

SDD-05

SOFTWARE DESIGN DOCUMENT

Tukang Parkir

Client:

Transmart Buah Batu

Prepared by:

Department Store Parking Control System (Group 5)

Jauza Zahra Ulaya (1301224452)

Ebrahim Abdul Hamid Nakhwa (1301224493)

Muhammad Akmal Mutohar (1301224209)


Agam Marichal Daniswara Aranda (1301224330)

Grizelda Audria Wijaya (1301224249)

Bachelor of Informatics

Faculty of Informatics

Jl. Telekomunikasi 1, Dayeuhkolot Bandung

	Bachelor of Informatics Telkom University	Document Number		Number of Pages
		<i>SDD-xx</i>		<#>
		Revision	<revision number>	<i>Date: <date></i>

List of Revisions

Revision	Description
A	
B	
C	
D	
E	
F	
G	

DATE INDEX	-	A	B	C	D	E	F	G
Revised by								
Checked by								
Approved by								

List of Revised Pages

Page	Revision	Page	Revision

Table of Content

Table of Content.....	5
List of Tables.....	6
List of Figures.....	7
1. Introduction.....	8
1.1 Aims.....	8
1.2 Problem scope.....	8
1.3 Terms and Definitions.....	8
1.4 References.....	8
2 General Design (High-Level).....	9
2.1 Software Implementation Environment Plan.....	9
2.2 Software Architecture Description.....	9
3 Detailed Design (Low-Level).....	11
3.1 Use Case Realization.....	12
3.1.1 Use Case #1 <Login>.....	11
3.1.1.1 Use Case Scenario #1 <Login>.....	11
3.1.1.2 UI Design and UI Object Description #1 <Login>.....	12
3.1.1.3 Object Identification and Class Type #1 <Login>.....	13
3.1.1.4 Sequence Diagram #1 <Login>.....	23
3.1.2 Use Case #2 <SignUp>.....	24
3.1.2.1 Use Case Scenario #2 <SignUp>.....	24
3.1.2.2 UI Design dan UI Object Description #2 <SignUp>.....	24
3.1.2.3 New Object Identification & Class Type #2 <SignUp>.....	29
3.1.2.4 Sequence Diagram #2 <SignUp>.....	29
3.1.3 Use Case #3 <Reserve Parking Slot>.....	30
3.1.3.1 Use Case Scenario #3 <Reserve Parking Slot>.....	30
3.1.3.2 UI Design dan UI Object Description #3 <Reserve Parking Slot>.....	30
3.1.3.3 New Object Identification & Class Type #3 <Reserve Parking Slot>.....	38
3.1.3.4 Sequence Diagram #3 <Reserve Parking Slot>.....	38
3.1.4 Use Case #4 <Monitoring Parking Status>.....	39
3.1.4.1 Use Case Scenario #4 <Monitoring Parking Status>.....	39
3.1.4.2 UI Design dan UI Object Description #4 <Monitoring Parking Status>.....	39
3.1.4.3 New Object Identification & Class Type #4 <Monitoring Parking Status>.....	43
3.1.4.4 Sequence Diagram #4 <Monitoring Parking Status>.....	44
3.1.5 Use Case #5 <Accept Offer>.....	44

3.1.5.1	Use Case Scenario #5 <Accept Offer>.....	44
3.1.5.2	UI Design dan UI Object Description #5 <Accept Offer>.....	45
3.1.5.3	New Object Identification & Class Type #5 <Accept Offer>.....	49
3.1.5.4	Sequence Diagram #5 <Accept Offer>.....	50
3.2	Overall Class Diagram.....	50
3.3	Detailed Class Design.....	51
3.4	Algorithm and/or Query Design.....	58
4	Requirement Traceability Matrix.....	64

After the Table of Contents, you may include a List of Tables and a List of Figures

List of Tables

- 1.2 Software Architecture Description
- 2.1 Use Case Realization
- 3.1

List of Figures

1. Introduction

1.1 Aims

The Software Design Document (SDD) for the Tukang Parkir Application aims to outline the architectural design, functionality, and technical aspects of the system's development. The primary purpose is to provide a comprehensive blueprint for the development team, guiding them through the software's structure, components, interfaces, and interactions. It serves as a reference for stakeholders to understand the system's design principles, aiding in decision-making, and ensuring alignment with project goals and requirements. This document acts as a technical guide, ensuring a clear understanding of the system's design and aiding effective collaboration among team members and stakeholders throughout the software development lifecycle.

1.2 Problem scope

Tukang Parkir is a user-friendly mobile application designed to streamline the parking experience at department stores. It offers a hassle-free solution for shoppers to easily locate available parking spots, reserve spaces in advance, and navigate to designated spots within the store premises. The application aims to alleviate parking congestion, enhance customer convenience, and optimize the overall parking management system for both shoppers and department store management. ParkEase integrates real-time data, user-friendly interfaces, and intuitive functionalities to revolutionize the parking experience, ensuring a seamless and stress-free process for all users.

1.3 Terms and Definitions

1. SDD: Software Design Document - A comprehensive document detailing the architectural design, components, and technical aspects of the software system.
2. SRS: Software Requirements Specification - A comprehensive document that outlines the functional and non-functional requirements of a software system.
3. UI: User Interface - The visual elements and interactive components through which users interact with the application.
4. UX: User Experience - The overall experience of a person using the application, encompassing usability, accessibility, and satisfaction.
5. IEEE: Institute of Electrical and Electronics Engineers - An organization that sets standards for various technological fields, including software engineering documentation.

1.4 References

IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.
SRS-005. Department Store Parking Control System. Tukang Parkir, 2023.

2 General Design (High-Level)

2.1 Software Implementation Plan

The software architecture for the "Tukang Parkir" mobile application will follow a typical three-tier architecture, consisting of the presentation layer, the application layer, and the data layer:

a. Presentation Layer:

- The front-end will be developed using a mobile application framework, such as React Native or Flutter, for cross-platform compatibility.
- The user interface will include screens for authentication, parking spot management, reservation, and payment processing.

b. Application Layer:

- The application layer will handle business logic and be an intermediary between the presentation and data layers.
- Key components include user authentication, reservation processing, and payment integration.
- This layer will communicate with the data layer to retrieve and update information.

c. Data Layer:

- The data layer will manage the storage and retrieval of data related to users, parking spots, reservations, and transactions.
- A relational database, such as MySQL or PostgreSQL, will be employed to store structured data efficiently.
- Object-relational mapping (ORM) tools may be used to interact with the database.

d. Communication:

- RESTful APIs or GraphQL will facilitate communication between the application and data layers.
- Secure protocols will be implemented to ensure the confidentiality and integrity of data during transmission.

e. Security:

- Security measures, including encryption for sensitive data, secure authentication mechanisms, and authorization controls, will be implemented to protect user information and transactions.

f. Scalability:

- The architecture will be designed with scalability in mind to accommodate potential growth in user base and data volume.
- Load balancing and caching strategies may be employed to optimize performance.

g. Integration:

- Integration points with external services will include payment gateways for transaction processing and push notification services for real-time updates.
- APIs will be leveraged to enable seamless communication with third-party systems.

h. Testing:

- Unit testing for individual components, integration testing for interactions between components, and end-to-end testing for overall system functionality will be conducted to ensure the reliability and robustness of the architecture.

i. Documentation:

- Comprehensive documentation will be provided for developers, outlining the architecture, API endpoints, data models, and any other relevant information.

This three-tier architecture provides a scalable, maintainable, and secure foundation for the "Tukang Parkir" mobile application, ensuring efficient communication between the user interface, business logic, and data storage components.

2.2 Software Architecture Description

No	Component Name	Description
1.	Authentication	Responsible for user registration, login, and authentication. Manages user-profiles and ensures secure access to the application.
2.	Parking Spot Management	Allows users to add, edit, and delete parking spots. Provides functionalities for searching and filtering available parking spots.

3.	Reservation	Handles the reservation process, allowing users to select and book parking spots. Manages reservation status and history.
4.	Payment Processing	Integrates with a secure payment gateway for processing reservations. Manages transaction details and ensures financial security.
5.	Database Access	Interacts with the database for storing and retrieving user data, parking spot details, reservations, and transaction information. Utilizes an ORM (Object-Relational Mapping) tool for efficient data access.
6.	Security	Implements security measures such as encryption for sensitive data. Ensures secure authentication and authorization processes.
7.	Scalability	Implements strategies for system scalability to handle increased user and data loads. Includes load balancing and caching mechanisms.
8.	Integration	Integrates with external services such as payment gateways and push notification services. Manages APIs for seamless communication with third-party systems.
9.	Testing	Includes unit testing for individual modules, integration testing for interactions between modules, and end-to-end testing for overall system functionality. Ensures the reliability and robustness of the architecture.
10.	Documentation	Provides comprehensive documentation for developers, outlining the architecture, API endpoints, data models, and other relevant information. Facilitates ease of understanding, maintenance, and future development.

