CMPE 281 Cloud Technologies

Robot Cloud (MRC)

Team - 17

Deliverable -2

Comp 4 – User Service and Billing Management

Girish Bisane (016650348)

Index

<u>User Service and Billing Management</u>

In a front-end application, this component's interface facilitates control of user services such user profiles, user histories, billing, and other user-related data. The user profile information, billing information, and booking history for security robots are all displayed in this component.

Purpose:

In order to employ campus security robots, this component processes, manages, and stores the data of end users who would use the complete system. The front end user interface for this project will allow robot owners or users to access the service, billing, tracking statuses, and other general services of a security robot. Users can extract data from ROS, save it in the DB, and then retrieve it to view data connected to various sensors on the dashboard.

Objective:

The goal of this module is to make it easy for clients to rent robots and see all the billing history. It also provide a dashboard for the transaction history of previous billings. This part's job is to send and receive data with the backend databases that power tracking and other information shown on the user interface (UI).

The management of users will also migrate to the cloud platform. Additionally, we will provide cloud platform services for robot users, including account management, catalog management, and billing for robot services.

The users we hope to attract include the following:

Robot users —> Students

Cloud Staff —> Collage Staff

Robot System Administrator —> Administrator

Function Scope

As the robot makes his way around the classroom, it ask students for their student ID numbers. Verify this SID's enrollment in this class in the database, validate it, and reply with an acknowledgment. It also notes whether or not the student was present that day as well. The robot can be configured to attend that class on a weekly basis at a specific hour. It can proceed to the next class once it has visited every seat in the previous one. A security robot's most frequent and fundamental task is to keep watch over a certain region when nobody is around.
The following is a list of qualities that a self-operating mobile security robot must possess.

Four categories of needs exist:
Controle: A robot's hardware consists of its electric power sources, cameras, computers, and engines. The first and most important stage of development is this.

Task: The security robot simulator follows the security guard's instructions and completes the task. There may be more than one component to a task, but they all work together to complete it. Make a patrol schedule, search for those who shouldn't be there, explore an unfamiliar area, and perform a watchdog assignment.

Function: The fundamental "tasks" that a robot must be able to perform are called operations. Billing, mapping, tracking, localization, motion planning (for a moving platform), and path planning are a few of the capabilities.

Scenario: The robot employs instances, which are collections of rules, to determine what to do in various circumstances. In order to accomplish tasks like show a visitor around, inform someone about an accident, or locate an invader, people and robots can collaborate. The activities a robot takes in response to events in its surrounding environment, such as a fire, a blackout, or an open door, are known as environment-robot interactions. High-level planning techniques and algorithms for making strategic judgments on which task function should be performed are necessary to cope with scenarios.

Usage

The Robot cloud platform's mission is to improve people's lives by making routine activities simpler, securing college campuses, and utilizing a variety of remote robot services. In order to mimic robots, AWS Robomaker will be used, and SQL DB will be used to store user profiles and details about registered robots. To access the various records and data for robot services, we will use a NoSQL database. Making sure our customers are satisfied is our first priority, and we always work to provide the best pricing we can. We will also be using STRIPE API to make successful transaction through our MRC website.

<u>**User Service and Billing Management- Interface, Design and Analysis**</u>

Component API Design:

Webservice to login to the user service interface for the security robot application.

```
POST <webservice URL>/Login
Request Body
Accept: {application/json}
Content-type: application/json
{
 "username":"girishbisane@securityrobot.com",
 "password":"CMPE281robocloudproject"
}
Response Body
{
 “authentication_status”: “success/failure”
}
```

API to fetch user profile details.

```
POST <webservice URL>/userProfile
Request Body
Accept: {application/json}
Content-type: application/json
{
 "user_ID":"girish124"
}
Response Body
```

```
{
“first_name” : “Girish”,
“last_name” : “Bisane”,
“DOB” : “12/11/1994”,
“email” : girishbisane@gmail.com,
“phone” : “+6696493871”,
“address” : “1302 The Alameda, San Jose”,
“state” : “CA”,
“zip” : “95126”
}
```

