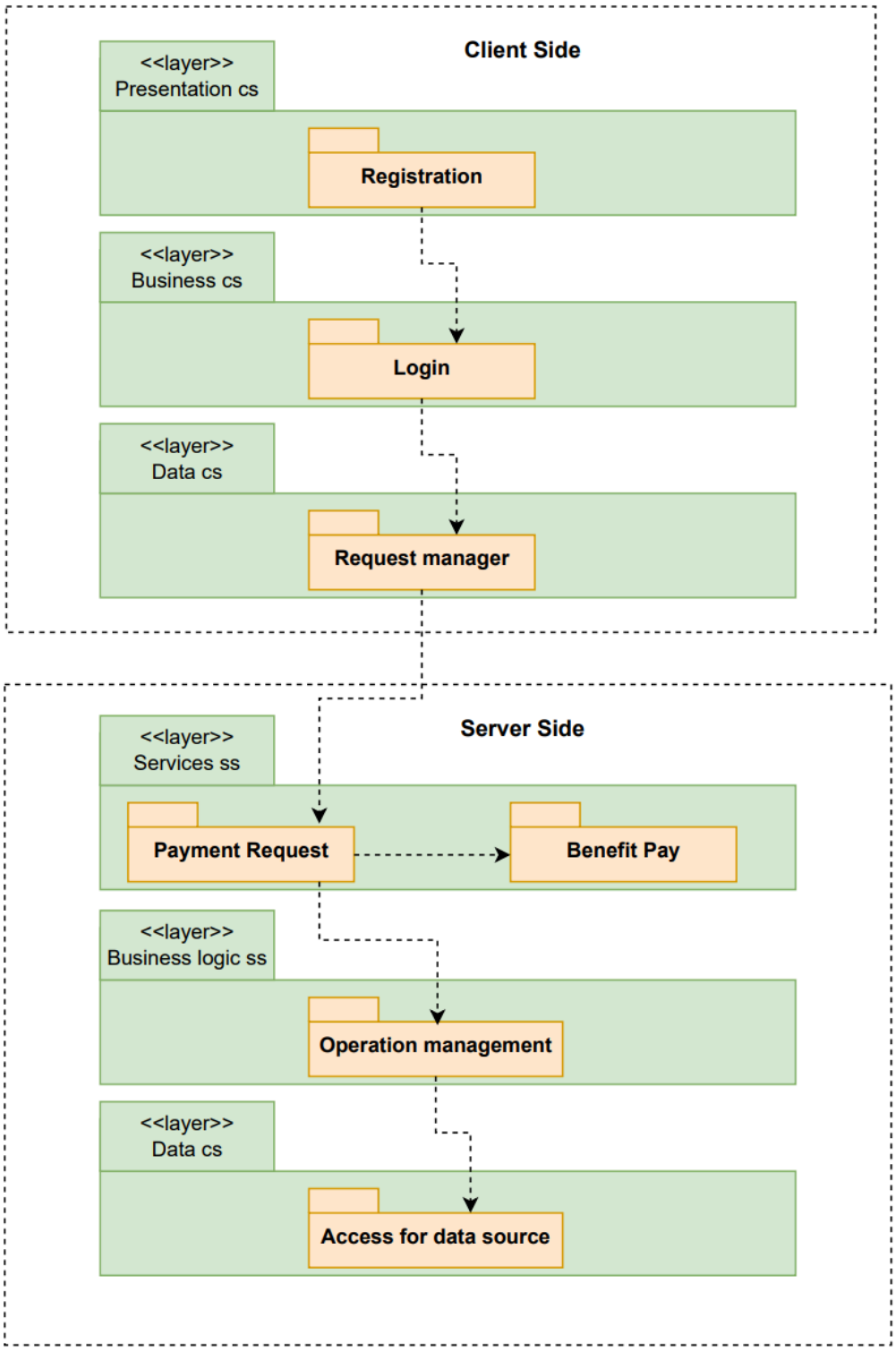

FIGURE 7.0 Modules that support the primary use cases

Element	Responsibility
Registration	This element is responsible for allowing the user to create a new account by filling a form.
Login	Responsible for displaying the login option for the user to access the application.
Request Manager	Responsible for communication and interaction with the server-side logic.
Payment Request	Responsible for prompting the user to payment of the calculated price after customizing the request based on the preference.
Benefit Pay	A 2nd party application which can be used for bill payments. It acts as a fast and convenient method of payment.
Operation Management	This element can be found in the server-side of the system, and it is responsible for the complementation of the car rental operations.
Access for data source	Responsible for saving the data in the server-side.

Now, the sequences diagrams for UC-2, UC-7, and UC-14 are created to define interfaces.

UC-2: Login

The following diagram displays the login process for the user. The user registeres the username and password into the system. The system receives the inputs from the user and validates the information inputted and then allows the user to either edit the account username or password if the user wishes to do so.

Identifying initial methods for the interfaces of interacting elements can be done from the interacting elements in the sequence diagram:

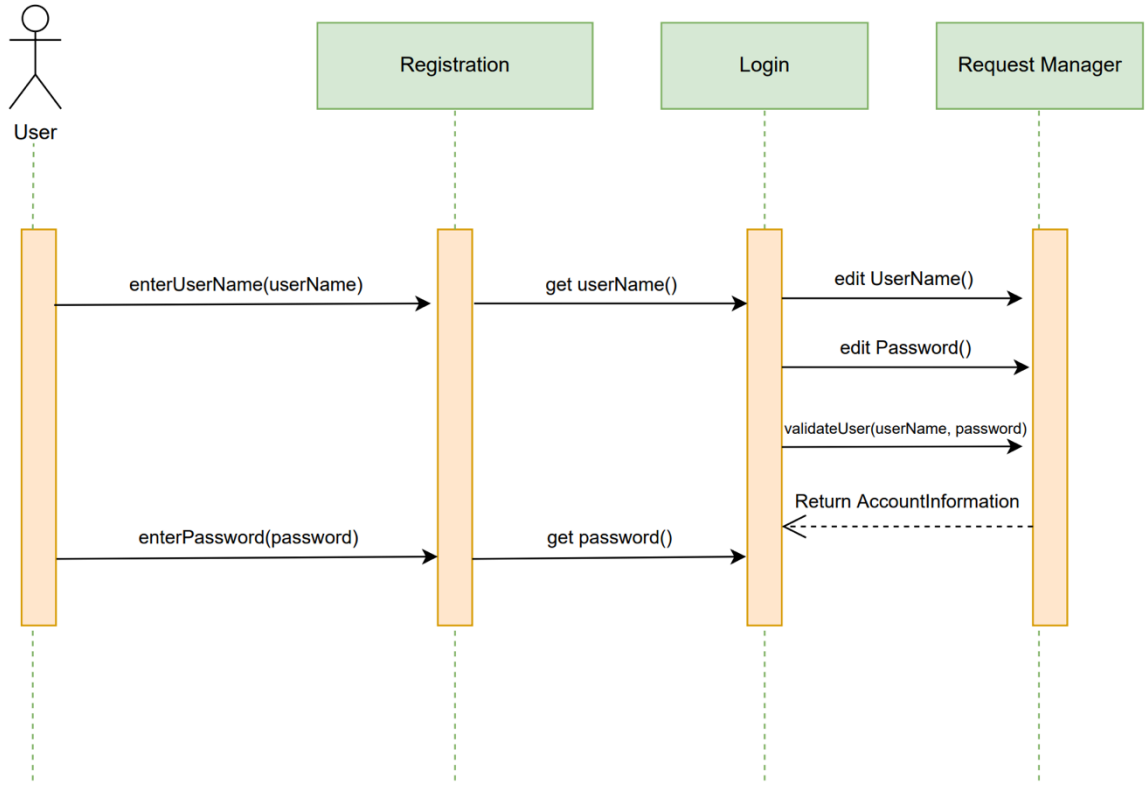

FIGURE 8.0 Sequence diagram for use case UC-2

Element	Responsibility
Registration	• enterUserName(username): Allows the user to enter the username. • enterPassword(password): Allows the user to enter the password.
Login	• getUserName(): System obtains the username. • getPassword(): System obtains the password.
Request Manager	• editUserName(): Allows the user to edit the username. • editPassword(): Allows the user to edit the password. • validateUser(): Validates information entered by the user. • returnAccountInformation(): System sends the account information back to the login element.

UC-7: Book a Car

The following diagram displays the process of a user booking a car for rental. The system displays the list of cars, and then the user selects the car they wish to rent and the system displays the details of the car so the user can make sure of the car they have chosen. The user then selects the rental and return dates for the chosen car and the system checks whether the car is available during that time or not and returns the availability status to the user.