3 Detailed Design (Low-Level)

3.1 Use Case Realization

Contains the USE CASE TABLE as follows:

No	Use Case Name	Use Case Description
#1	Login	The user shall be able to log into the application by entering valid credentials that are stored within the database or else the users wont be granted access and will be redirect to re entering their account credentials.
#2	Sign Up	The Sign Up make new users can create an account in the application by entering a username, password, and verifying with Gmail. This account is required to access the feature and services available in the application
#3	Reserve Parking Slot	The customer shall be able to reserve parking slot(s) in a department store. The customer can choose the parking slot(s) in a particular department store and pay them. After the transaction has been done, the system will update the parking slot availability and the parking slot has been completely reserved for the customer.
#4	Monitor Parking Status	The security personnel shall be able to choose the location including location of department store and the parking slot(s). After they choose the location, they can update the availability status.
#5	Accept Offer	The "Accept Offer" use case involves a staff member reviewing and responding to reservation offers made within the system. This use case allows the staff member to approve or deny reservation offers based on various criteria, such as availability, customer requests, system constraints or emergency situations..

3.1.1 Use Case #1 Login

3.1.1.1 Use Case Scenario #1 Login - Muhammad Akmal

i. Pre-Condition : The user has an account on the application

ii. Use Case Description

a) Primary Flow:

- The user accesses the system's login page.
- The system prompts the user to enter their credentials, including a username and password.
- The user enters the correct username and password.
- The system validates the entered credentials against the stored user information.
- If the credentials are valid, the system grants access to the user and proceeds to the main dashboard according to their roles.
- The main dashboard displays relevant information and options based on the user's role and permissions.

b) Alternative Flow:

1. Invalid Credentials:

- If the entered credentials are invalid, the system displays the following error message: "The username or your password is incorrect."
- The user is prompted to re-enter the correct username and password

2. Forgot Password:

- If the user forgets their password, they can click on the "Forgot Password" link.
- The system provides a password recovery option, such as sending a reset link to the user's registered email or SMS/WhatsApp.
- The user follows the provided instructions to reset their password

iii. Post-Condition:

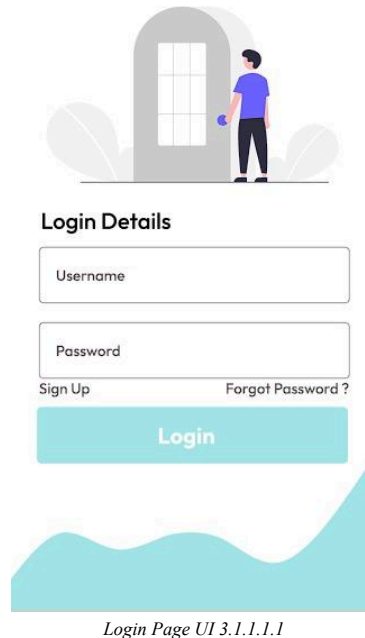
- Upon successful login, the system displays the order display or the relevant

landing page based on the user's role.

- The user has access to the features and functionalities associated with their role.
- The system maintains the user's session until signout.

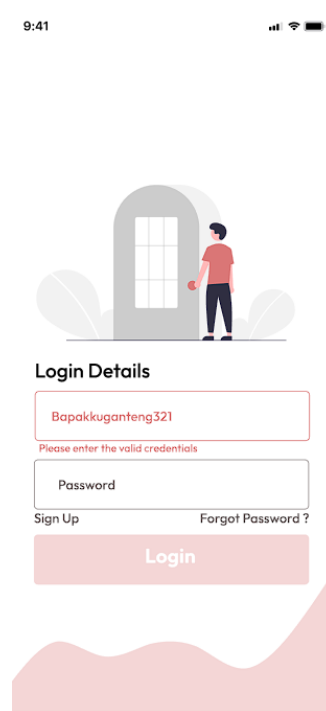
Table of UI Object Descriptions

SCREEN ID	SCREEN NAME	DESCRIPTION
999	Login page	In this login page, customer is asked to enter a valid credential to enter their respective account. else user can choose to click another button such as “Forgot Password” or “Sign up”.
998	Invalid login credential page	If user entered the invalid login credential, then the user will be redirected into this page and will be asked to re enter the password again.
997	Forgot password page	User will enter this page if they click on the “Forgot password” button. In this page, user will be asked to choose a recovery option either “Send code via E-mail” or “Send code via SMS”.
996	Recovery via email page	User will enter this page if they click on the “Send code via E-Mail” button. In this page, user will be asked to enter their email address.
995	Recovery via SMS page	User will enter this page if they click on the “Send code via SMS” button. In this page, user will be asked to enter their mobile phone number.
994	Enter verification code page	User will enter this page after they enter either email address or mobile phone number. in this page, user will be asked to enter a verification code that have been sent to their either email address or mobile phone SMS.
993	Reset Password page	If the verification code is valid, user will then be redirect into reset password page and will be asked to enter new password and confirm it.
992	Reset Password Success page	Will be displayed a message that tells that password reset is successful and user can click “Back to login page” button to get back to the first login page.
991	Customer dashboard page	After login, the customer dashboard, as shown in the first use case scenario above will appear for the customer.
990	Staff dashboard page	After login, the staff dashboard will appear for staff.
989	Security dashboard page	After login, the security dashboard will appear for the security personnel.



Object_ID	TYPE	LABEL*	Description**
3482	Text	Login Details	Display the words “Login Details”.
2206	Text Box	Username	User can input their Username credentials in this text box
2211	Text Box	Password	User can input their password credentials in this text box
3115	Button	Sign Up	If clicked, it will activate the method LoadSignUpPage().
3492	Button	Forgot Password?	if clicked, it will activate the method ClickForgotPassword()
2374	Button	Login	If clicked, it will activate EnterCredential() method

Invalid Login Credential - 998



Invalid Credential Page 3.1.1.1.2

Object_ID	TYPE	LABEL*	Description**
6645	Text	Login Details	Display the words “Login Details”,
5111	Text	Please enter the valid credentials	This message will appear if user enter the invalid credentials
1331	Text Box	Username	User can input their Username credentials in this text box
3132	Text Box	Password	User can input their password credentials in this text box
6161	Button	Sign Up	If clicked, it will activate the method LoadSignUpPage().
4752	Button	Forgot Password?	if clicked, it will activate the method ClickForgotPassword()
8524	Button	Login	If clicked, it will activate EnterCredential() method

Forgot Password Page - 997

Choose Recovery Option



Send code via E-mail

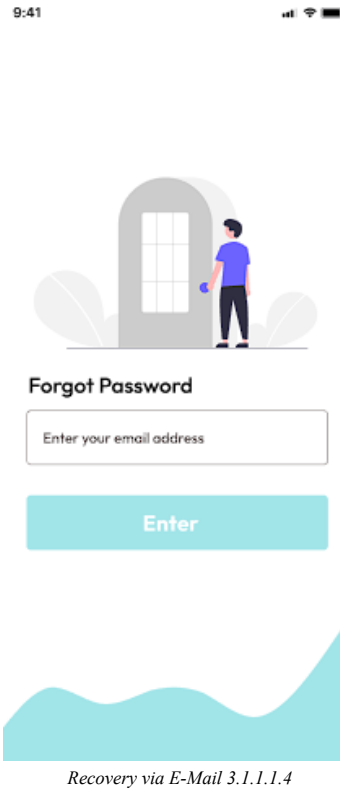


Send code via SMS

Forgot Password Page 3.1.1.1.3

Object_ID	TYPE	LABEL*	Description**
6611	Text Box	Password	User can input their password credentials in this text box
6524	Button	Send code via E-Mail	if clicked, it will activate the method InitiateReset()
2934	Button	Send code via SMS	If clicked, it will activate InitiateReset() method

Recovery via E-Mail - 996



Object_ID	TYPE	LABEL*	Description**
3253	Text\	Forgot Password	Displaying the word “Forgot Password”
3423	Text Box	Enter your email address	The user can input their email address here
2973	Button	Enter	If clicked, it will then redirect to verification code page

Recovery via SMS - 995



Forgot Password

Enter your mobile phone number

Enter



Recovery via SMS 3.1.1.1.5

Object_ID	TYPE	LABEL*	Description**
6351	Text	Forgot Password	Displaying the word “Forgot Password”
9453	Text Box	Enter your mobile phone number	The user can input their mobile phone number here
6161	Button	Enter	If clicked, it will then redirect to verification code page