API to fetch billing history

```
POST <webservice URL>/user_order_history
Request Body
Accept: {application/json}
Content-type: application/json
{
 "user_Id":"girishbisane",
}
Response Body
{
 “billing” : [
“Date” : “10/11/2009”,
“Time”:”10:40”,
“Robot Type” : “Max-341”,
“Serial No” : “32141A”,
“Total Amount”:”1200”,
“Payment Type”:”COD”
}
```

Endpoints:

Endpoint	Method	Description	Parameters
/test	GET	Test if API is alive	N/A
/user	GET/POST	Get a list if registered users/post a new user	N/A /user info in Json format
/users/login	POST	Check login	User_name and

		credentials for users	password info in JSON
/users/{user_name}	GET/DELETE	Get user info/ remove for a specific user	user_name
/users/{user_name}	PUT	Update user info	User_name and user info in JSON
/admins	GET/POST	Add new admin /Get a list of admins	N/A /Admin info in JSON format
/admins/login	POST	Check admin login credentials	Admin_name and password info in JSON format
/robot	GET/POST	Add new car/Get a list of cars	N/A / robot info in JSON format
/robot/{robot_id}	GET/DELETE	Get car info /remove a specific robot	robot_id
/robot/{robot_id}	PUT	Update robot info	Robot_id and robot info in JSON format
/robot/available	GET	Get a list of robot available for booking	N/A
/bookings	GET	Get booking details	N/A
/bookings/{user_name}	POST	Update clients name who requested for MRC	NA
/order	GET	Get billing history	user_id

Architecture Design and Analysis

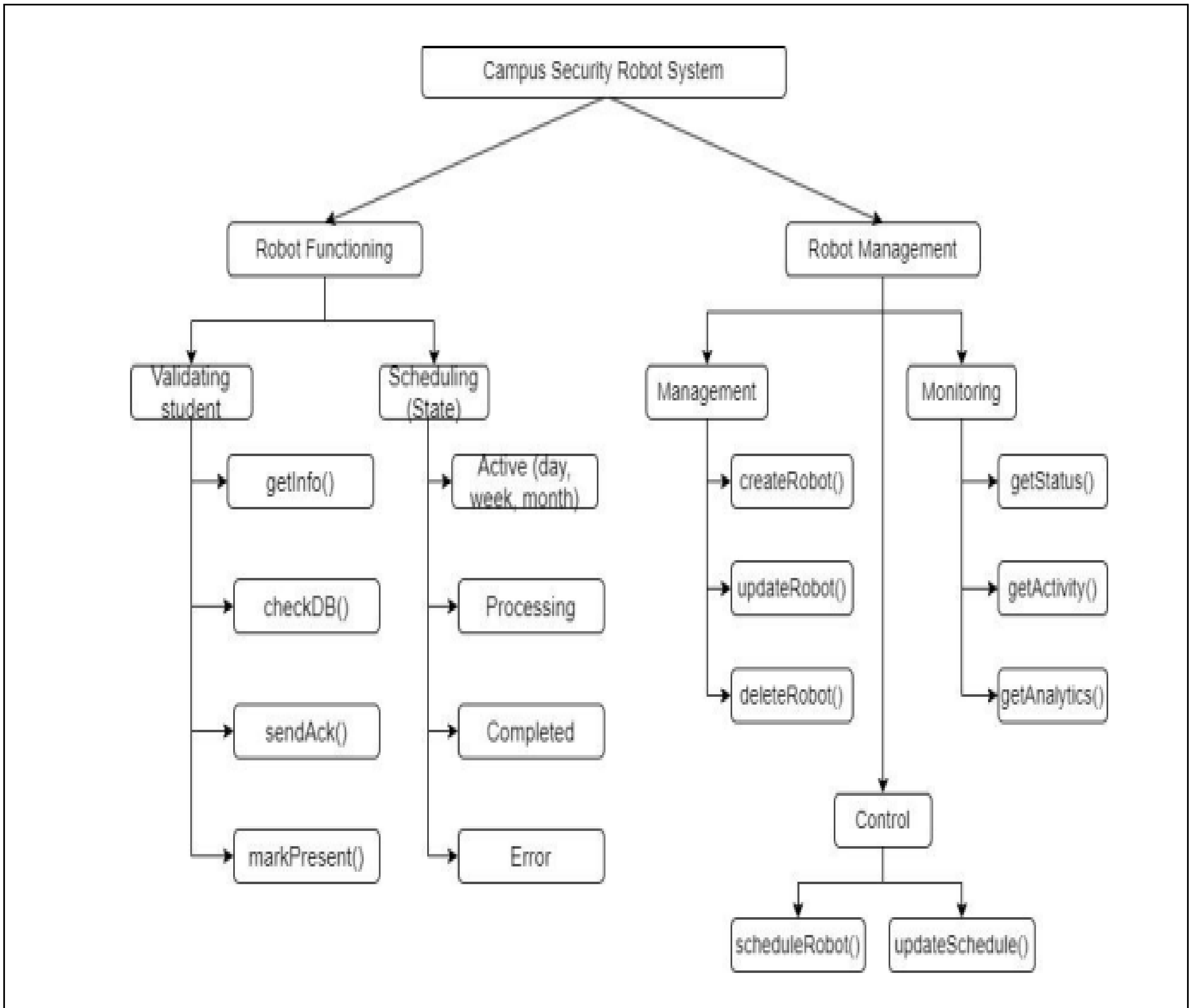
By demonstrating how the modules interact with one another, the graphic below demonstrates how redundancy is used. Network measurement packets for quality of service are also transmitted asynchronously between machines. To make things apparent, they are concealed. It is feasible to use several computing infrastructures. A mobile robot and a central server can communicate with one another via a wireless network. The robot has built-in motors and sensors. An alternate method of configuring a computer system is to communicate with a mobile robot across a wireless network using a secondary server. You can install redundant navigational modules on the mobile robot or the backup server. The DSS structure makes it simple to divide SRS services. One service is displayed.