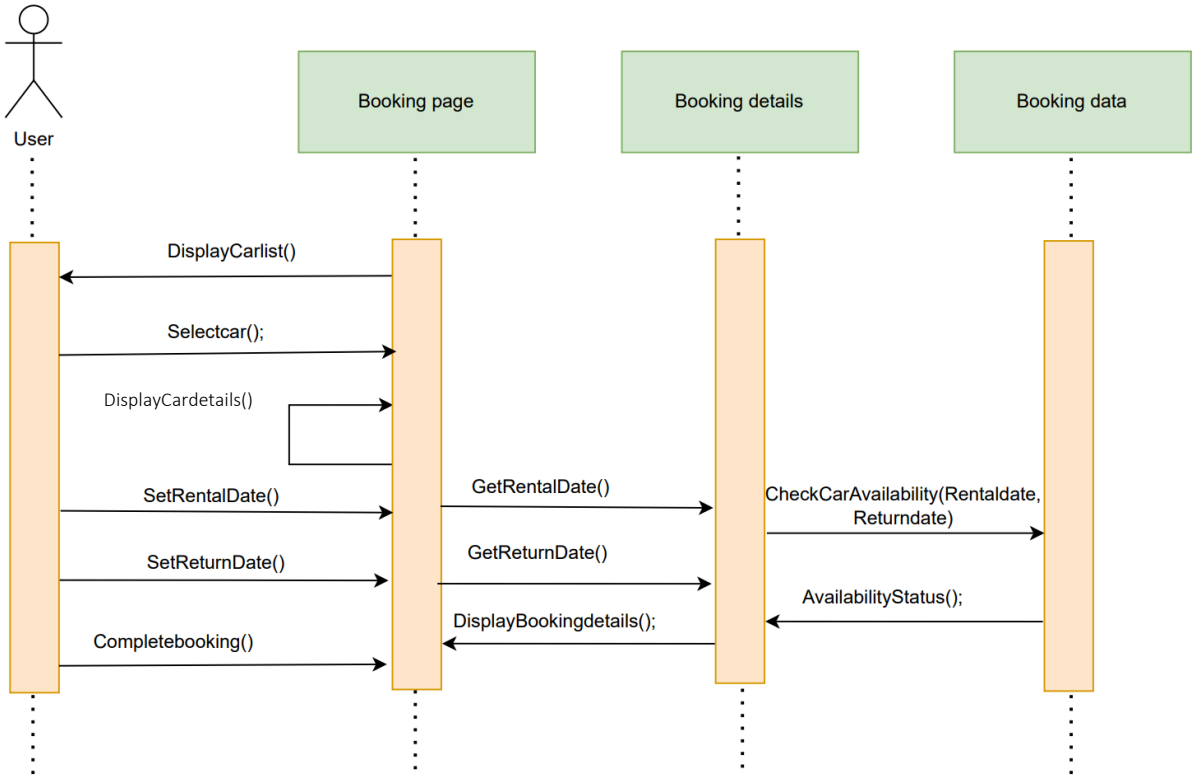

FIGURE 9.0 Sequence diagram for use case UC-7

Element	Responsibility
Booking Page	• DisplayCarlist(): Displays the list of cars for the user. • Selectcar(): The user selects a car from the displayed list. • DisplayCardetails(): The system displays the car details on the page for the user. • SetRentaldate(): The user selects the date they want to receive the car for rental. • SetReturndate(); The user Selects the date they want to return the car on. • Completebooking(): The user confirms the booking after selecting the car and rental/return dates.
Booking Details	• GetRentalDate():System obtains Rental date. • GetReturndate():System obtains the Return date. • Displaybookingdetails(): The details are sent to the page and displayed to the user.
Booking Data	• CheckCarAvailability: The system checks whether the chosen car is available at the selected date • AvailabilityStatus(): The system sends the availability status and displays it to the user.

UC-14: Generate Bill

The system uses the car chosen and the duration of the rental to calculate the price of the rental over the entire duration of the rental.