Enter The Verification Code Page - 994

Enter Verification Code



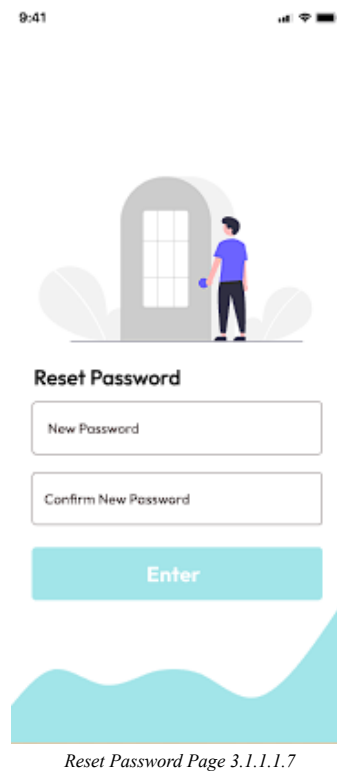
Enter



Enter Verification Code Page 3.1.1.1.6

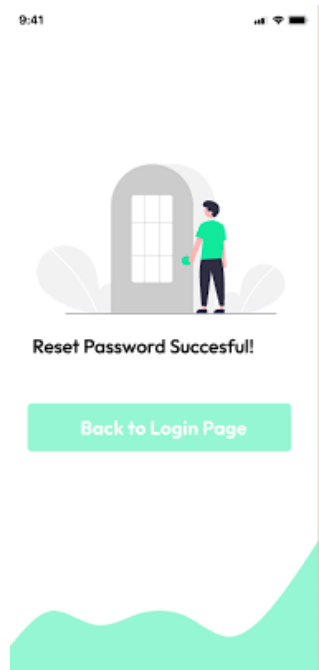
Object_ID	TYPE	LABEL*	Description**
3213	Text Box1	...	User can enter the first digit of verification code here
5115	Text Box2	...	User can enter the second digit of verification code here
6163	Text Box3	...	User can enter the third digit of verification code here
3642	Text Box4	...	User can enter the fourth digit of verification code here
4823	Button	Enter	If clicked, will then redirect user to the Reset Password Page

Reset Password - 993



Object_ID	TYPE	LABEL*	Description**
1568	Text Box	New Password	User can enter their desired new password here
6645	Text Box	Confirm New Password	User will need to reenter their new password to confirm it
6261	Text	Reset Password	Display the word "Reset Password"
2315	Button	Enter	If clicked, will then redirect user to the Reset Password Page

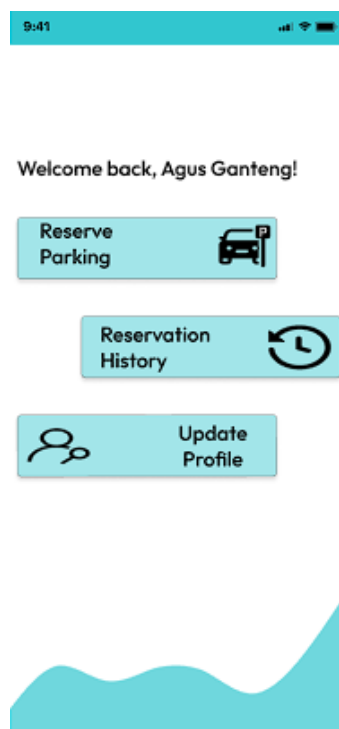
Reset Password Success - 992



Reset Password Success Page 3.1.1.1.8

Object_ID	TYPE	LABEL*	Description**
9921	Text Box	Enter your mobile phone number	The user can input their mobile phone number here
9922	Button	Back to login page	If clicked will redirect user back to login page

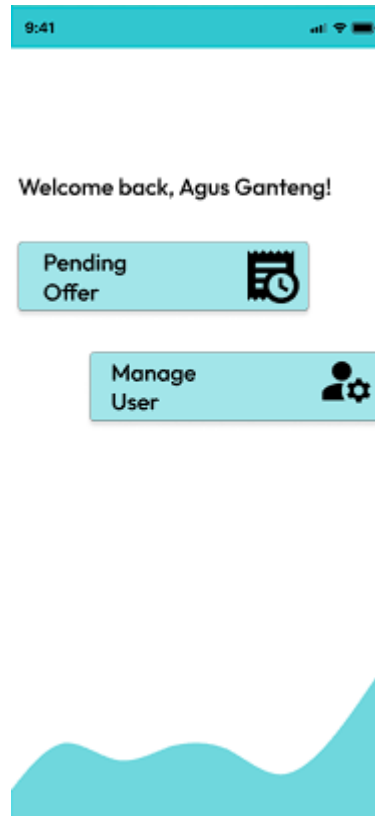
Customer Dashboard Page - 991



Customer Dashboard Page 3.1.1.1.9

Object_ID	TYPE	LABEL*	Description**
9911	Label	Welcome back, [username]!	Display the words “Welcome back, [username]!”, with the customer username in it.
9912	Hyperlink	Reserve Parking	If clicked, it will activate the method reserveParking().
9913	Hyperlink	Reservation History	If clicked, it will activate the method reservationHistory().
9914	Hyperlink	Update Profile	If clicked, it will activate the method updateProfile().

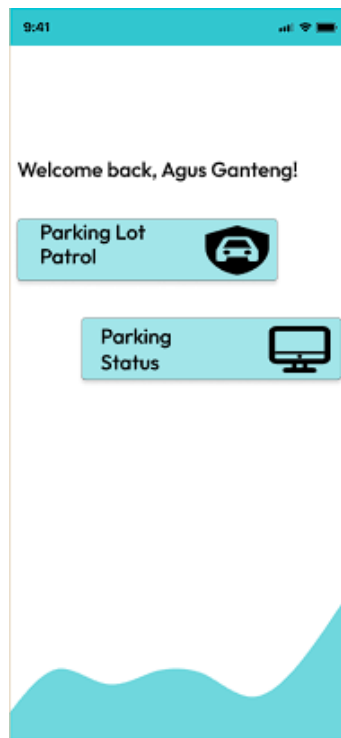
Staff Dashboard Page - 990



Staff Dashboard Page 3.1.1.1.10

Object_ID	TYPE	LABEL*	Description**
7699	Button	Pending offer	If clicked, it will redirect to pending offers page
7688	Button	Manage User	If clicked, it will redirect to manage user page
7598	Text	Welcome Back	Displays “Welcome Back, <username>”

Security Dashboard Page - 989



Security Dashboard Page 3.1.1.1.11

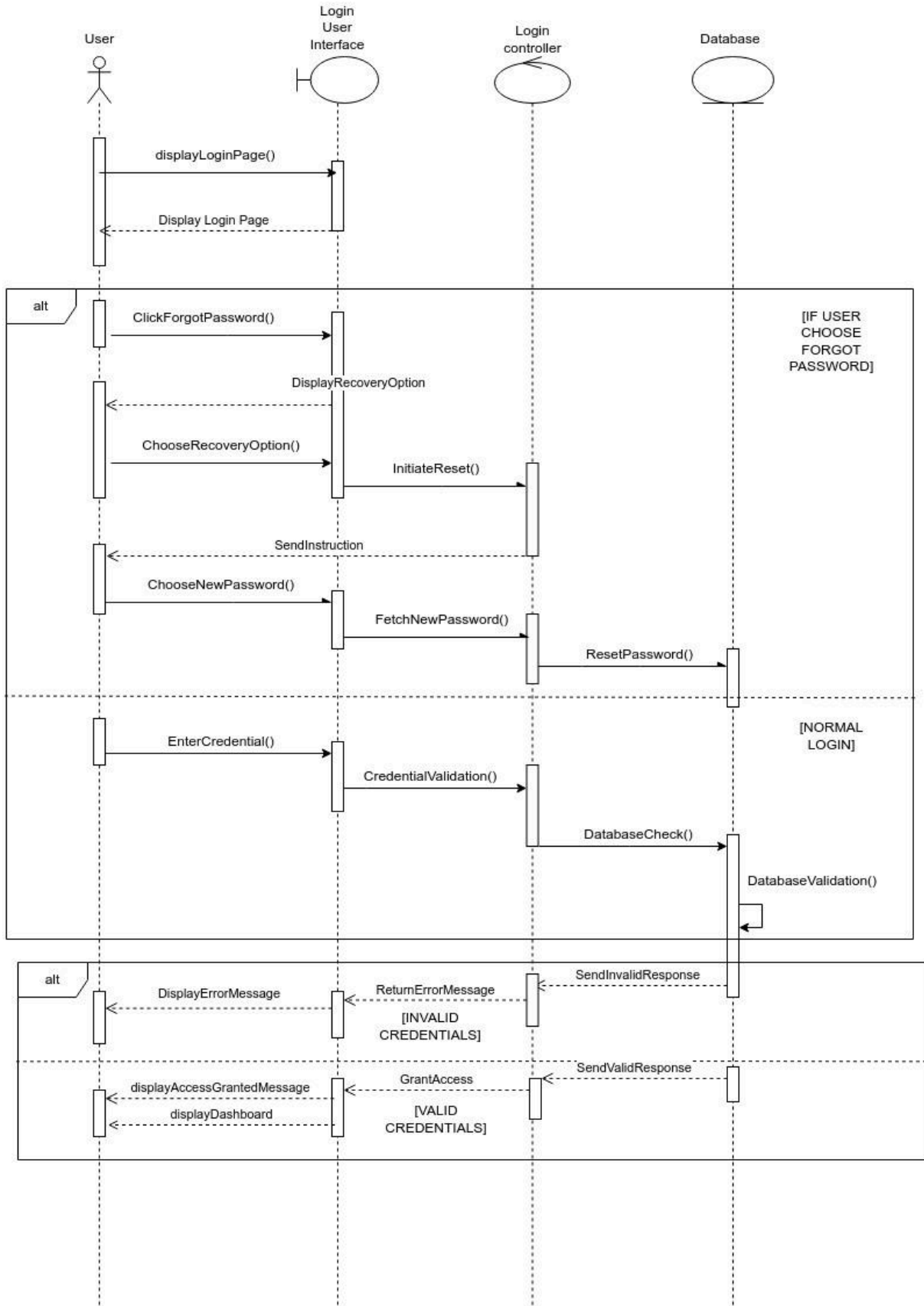
Object_ID	TYPE	LABEL*	Description**
Text1	Text	Welcome back, [username]!	Display the words “Welcome back, [username]!”, with the customer username in it.
Patrol1	Button	Parking Lot Patrol	If clicked, it will redirect to parking lot patrol page
ParkStatus1	Button	Parking Status	If clicked, it will redirect to monitor parking status page