To decide what to do, they text. Robot motor control is used to move a robot. The "Drive Distance" command instructs the robot to travel a specific distance. In a patrol situation, various items serve as fire indicators. Robot Simple Task performs a completion check for each task. Robot Simple Task receives virtual sensor data from the MRS, a laser range finder, or a bumper service. This enables it to determine what to do, such as the number of degrees to turn. The information reaches Robot Motor Control. The Robot Simple Task is made up of the sections Go To, Trace, Explorer, and Draw Map. Users can instruct "Robot Simple Task" using the GUI.

Planning determines where the security robot will patrol in the simulation, as well as if the strategy will be successful. A routine patrol route is planned using Routine Patrol, and an emergency situation—such as a break-in—is handled using Emergency Patrol.

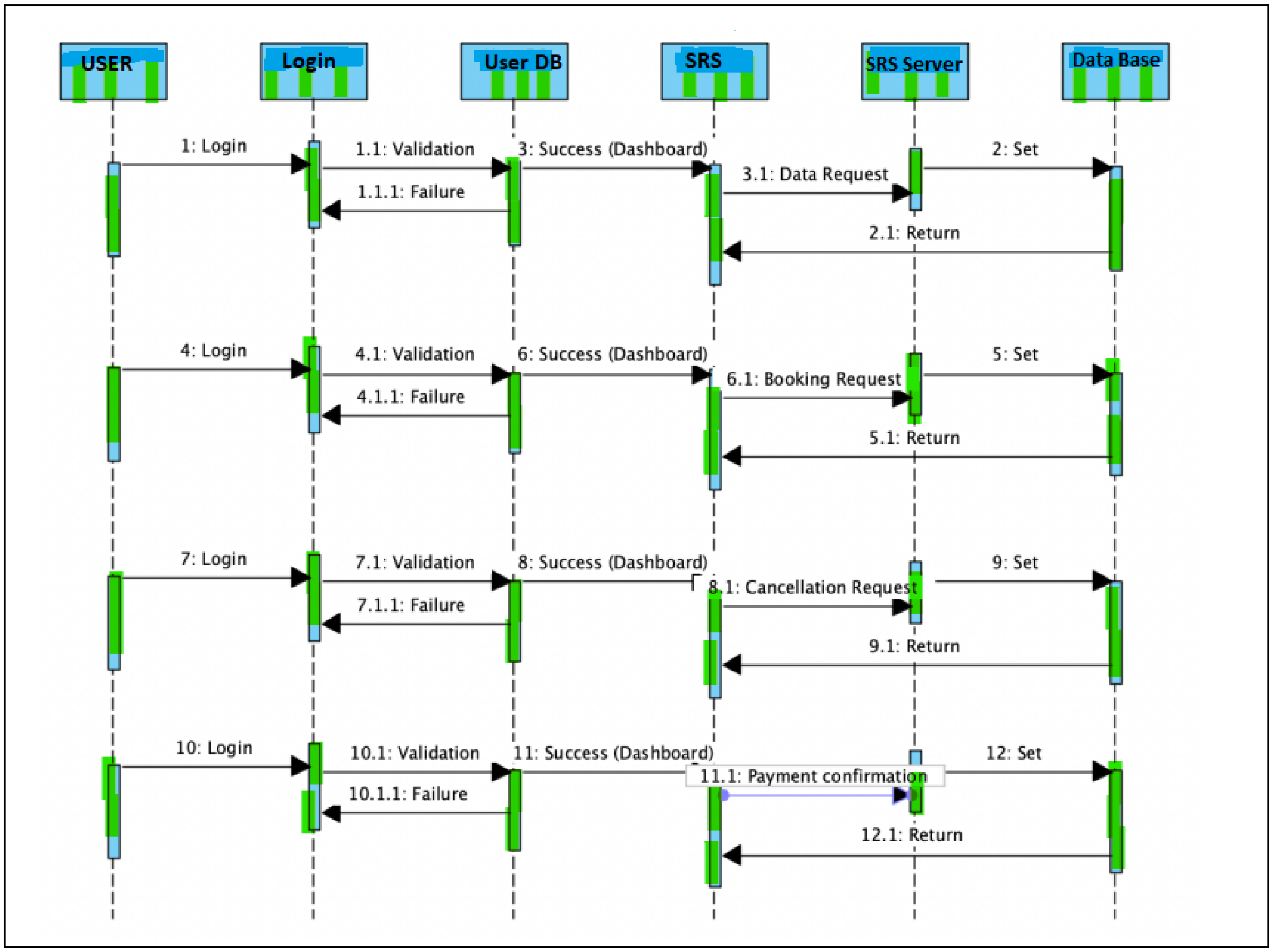
User Service and Billing Management- Function Design, Data and DB Design and Behavior Analysis