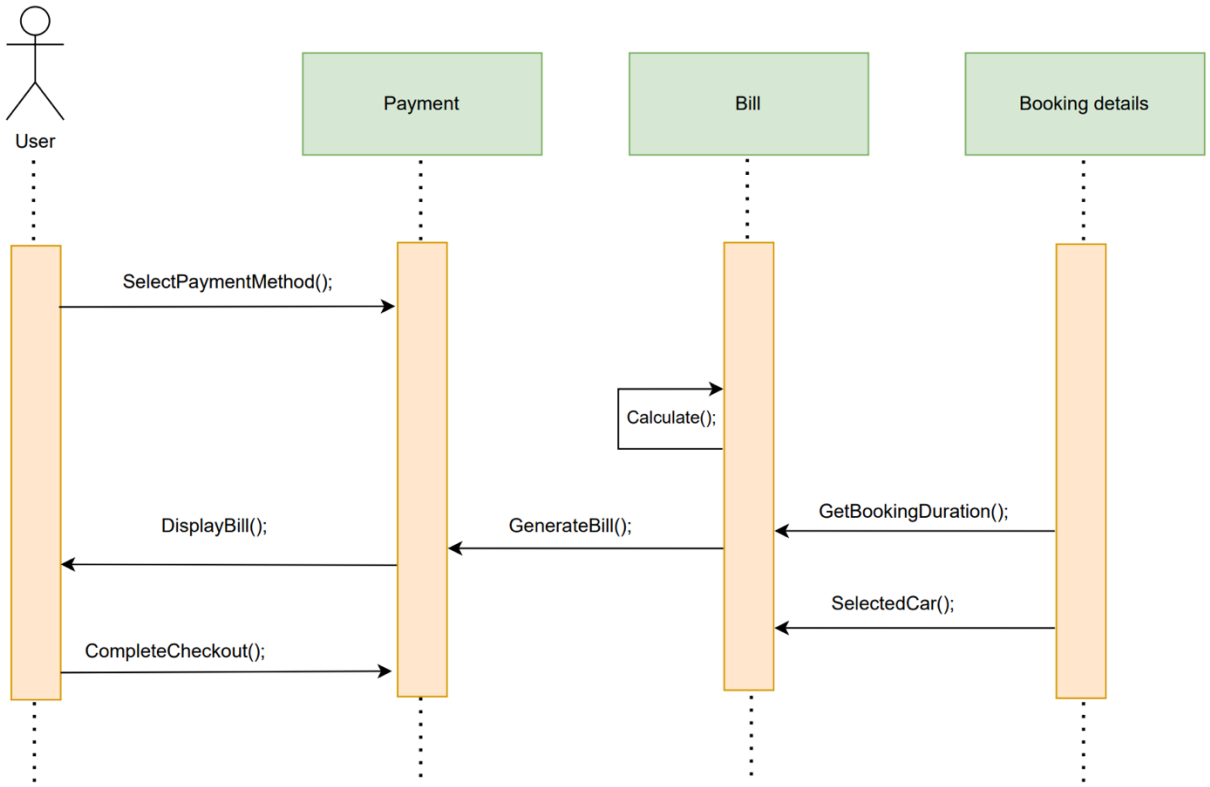

FIGURE 10.0 Sequence diagram for use case UC-14

Element	Responsibility
Payment	• Selectpaymentmethod(): The user selects the payment method most suitable for them • DisplayBill(): The system displays the Bill for the user • CompleteCheckout(): The user completes the checkout after reviewing the details and the bill.
Bill	• Calculate(): The system uses the booking details to calculate the cost of the rental. • GenerateBill(); The system uses the calculation to generate the bill with the cost and rental details.
BookingDetails	• GetbookingDuration(): The system gets the duration from the Rental and return dates. • SelectedCar(): The system gets the car chosen by the user.

Step 7: Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During the Iteration
		UC-2	Preliminary Interfaces and modules across the layers have been identified to support this use-case.
		UC-7	Preliminary Interfaces and modules across the layers have been identified to support this use-case.
		UC-14	Preliminary Interfaces and modules across the layers have been identified to support this use-case.
	QA-1		Identification of Elements which supports the associated use case (UC-7) were identified.
	QA-3		No relevant decisions were made.
	QA-6		Identification of Elements which supports the associated use case (UC-2) were identified.
	CON-4		Due to the project manager, the timeline and project's progress is still on track and enough resources are available.

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During the Iteration
		CRN-1	The team members ability and knowledge were taken into consideration when new technologies were introduced.
		CRN-3	Identification of modules which are associated with almost all use cases helped in addressing this concern which is allocation of work between maintenance team members.
		CRN-4	Identification of modules which are associated with almost all use cases and the project manager decisions helped in addressing this concern which is allocation of work between stakeholders in the design phase.
	CRN-5		Unit-testing architectural concern which was introduced earlier in this iteration is partially addressed since use of an inversion control methodology to connect components associated with the modules helped addressing this concern.

Iteration 3: Addressing Quality Attribute Scenario Driver (QA-3)