3.1.1.3 Object Identification and Class Type #1 Login

TABLE OF DESIGN OBJECTS

No	New Object Name	Class Type
1.	User	Actor
7.	Login User Interface	Boundary (Interface)
8.	Login Controller	Control
9.	Database	Entity (Database)

3.1.1.4 Sequence Diagram #1 Login



3.1.2 Use Case #2 Sign Up

3.1.2.1 Use Case Scenario #2 Sign Up - Agam Marichal

i. Pre-Condition

- The system is installed and operational.
- The user has access to the sign-up page.

ii. Use Case Description

Primary Flow

- The user accesses the system's sign-up page
- The system prompts the user to provide necessary information, including a unique username, email address, and password.
- The user enters the required information.
- The system validates the entered data, ensuring that the username is unique, and the email address is properly formatted.
- If the entered information is valid, the system creates a new user account and stores the provided details.
- The system sends a confirmation email to the user's provided email address with a verification link.
- The user clicks on the verification link to confirm their email address and activate their account.
- After successful verification, the system displays a confirmation message, and the user can now log in using their credentials.

Alternative Flow

1) Username or email already in use:

- If the chosen username or email address is already associated with an existing account, the system prompts the user to choose a different username or use a different email address

iii. Post-Condition

- Upon successful sign-up and email verification, the user can log in using their newly created credential
- The system will redirect the user to the login page after successful verification

3.1.2.2 UI Design and UI Object Description #2 Sign Up

Table of UI Object Descriptions

SCREEN ID	SCREEN NAME	DESCRIPTION
	Sign Up Page	On the Sign Up page, user must enter a username, email address, password, and confirm password. after completing these field, they can click "Sign Up" for email verification
	Invalid Sign Up Page	New users are required to enter unique username and email address. if a user enters data that has already been used, they will be prompted to re-enter the data until they enter a unique value
	Verification Page	after signing up, user will be directed to the verification page. on this page, a verification code will be sent via the email link that they provided in the sign-up page. the user will have a limited amount of time to enter the verification code. if the code is correct, the user's account will be created and they will be directed to the login page
	Log In Page	On the login page, users enter their username and password, which they registered during sign-up. they can also click the "forgot password" or "Sign Up" button

Sign Up Page - 101

9:41

<

Create Account With Email

Username

Email

Password

Verify Password

Sign Up

Already have an account? [Log In](#)

Object_ID	TYPE	LABEL*	Description**
23455	Text	Create Account With Email	Display the word "Create Account With Email"
23943	Textbox	Username	the new user can input new username in this text box
23455	Textbox	Email	the new user can input their email in this textbox
43534	Textbox	Password	the new user can input the password in this textbox
34829	Textbox	Verify Password	the new user double check their password for verify
29374	button	Sign Up	if clicked will active the function CheckData()
42837	Text	Already have an account?	Display the word "Already have an account?"
28374	button	Log In	if clicked will active the function LoginRedirect()
23456	button	left arrow on corner	if clicked will active the function LoginRedirect()

Invalid Sign Up Page - 102

9:41

< Create Account With Email

Username
UdinParkir
The username already taken

Email
UdinParkir@gmail.com
The email already taken

Password
•••••

Verify Password
•••••

Sign Up

Already have an account? [Log In](#)

Object_ID	TYPE	LABEL*	Description**
2435	Text	Create Account With Email	Display the word “Create Account With Email”
6151	Text Box	Username	the new user can input new username in this text box
3131	Text Box	Email	the new user can input their email in this textbox
6131	Text Box	Password	the new user can input the password in this textbox
2613	Text Box	Verify Password	the new user double check their password for verify
51611	Button	Sign Up	if clicked will active the function CheckData()
31332	Text	The username already taken	Display the word “The username already taken”
303023456	Text	The email already taken	Display the word “The email already taken”
23456	text	already have an account?	Display the word “already have an account?”
2233	button	Log In	if clicked will active the function LoginRedirect()

Verification Page - 103

9:41

Enter verification code

We sent you verification code by email to sa*****@gmail.com. You have 5 attempts.

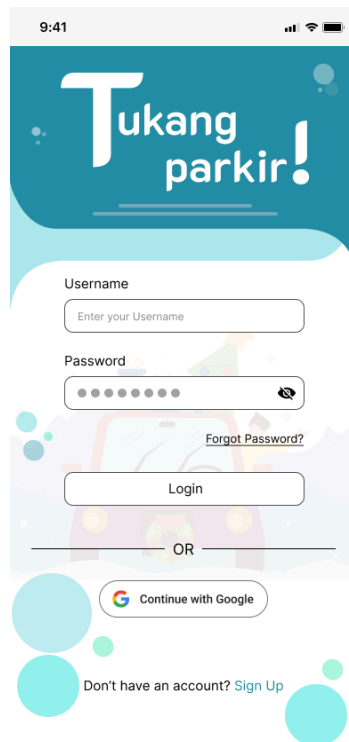
Verification Code

00 : 59

Continue

Resend Code

Object_ID	TYPE	LABEL*	Description**
56324	Text	Enter Verification code	Display the word “Enter verification code”
23456	text	we sent you verification code by email to ***@gmail.com. You have 5 attempt	Display the word “we sent you verification code by email to ***@gmail.com. You have 5 attempt”
34211	text	verification code	Display the word “verification code”
53220	box 1	...	User can enter the first digit of verification code here
3511	box 2	...	User can enter the second digit of verification code here
351320	box 3	...	User can enter the third digit of verification code here
3512	box 4	...	User can enter the fourth digit of verification code here
6130	box 5	...	User can enter the fifth digit of verification code here
234	text	00 : 59	Display the remaining time for enter the verification code
34330	button	Continue	If clicked, it will active the function LoginRedirect()
34566	button	resend code	if clicked, it will active the function SendVerificationLink()



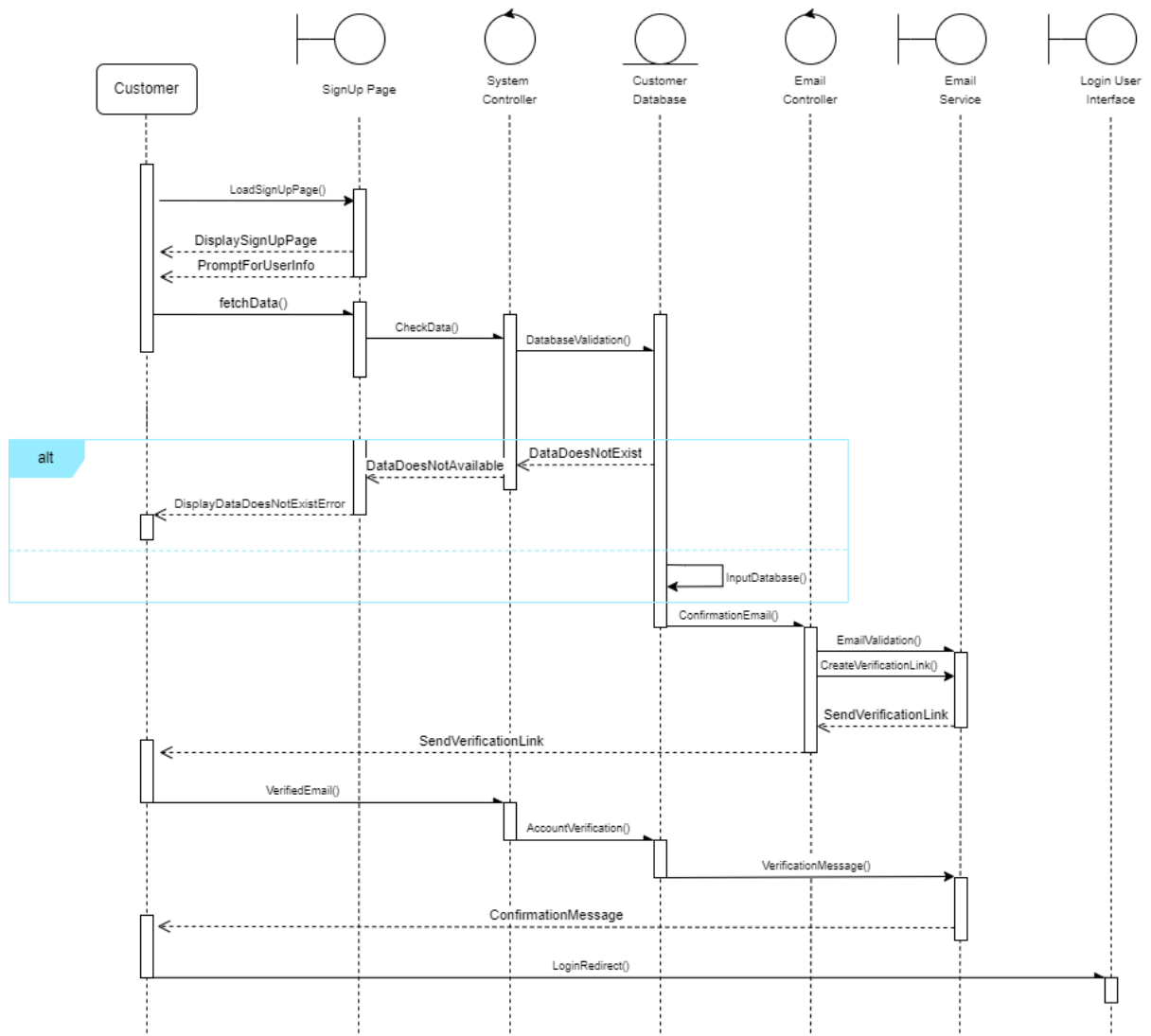
Object_ID	TYPE	LABEL*	Description**
87554	Text	Tukang Parkir!	Display the word “Tukang parkir!”
23469	Textbox	Username	the new user can input username in this text box
23456	textbox	Password	the new user can input their password in this textbox
46772	button	Forgot Password?	If clicked, it will active function ClickForgotPassword()
34456	button	Login	If clicked, it will activate EnterCredential() method
20646	text	OR	Display the word “OR”
62223	button	Continue with Google	If clicked, it will active function ConfirmationEmail()
21243	text	Don't have an account?	Display the word “Don't have an account?”
35223	button	Sign Up	If clicked, it will active function LoadSignUpPage()