- **Campus Security Robot System Design**

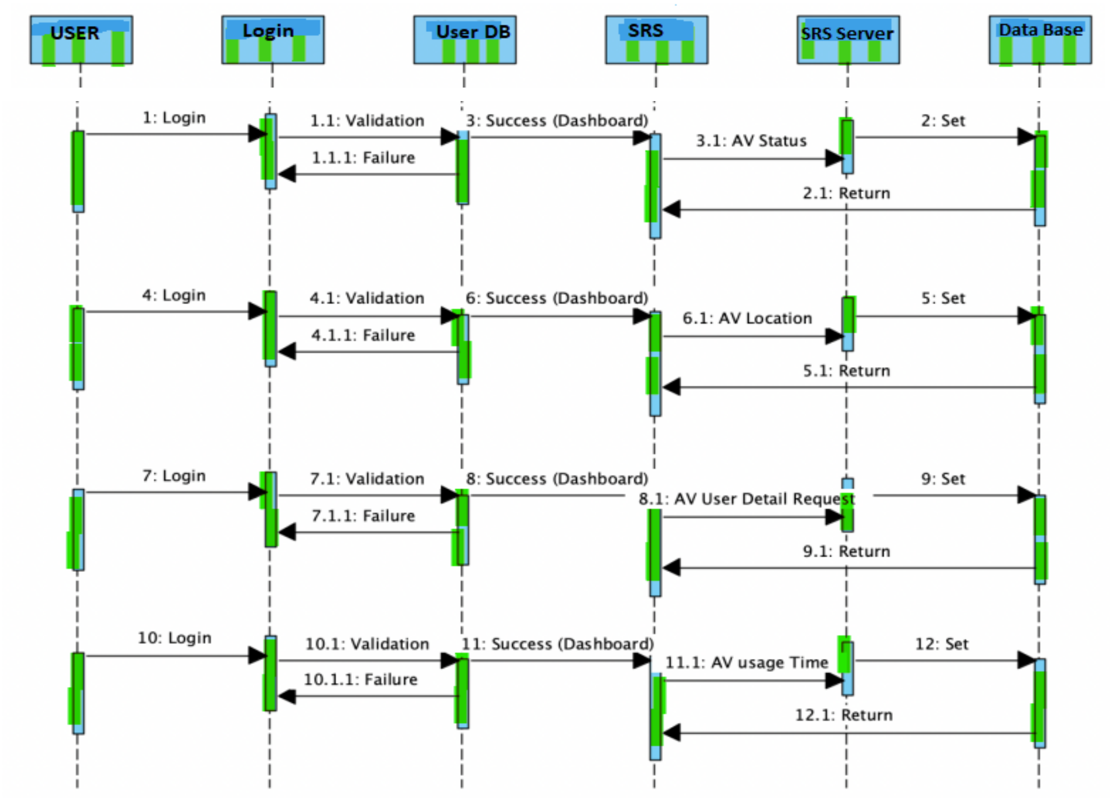


Sequence Diagram:

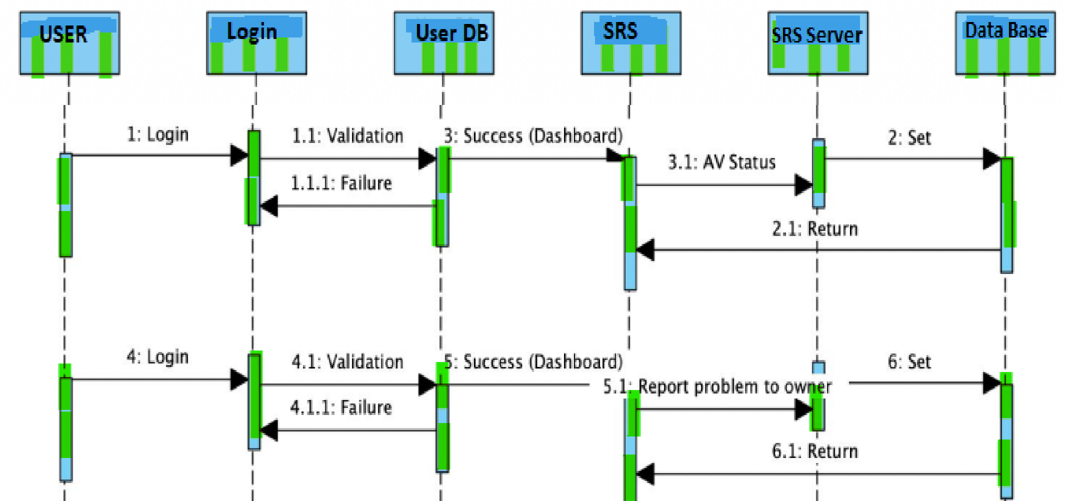
- **User**



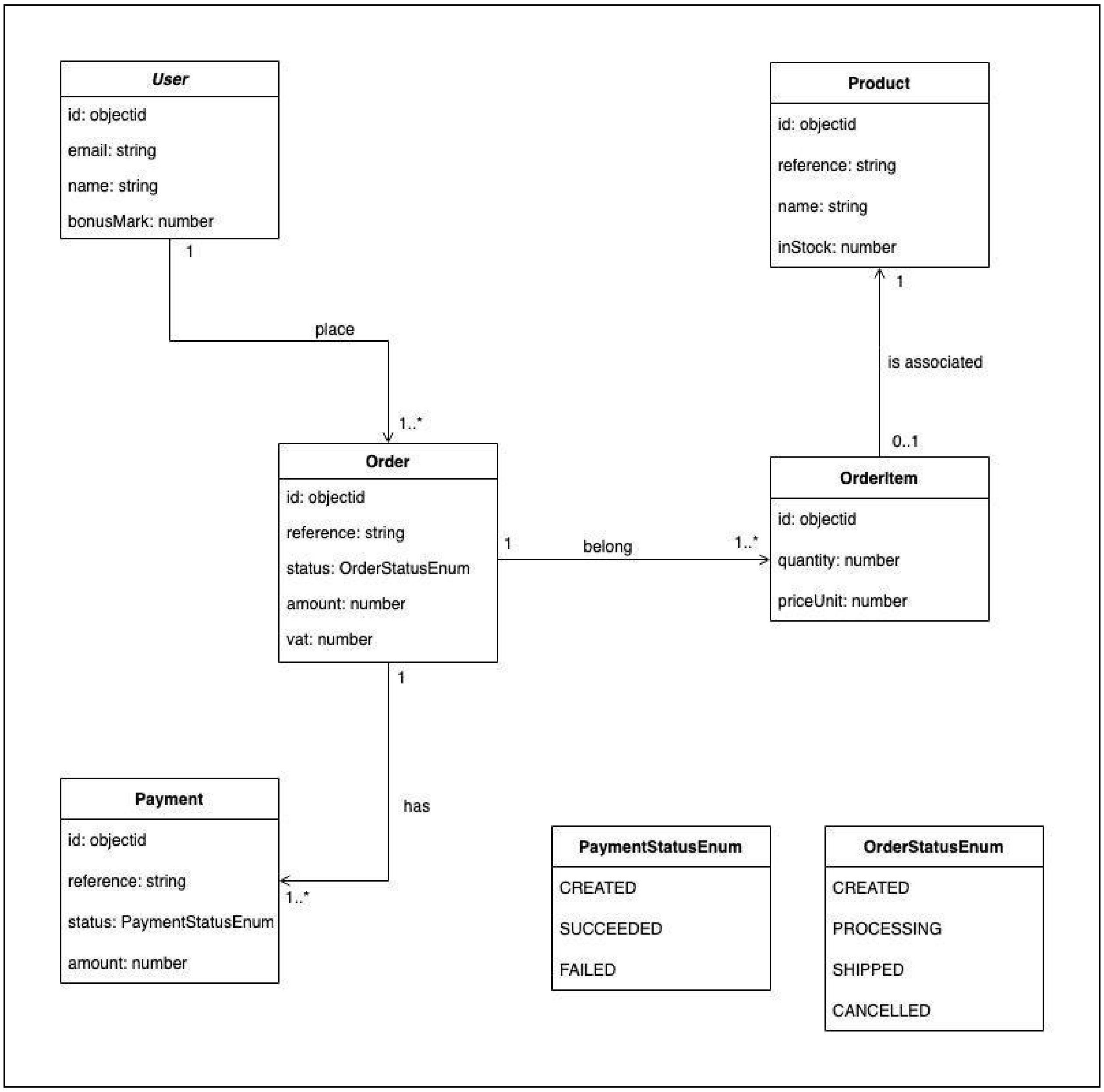
- **Owner**



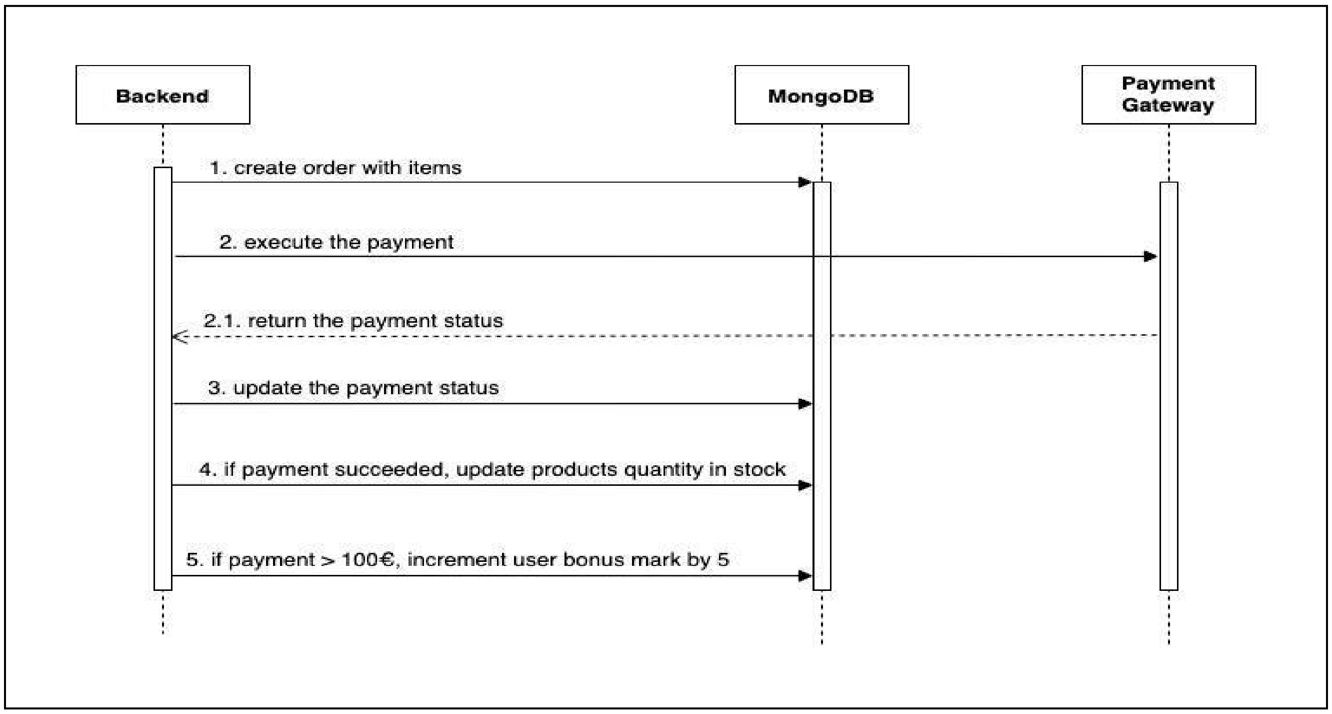
- **Manager**



- **Security Robot Payment Class Diagram**



- **Payment Sequence Diagram**


Backend
MongoDB
Payment Gateway
1. create order with items
2. execute the payment
2.1. return the payment status
3. update the payment status
4. if payment succeeded, update products quantity in stock
5. if payment > 100€, increment user bonus mark by 5

User Service and Billing Management - Component Function and Business Logic

1. **Function name: user_login**

Parameters: Username, Password

Description: This Function is used to take in user credentials as input and authenticate
the user to provide access to the web portal

Pseudocode:

```
{
    Checks that username and password are Not NULL
        Checks that username already exists in database of user
Checks that password matches with user password attribute inside the database
                If True :
return Success }
```

2. **Function name: fetch_user_details**

Parameters: user_id

Description: This function is used to retrieve the relevant user information from the user database using the user id as input.

Pseudocode:

```
{
    Checks if user id is Not NULL
        Checks if user id already exists in user database
            Fetches all the user information from the database and returns it
 }
```

3. **Function name: retrieve_billing_history**

Parameters: order_id

Description: This function is used to obtain all of the user's prior orders by using the order id as input.

Pseudocode:

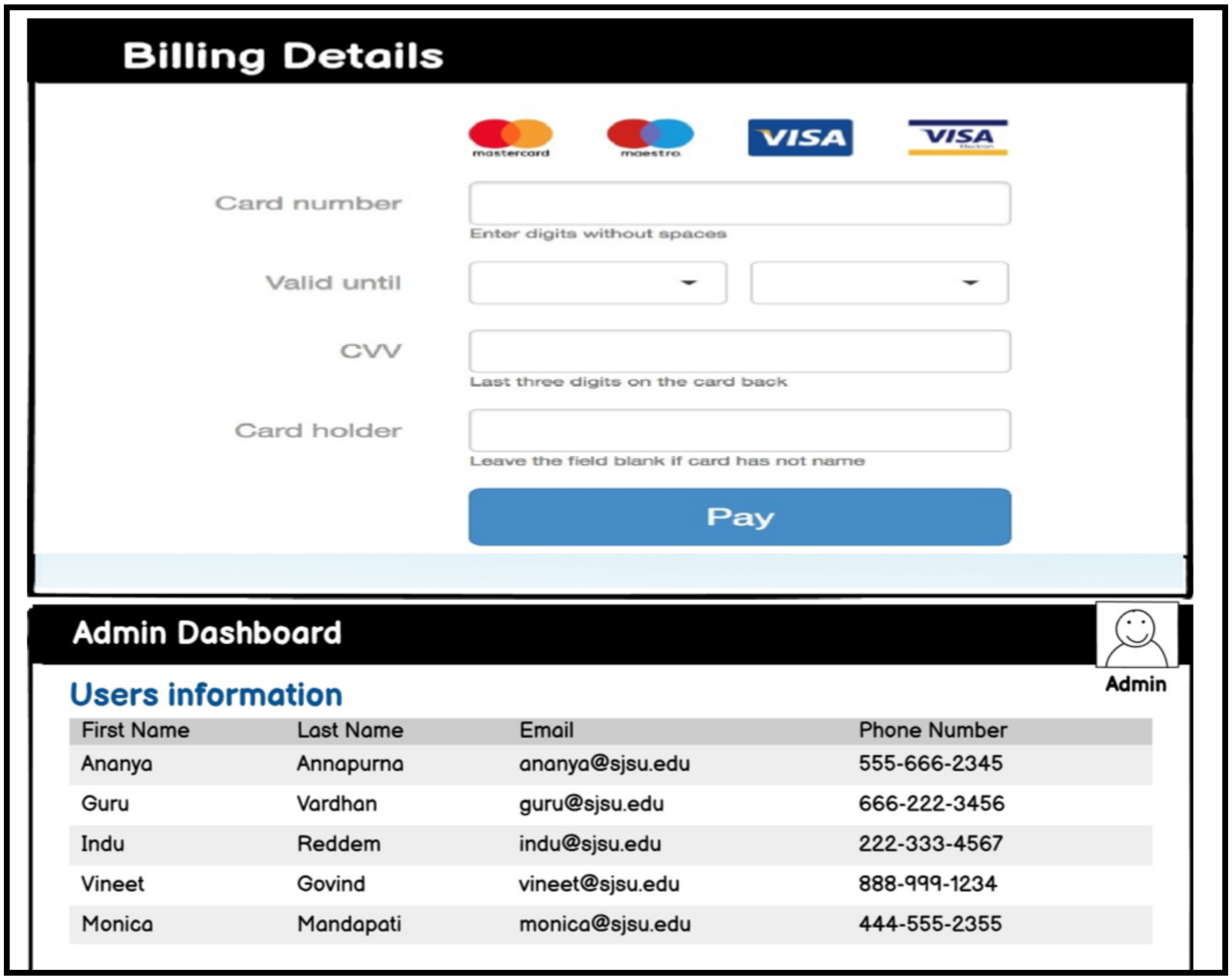
```
{
    Checks if order id Not NULL
        Check if username corresponding to the order id exists in order id database
            Check if user has made any previous orders in the past or is currently in process
                If True :
retrieve all the previous order history details of user and return it
 }
```

Component Business Logic -

The user's credentials are initially compared against the user id database to confirm user authentication. Upon verification, the user may log in. A user's profile information is retrieved from the user database and shown when they go to their profile page. The user participates in a variety of activities on the user profile page. Updates to user profiles are possible, and requests for setting changes made using the API are processed by the database.

The User may also review the order history and retrieve all the pending and prior order history by clicking on the Order History page, which will send an API request to the order history database to obtain the information and present it on the UI. The user can order from any open restaurant, and the order is processed and updated in the active order database before being returned to the user interface (UI) until it has been delivered. This information includes the order id, restraint address, destination address, destination type, payment method, delivery status, and total cost.

User Service and Billing Management Graphic User Interface Design Reference


Billing Details
mastercard
maestro
VISA
VISA
Card number
Enter digits without spaces
Valid until
CVV
Last three digits on the card back
Card holder
Leave the field blank if card has not name
Pay
Admin Dashboard
Admin

Users information

First Name	Last Name	Email	Phone Number
Ananya	Annapurna	ananya@sjsu.edu	555-666-2345
Guru	Vardhan	guru@sjsu.edu	666-222-3456
Indu	Reddem	indu@sjsu.edu	222-333-4567
Vineet	Govind	vineet@sjsu.edu	888-999-1234
Monica	Mandapati	monica@sjsu.edu	444-555-2355