The actions that are carried out in each of the ADD phases during the third iteration of the design process are summarized in this section. Adding on the fundamental structure decisions made in the above two iterations, the fulfillment of some of the more important quality attributes needs to be done. Only one quality attribute scenario will be covered in this iteration.

Step 2: Establish the Iteration Goal by Selecting Drivers

For this iteration, the architect will preliminarily focus on QA-3 quality attribute scenario:

(All user movements will be tracked by the CRS and the system will keep track of all the attempts to access data and will compare all the request patterns to identify any unauthorized behavior to protect user's sensitive data like bank data information.)

Step 3: Choose One or More Elements of the System to Refine

The physical nodes mentioned in the first iteration are the elements chosen to be refined for this **security** scenario.

1. Database Server
2. Application Server

Step 4: Choose One or More Design Concepts That Satisfy the Selected Drivers

Design Decisions and Location	Rationale and Assumptions
Introduce the **detect intrusion tactic** by comparing the service request patterns or the network traffic within the car rental system to identify or find any unauthorized behavior within the database	By comparing the service request Patterson within the system for any task, the system can distinguish and identify patterns and match them with known signs of intrusions. In addition, aside from detecting and preventing intrusions, this solution monitors the inbound and outbound traffic in the system's network for any suspicious activity and data breach.
Authenticating and Authorizing actors to ensure that only authenticated actors have the right to access the system to modify data or make any change	By authorizing every user accessing the system, this will help in limiting the access since parts of the system will be controlled and who may access them. Consequently, this helps in tracing back any actor if any problem or issue occurred.
Present the **active redundancy tactic** by replicating the application and database server	By replicating the application and database server, the system will be well preprepared in case of any system failure happening due to either traffic or service request. In addition, these replicated servers will take place of the original ones without affecting the server functionality.