3.1.2.3 Object Identification and Class Type #2 Sign Up

TABEL OF DESIGN OBJECTS

No	New Object Name	Class Type
01	Sign Up Page	Boundary (Interface)
02	System Controller	Controller
03	Customer Database	Entity (Database)
04	Email Controller	Controller
05	Email Service	Boundary (Interface)
06	Login User Interface	Boundary (Interface)

3.1.2.4 Sequence Diagram #2 Sign Up



3.1.3 Use Case #3 Reserve Parking Slot - Jauza Zahra

3.1.3.1 Use Case Scenario #3 Reserve Parking Slot

i. Pre-Condition

- The user is logged into the system.
- The user has a valid and confirmed payment method associated with their account.
- The system is operational and has up-to-date information on available parking slots.

ii. Use Case Description

a. Primary Flow

- The user navigates to the "Reserve Parking" section within the system.
- The system displays a list of available parking slots, organized by department stores.
- The user selects their desired department store from the list. The system presents a visual representation of available parking slots for the selected department store.
- The user chooses a specific parking slot from the available options.
- The system calculates the parking fee based on the selected slot, displaying the cost to the user.
- The user confirms the reservation and initiates the payment process.
- The system redirects the user to a secure payment gateway where they provide payment details.
- The payment gateway processes the transaction and notifies the system upon successful payment confirmation.
- The system updates the reservation status to "confirmed" and assigns the chosen parking slot to the user.

b. Alternative Flow

1) Payment Failure:

- If the payment transaction fails, the system informs the user about the issue.
- The user is prompted to review and correct payment details or choose an alternative payment method.

2) Parking Slot Unavailability:

- If the selected parking slot becomes unavailable during the payment process (e.g., due to simultaneous reservations), the system informs the user.
- The user can choose an alternative slot or proceed with a different reservation.

3) Reservation Cancellation:

- The user has the option to cancel the reservation before making the payment.
- If the user chooses to cancel, the system releases the selected parking slot back to the available pool.

iii. Post-Condition

- The user receives a confirmation message with details about the reserved parking slot and the corresponding parking fee.
- The system updates the parking slot status to "reserved" and associates it with the user's account.
- The user can view their parking reservation details in the system as a part of their reservation history, including the reserved slot, department store, and payment information.

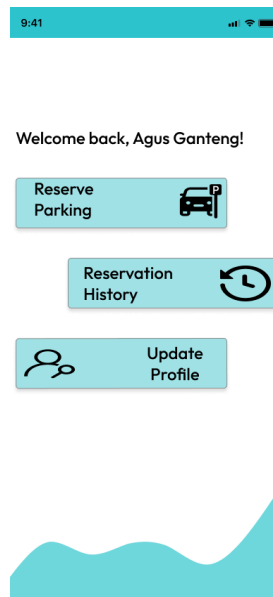
3.1.3.2 UI Design and UI Object Description #3 Reserve Parking Slot

Table of UI Object Descriptions

SCREEN ID	SCREEN NAME	DESCRIPTION
31119	Customer Dashboard	After login, the customer dashboard, as shown in the first use case scenario above will appear for the customer.
321	Reservation Page	The customer will need to choose the date, total slots, start hour, and finish hour for the parking slot that they want to reserve.
322	Cancel Reservation Page	There will be a message that the payment method is not valid. The customer ought to go back to the dashboard

		and change their payment method in order to proceed with the reservation.
323	Choose Loc DS Reservation Page	The customer will need to choose a particular department store in order to reserve the parking slot(s).
324	Choose Loc Park Reservation Page	The customer will choose the specific location of the parking lot.
325	Payment Page	After they choose the location, the customer will go to the payment page to proceed with the payment and complete the reservation. This page will show the total amount of the reservation and also the details. The customer can either cancel or proceed (confirm) the payment.
326	Cancel Payment Page	There will be a message about the customer wanting to cancel the payment. The customer ought to go back to the dashboard.
327	Disapprove Payment Page	There will be a message about a failed payment/transaction. The customer ought to go back to the dashboard. The failed payment will also be shown in the reservation history as a failed transaction.
328	Completed Payment Page	There will be a message that the reservation has been completed. The customer ought to go back to the dashboard. The successful payment will also be shown in the reservation history as a successful transaction.

Customer Dashboard



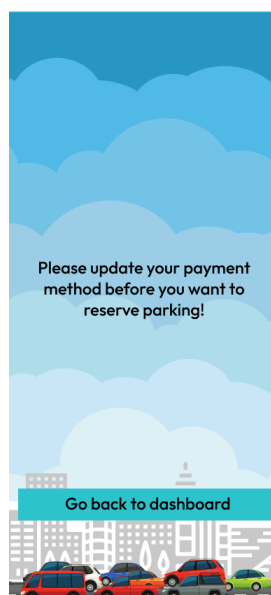
Object_ID	TYPE	LABEL*	Description**
9911	Label	Welcome back, [username]!	Display the words “Welcome back, [username]!”, with the customer username in it. (Welcome Message)
9912	Hyperlink	Reserve Parking	If clicked, it will activate the method reserveParking().
9913	Hyperlink	Reservation History	If clicked, it will activate the method reservationHistory().
9914	Hyperlink	Update Profile	If clicked, it will activate the method updateProfile().

Reservation Page - 321

The screenshot shows a mobile app interface for a reservation system. At the top, the status bar displays the time 9:41 and signal strength. The app title 'Reservation' is centered at the top. Below it, there are four input fields, each with a calendar icon and a label: 'Reservation Date', 'Total Vehicle', 'Start Hour (16.30)', and 'Finish Hour (19.30)'. A green rectangular box highlights these four input fields. Below the input fields is a green button labeled 'Find Location'. At the bottom of the screen, there is a decorative illustration of a city skyline with several cars parked in front of it.

Object_ID	TYPE	LABEL*	Description**
3211	TextBox	Reservation Date	The customer can input the reservation date.
3212	TextBox	Total Vehicle	The customer can input the total vehicle.
3213	TextBox	Start Hour	The customer can input when they want to start reserving the parking slot.
3214	TextBox	Finish Hour	The customer can input when they want to finish reserving the parking slot.
3215	Hyperlink	Find Location	If clicked, the function chooseLoc() will be activated and redirected to the Choose Loc DS Page if the customer already has the requirement (a valid payment method).
3216	Label	Reservation	Display the word “Reservation”, which means that the customer is already in the reservation page.