Step 5: Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces

Design Decisions and Location	Rationale
Implement a **proxy firewall** between the client and application server and a **database firewall** between the application server and the database server	By placing a proxy firewall between the client and application server, this will provide the application layer filtering and can analyze the messages to differentiate between valid and malicious system requests. In addition, a database firewall implemented will be responsible for data protection and identification of attempted attacks to the database.
Implement a **load balancer** in the application server and the database server as well as the replicated database server	The load balancer implemented will be responsible for distributing the workload between the two servers which makes the system more efficient. In addition, in case of any system attack, system recovery is easier and faster.

Step 6: Sketch Views and Record Design Decisions

Figure 11.0 shows a refined deployment pattern which includes the new design decisions of this iteration.

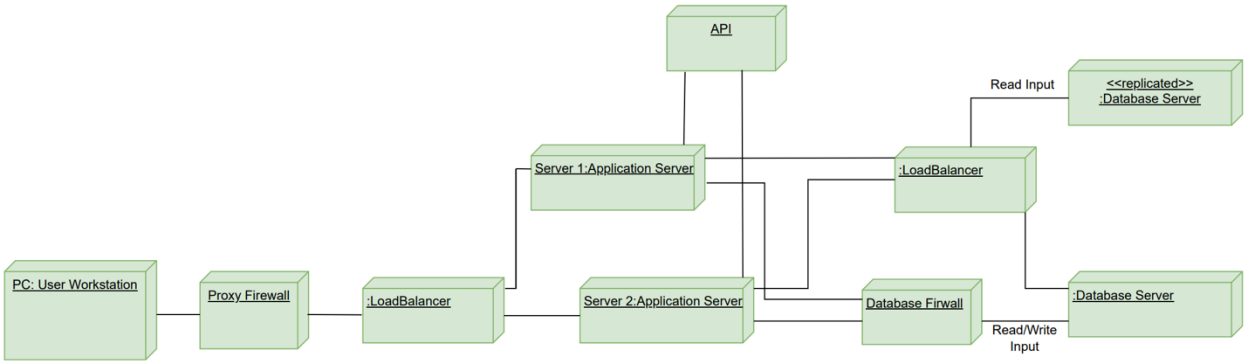

FIGURE 11.0 Refined deployment diagram

The following table describes responsibilities for elements which are not listed in earlier in Iteration 1.

Element	Resposnibility
Proxy Firewall	This element will be responsible in protecting the application server by filtering all service requests.
Database Firewall	This element will prevent unauthorized access from any unauthorized users and will monitor the database to protect it from any attacks.
Load Balancer	A load balancer will split users between two different application servers and will reduce traffic and heavy load on just one server, which could cause the system to crash. If a server crashes due to high demand for system services, the other server will allow users to continue using it normally while the crashed server is recovering.

The UML sequence diagram shown below in figure 12.0 illustrates how the proxy firewall and database firewall was introduced in this iteration blocks any unauthorized or suspicious access to the car rental system and database server. In addition, the way these two additions interact with other elements in the deployment diagram could be analyzed from this sequence diagram and they support UC-2 (Login) and UC-18(Bill Payment), which is associated with QA-3 (Security).