Cancel Reservation Page - 322



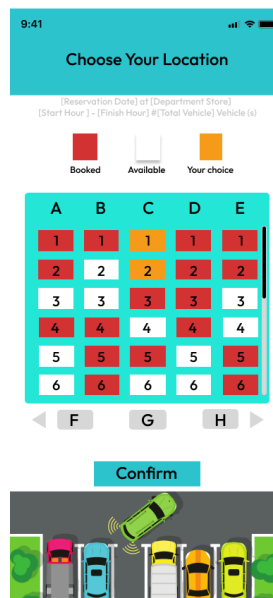
Object_ID	TYPE	LABEL*	Description**
3221	Hyperlink	Go back to dashboard	If clicked, it will go back to the dashboard.
3222	Label	Please update your payment method before you want to reserve parking!	Display the words “Please update your payment method before you want to reserve parking!”, which means the customer asked to update the payment before they reserve. (cancelReservationMessage)

Choose Loc DS Reservation Page - 323



Object_ID	TYPE	LABEL*	Description**
3231	Hyperlink	Park my car here	If clicked, it will redirect to the Choose Loc Park Page of that department store.
3232	Label	Choose Your Location	Display the words “Display Your Location”, which means the customer is already on the Choose Loc Page.
3233	Label	[Reservation Date]	It will show the chosen reservation date.
3234	Label	[Start Hour]-[Finish Hour] #[Total Vehicle] Vehicle(s)	It will show the chosen start hour, finish hour, and the total vehicle(s).
3235	Label	Paris Van Java	It will show the available department store (exp.Paris Van Java). (Department Store Name)
3236	Label	76 slots available	It will show the total slots that are available in that department store (exp. 76 slots are available). (Slots Available)
3237	Label	Price starts from	Displat the words “Price starts from”.
3238	Label	Rp. ##.000 / hour	It will show the parking slot reservation price per hour depending on the department store. (Price per hour)

Choose Loc Park Page - 324



Object_ID	TYPE	LABEL*	Description**
3241	Hyperlink	Confirm	If clicked, it will activate function <code>paymentStatus()</code> and will be redirected to the Payment Page with the chosen parking slot.
3242	Button	locNumber	If clicked, it will activate <code>chooseLoc()</code> for that specific parking slot.
3243	Button	locAlphabet	If clicked, it will show the parking slot of that alphabet location, the previous two, and the next two locations.
3244	Button	Right Arrow	If clicked, will display the next three alphabet locations
3245	Button	Left Arrow	If clicked, will display the previous three alphabet locations
3246	Label	Choose Your Location	Display the words "Choose Your Location", which means the customer is already in the Choose Loc Page.
3247	Label	[Reservation Date] at [Department Store]	It will show the chosen reservation date and the department store.
3248	Label	[Start Hour]-[Finish Hour] #[Total Vehicle] Vehicle(s)	It will show the chosen start hour, finish hour, and total vehicle.
3249	Label	Booked	It refers to the color information.
32411	Label	Available	It refers to the color information.
32412	Label	Your Choice	It refers to the color information.

Payment Page - 325

9:41

Payment

[Reservation Date] || [Start hour] - [Finish hour]

Department Store

#[Total Vehicle] Vehicle(s) Parking Slot(s) C1 - C2

Total Rp ##.000

Details

[Reservation Date] || [Start hour] - [Finish hour]

Department Store C1

#1 Parking Slot C1

Total Rp ##.000

[Reservation Date] || [Start hour] - [Finish hour]

Department Store C2

#2 Parking Slot C2

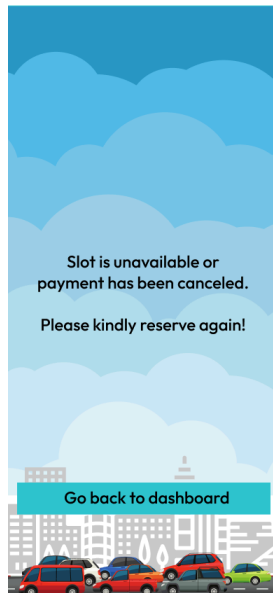
Total Rp ##.000

Cancel

Confirm

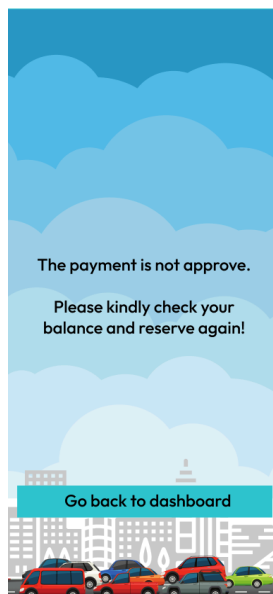
Object_ID	TYPE	LABEL*	Description**
3251	Hyperlink	Cancel	If clicked, it will redirect to the Cancel Payment Page.
3252	Hyperlink	Confirm	If clicked, it will activate the transaction() method and redirect to Completed Payment Page if the transaction is approved and will go to Dissapprove Payment Page otherwise.
3253	Label	Payment	Display the words “Payment”, which means that the customer is already in the Payment Page.
3254	Label	[Reservation Date] [Start Hour] - [Finish Hour]	It will show the reservation date, start hour, and the finish hour that the customer already fills in on the previous page.
3255	Label	Department Store	It will show the chosen department store by the customer.
3256	Label	#[Total Vehicle] Vehicle(s) Parking Slot(s) C1-C2	It will show the total slot(s) that the customer wants and the parking slot(s) location (exp. C1-C2). (Parking Location)
3257	Label	Total	Display the word “Total”.
3258	Label	Rp. ##.000	Display the total amount of the transaction. (Price)
3259	Label	Details	Display the word “Display”.
32511	Label	#1 Parking Slot C1	Display the chosen parking location for 1 slot (exp. 1 parking slot at C1). (Parking Location)
32512	Label	C1	Display the chosen parking location for 1 slot. (Parking Location)

Cancel Payment Page - 326



Object_ID	TYPE	LABEL *	Description**
3261	Hyperlink	Go back to dashboard	If clicked, it will redirect to the customer dashboard.
3262	Label	Slot is unavailable or payment has been canceled.	Display the words “Slot is unavailable or payment has been canceled.”, which means a failed transaction. (Cancel Payment Message)
3263	Label	Please kindly reserve again!	Display the words “Please kindly reserve again!”, which means the customer asked to reserve again in order to have the reservation. (Cancel Payment Message)

Disapprove Payment Page - 327



Object_ID	TYPE	LABEL*	Description**
3271	Hyperlink	Go back to dashboard	If clicked, it will redirect to the customer dashboard.
3272	Label	The payment is not approve.	Display the words “The payment is not approve.”, which means the transaction is failed. (Dissapprove Message)
3273	Label	Please kindly check your balance and reserve again!	Display the words “Please kindly check your balance and reserve again!”, which means the customer asked to reserve again with another payment method or correct balance. (Dissapprove Message)

Completed Payment Page - 328



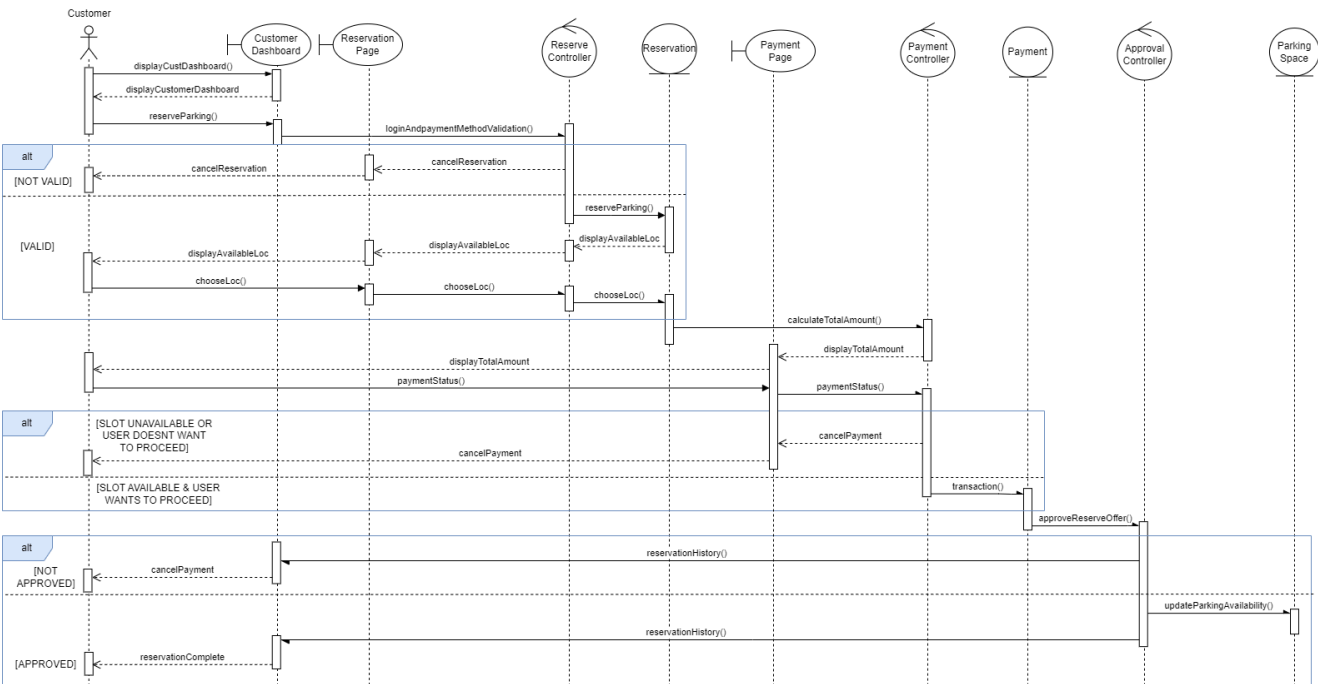
Object_ID	TYPE	LABEL*	Description**
3281	Hyperlink	Go back to dashboard	If clicked, it will redirect to the customer dashboard.
3282	Label	The parking slot(s) has been yours.	Display the words “The parking slot(s) has been yours.”, which means the parking slot(s) has been reserved. (Reservation Complete Message)
3283	Label	Thank you for choosing us and see you on the next reservation!	Display the words “Thank you for choosing us and see you on the next reservation!”, which is a closing statement. (Reservation Complete Message)