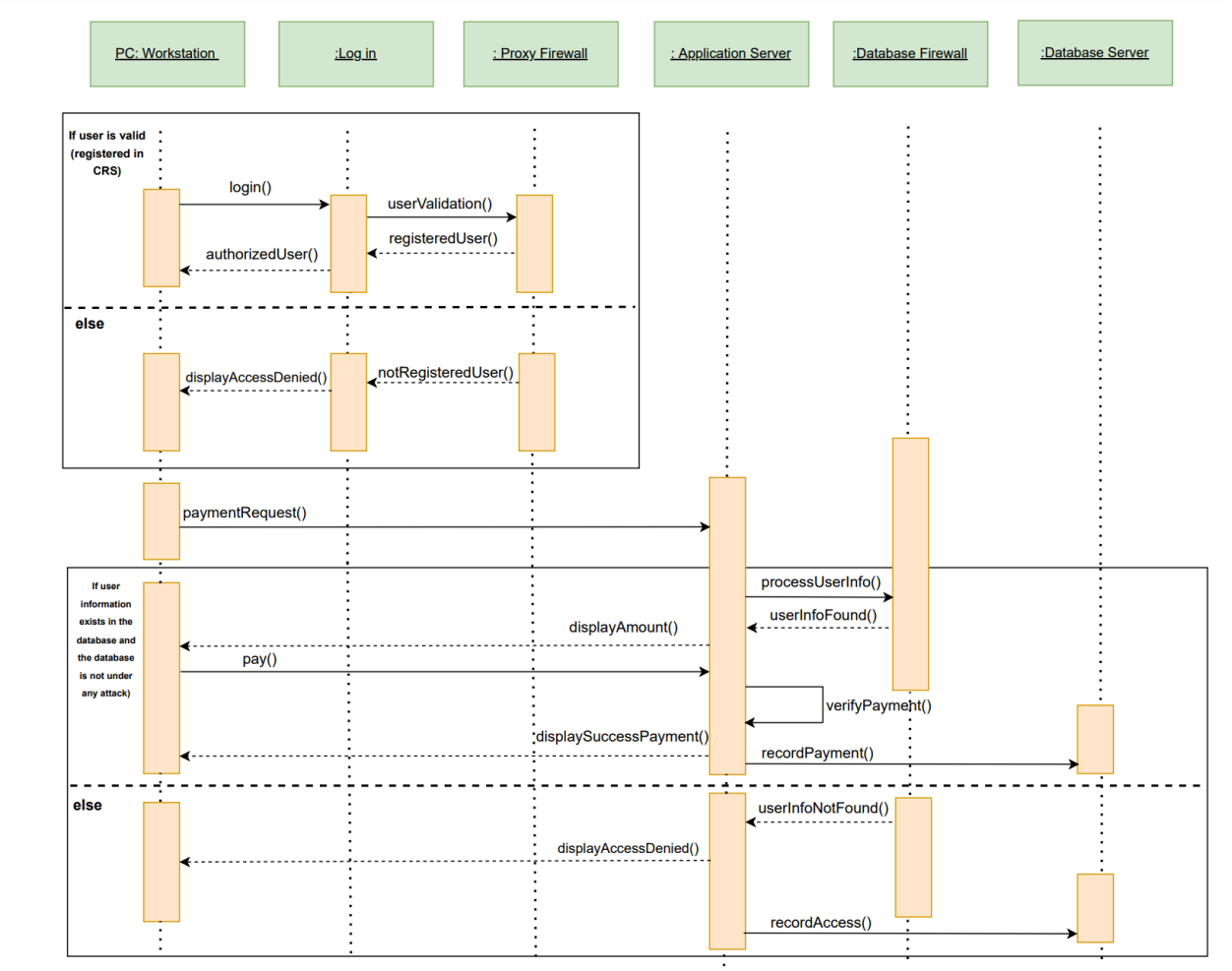

FIGURE 12.0 Sequence diagram illustrating the whole process to support the use cases UC-2 and UC-18 associated with QA-3.

Step 7: Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

In this iteration, crucial design decisions have been made to address QA-3. The table shown below summarizes the status of various drivers and the new decisions made during this iteration.

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During the Iteration
	QA-1		No relevant decisions were made.
		QA-3	Security quality attribute was carried out and addressed well in this iteration by implementing a proxy firewall between the client and application server and a database firewall between the application server and the database server.
	QA-6		No relevant decisions were made.
		CON-4	Due to the project manager, the timeline and project's progress is still on track and enough resources are available.
	CRN-5		No relevant decisions were made.