3.1.3.3 Object Identification and Class Type #3 Reserve Parking Slot

No	New Object Name	Class Type
1.	Customer Dashboard	Boundary (Interface)
2.	Reservation Page	Boundary (Interface)
3.	Payment Page	Boundary (Interface)
4.	Reserve Controller	Controller
5.	Payment Controller	Controller
6.	Approval Controller	Controller
7.	Reservation	Entity (Database)

8.	Payment	Entity (Database)
9.	Parking Space	Entity (Database)

3.1.3.4 Sequence Diagram #3 Reserve Parking Slot



3.1.4 Use Case #4 Monitoring Parking Status - Grizelda Audria W. (1301224249)

3.1.4.1 Use Case Scenario #4 Monitoring Parking Status

i. Pre-Condition

- The user has completed their parking session and left the parking lot.
- The security personnel responsible for updating parking lot availability has access to the system.
- The system is operational and accessible to security personnel.

ii. Use Case Description

a. Primary Flow

- The security personnel log into the system using their authorized credentials.
- The system displays a dashboard with options, including "Monitor Parking Status"
- The security personnel selects the "Monitor Parking Status" option.
- The system presents a list of parking lots organized by department stores.
- The security personnel selects a specific department store to check the parking lot availability.
- The system provides a real-time visual representation of the current occupancy status for each parking slot in the selected department store.
- The security personnel physically inspects the parking lot, updating the system with the current availability status.
- If there are changes in the availability (e.g., a parking spot becomes vacant), the security personnel updates the system accordingly.
- The system reflects the updated parking lot availability information in real-time.

b. Alternative Flow

1) Location Existence

- If the selected location is wrong or not available because the location is not on the list, the system informs error to the security personnel (user).

- If the selected location is exist, the security personnel can be update the parking slots availability.

iii. Post-Condition

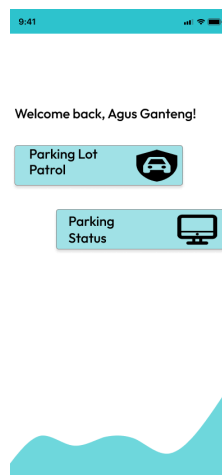
- The system displays the updated parking lot availability for the selected parking location/parking slots in department store.
- Users accessing the system for parking reservations can view real-time information about available parking slots based on the recent update.

3.1.4.2 UI Design and UI Object Description #4 Monitor Parking Status

Table of UI Object Descriptions

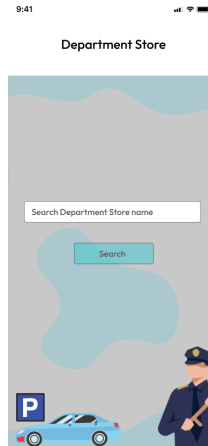
SCREEN ID	SCREEN NAME	DESCRIPTION
001	Security Dashboard	After login, the security dashboard will appear for the security personnel.
002	Select Departement Store Page	Security personnel will need to search Departement Store name.
003	Select DeptStore Error	This page will show the message that department store that security personnel choose not available in the list. It will show error page, security have to go back to the dashboard page.
004	Select Location Page	This page will show the location page because of department store that security personnel looking for is found. Security personnel need to search Parking Location Code.
005	Location Error Page	This page will show the message that parking location that security personnel looking for is not available. It will show error page, security have to go back to the dashboard page.
006	Update Page	Security personnel will need to click the “Yes” or “No” button to update the status availability. Security also need to input the time based on what time they change the update.
007	Updated Data Page	There will be a list of total amount availability status and total amount booked status, including parking location code and starting time availability and range time the parking lot booked.

Security Dashboard - 001



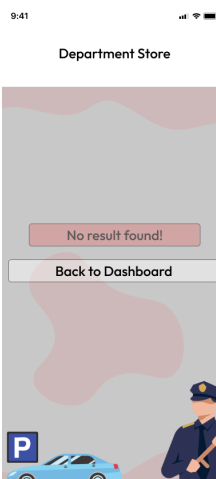
Object_ID	TYPE	LABEL*	Description**
1111	Text	Welcome back, [username]!	Display the words “Welcome back, [username]!”, with the customer username in it.
1112	Button	Parking Lot Patrol	If clicked, it will redirect to parking lot patrol page
1113	Button	Parking Status	If clicked, it will redirect to monitor parking status page

Select Departement Store Page - 002



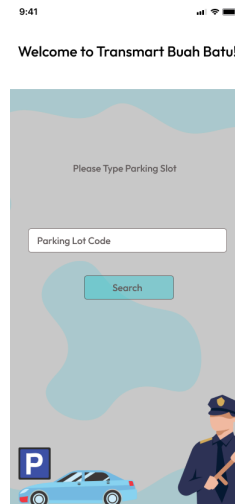
Object_ID	TYPE	LABEL*	Description**
2222	Text	Departement Store	Display the words “Departement Store”
2223	TextBox	Search Departement Store name	The security personnel can input the department store name
2224	Button	Search	If clicked, it will redirect to next page, either Select DeptStore Error page or select deptstore success page

Select DeptStore Error - 003



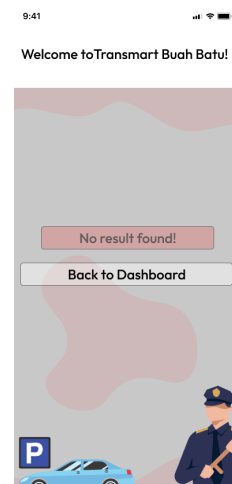
Object_ID	TYPE	LABEL*	Description**
3333	Text	Departement Store	Display the words “Departement Store”
3334	Text	No result found!	Display text “No Result Found!” means there are no data about what security personnel looking for.
3335	Button	Back to Dashboard	If clicked, it will redirect Security Dashboard page.

Select Location Page - 004



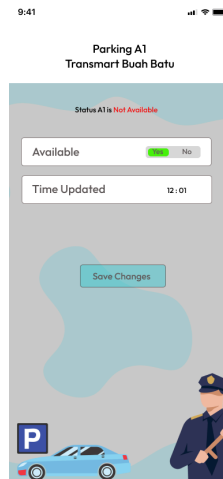
Object_ID	TYPE	LABEL*	Description**
4444	Text	Welcome to [Departement Store Name]!	Display the words “Welcome to [Departement Store Name]!” with the department store name in it.
4445	Text	Please Type Parking Slot	Display the words “Please Type Parking Slot”
4446	TextBox	Parking Lot Code	The security personnel can input the parking location code
4447	Button	Search	If clicked, it will redirect to next page, either Update page or Location Error page

Location Error Page - 005



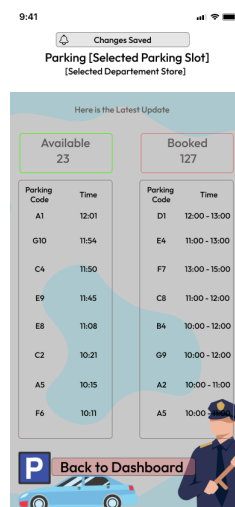
Object_ID	TYPE	LABEL*	Description**
5555	Text	Welcome to [Departement Store Name]!	Display the words “Welcome to [Departement Store Name]!” with the department store name in it.
5556	Text	No result found!	Display text “No Result Found!” means there are no data about what security personnel looking for.
5557	Button	Back to Dashboard	If clicked, it will redirect Security Dashboard page.

Update Page - 006



Object_ID	TYPE	LABEL*	Description**
834723	Text	Parking [Parking Code]!	Display the words “Parking [Parking Code]” with the parking code in it.
349238	Text	[Departement Store Name]	Display the words “[Departement Store Name]” with the department store name in it.
348923	Button	Yes No	If clicked, it will switch either it is “yes” or “no”
342873	TextBox	12:01	Security personnel can change or input the time when they update the parking lot data.
428374	Button	Save Changes	If clicked, it will save the input and redirect to the Update Data Page.

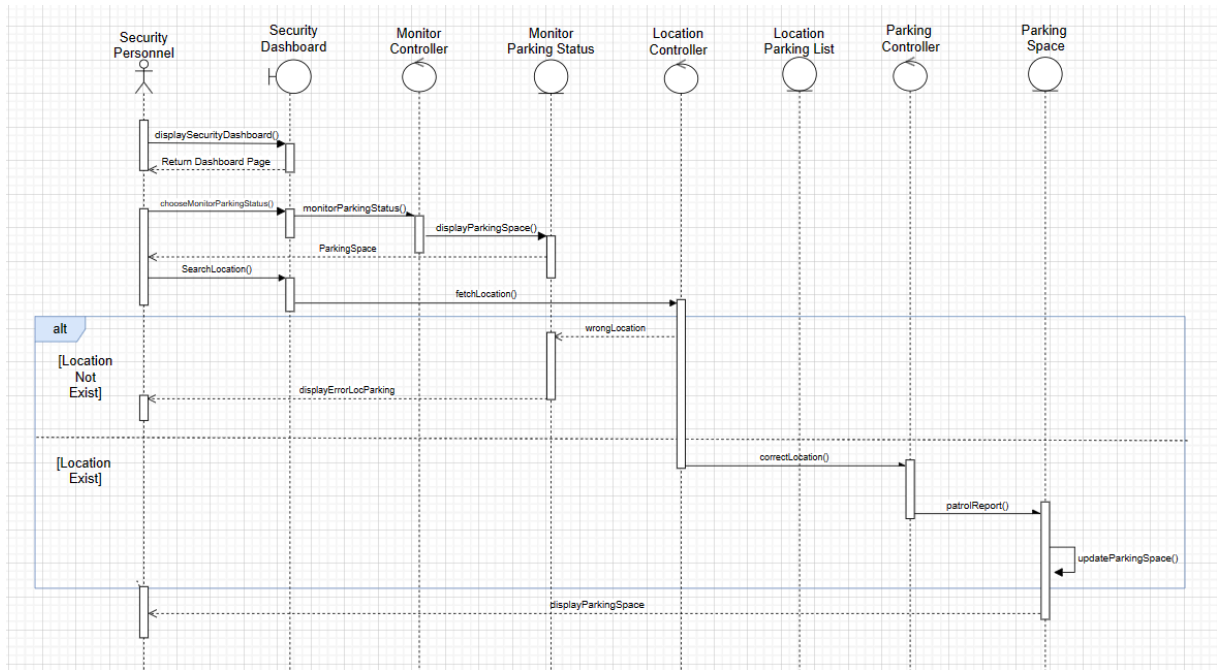
Updated Data Page - 007



Object_ID	TYPE	LABEL*	Description**
7777	Text	Changes Saved	Display text “Changes Saved” to inform security personnel that the last update is already saved.
7778	Text	[Selected Departement Store]	Display the words “[Selected Departement Store” with the department store name in it.
7779	Text	[Departement Store Name]	Display the words “[Departement Store Name]” with the department store name in it.
7770	Text	Here is the Latest Update	Display the words “Here is the Latest Update”
7776	Text	Available	Display the words “Available”
231234	Text	Booked	Display the words “Booked”
34111	Text	23	It will show the amount of available parking slot (exp. 23)
34255	Text	127	It will show the amount of booked parking slot (exp. 127)
1313	Text	Parking Code	Display the words “Parking Code”
18661	Text	Time	Display the words “Time”
3456	Text	A1	It will show parking slot code (exp. A1)
4283	Text	12:01	It will show the update data time(exp. 12:01)
7236	Text	12:00 - 13:00	It will show the booked range time (exp. 12:00 - 13:00)
4287	Button	Back to Dashboard	If clicked, it will redirect Security Dashboard page.

3.1.4.3 Object Identification and Class Type #4 Monitoring Parking Status

No	New Object Name	Class Type
1.	Security Dashboard	Boundary (Interface)
2.	Monitor Controller	Controller
3.	Monitor Parking Status	Entity (Database)
4.	Location Controller	Controller
5.	Location Parking List	Boundary (Interface)
6.	Parking Controller	Controller
7.	Parking Space	Entity (Database)



3.1.5.1 Use Case Scenario #5

i. Pre-Condition:

- The system is operational, and the maintenance and operations staff or administration staff have access to the system.
- There are pending parking reservation offers that require verification.

ii. Use Case Description

a. Primary Flow:

- The maintenance and operations staff or administration staff log into the system using their authorized credentials.
- The system displays a dashboard with options, including "Accept Offer".
- The staff selects the "Accept Offer" option to review pending parking reservation offers.
- The system lists pending offers with relevant details, including user information, reserved parking slot, payment status, and any associated issues or emergencies.
- The staff systematically reviews each offer to ensure accuracy and compliance with system policies.
- For each offer, the staff verifies the following:
 - a. Payment Status: Verify that the payment associated with the offer is successful.
 - b. Emergency Situations: Assess if there are any emergency situations that require special attention.
- Based on the verification, the staff can take one of the following actions:
 - a. Accept the offer if everything is in order.
 - b. Deny the offer if there are issues with payment or emergencies. Provide a reason for denial.
- The system updates the status of the parking reservation offer accordingly

b. Alternative Flow:

1) Parking Lot Full

If there is an emergency, the staff may prioritize the verification process for offers associated with emergency situations.

2) Payment Issues

If there are payment-related problems, the staff may deny the offer and provide a specific reason, such as payment failure or discrepancies.

3) Emergency Situations

If there is an emergency, the staff may prioritize the verification process for offers associated with emergency situations.

iii. Post-Condition

- The system reflects the updated status of the parking reservation offer based on the staff's verification.
- Users receive notifications regarding the status of their parking reservation, including acceptance or denial in the application and their emails.

3.1.5.2 UI Design and UI Object Description #5 Accept Offer

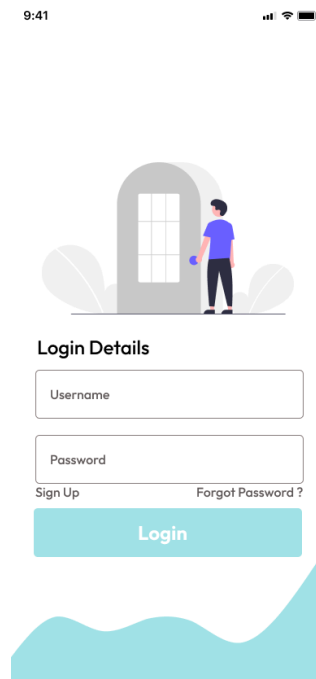
Fill in this section with the initial version of the interface prototype for each use case. Afterward, for each interface/screen, provide detailed specifications.

Table of UI Object Descriptions

SCREEN ID	SCREEN NAME	DESCRIPTION
001	Login Page	The login page which prompts the staff to enter their credentials.
101	Staff DashBoard	The staff dashboard which displays two options called pending offers and managing user accounts.
135	Pending Offers Page	Displays all pending offers. Gives staff the option to approve or deny the offer
202	No pending offers remain page	Displays when there are no pending offers remaining.

FOR EACH INTERFACE/PAGE, create detailed specifications.

Interface 001:

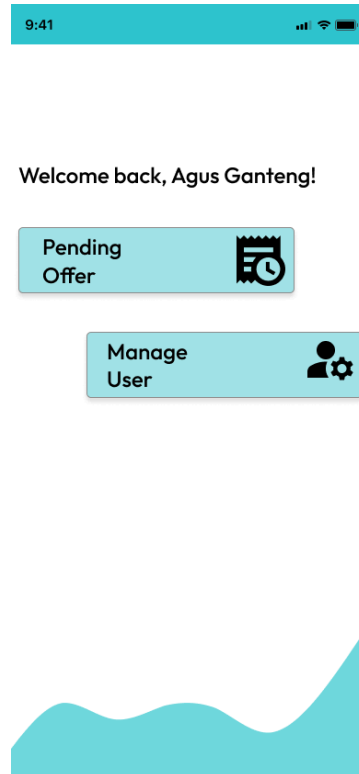


Login Page

Object_ID	TYPE	LABEL*	Description**
1234	Button	Login	If clicked, it will activate the Function login()
2345	Hyperlink	Sign up	If clicked, it will redirect to sign up page
3456	Hyperlink	Forgot Password?	If clicked, it will redirect to forgot password page
3564	TextBox	Username	Awaits text content for username

Object_ID	TYPE	LABEL*	Description**
6879	TextBox	Password	Awaits text content for passowrd
1568	Label	Login Details	Displays text “Login Details”

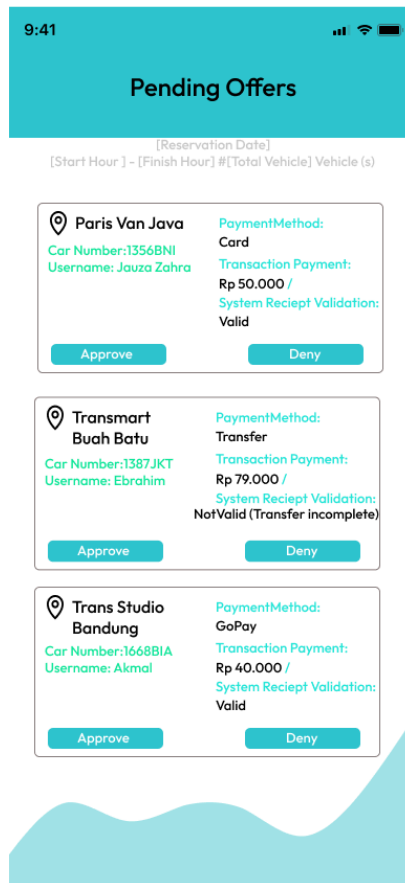
Interface 101:



Staff DashBoard

Object_ID	TYPE	LABEL*	Description**
7699	Button	Pending offer	If clicked, it will redirect to pending offers page
7688	Button	Manage User	If clicked, it will redirect to manage user page
7598	Text	Welcome Back	Displays “Welcome Back, <username>”

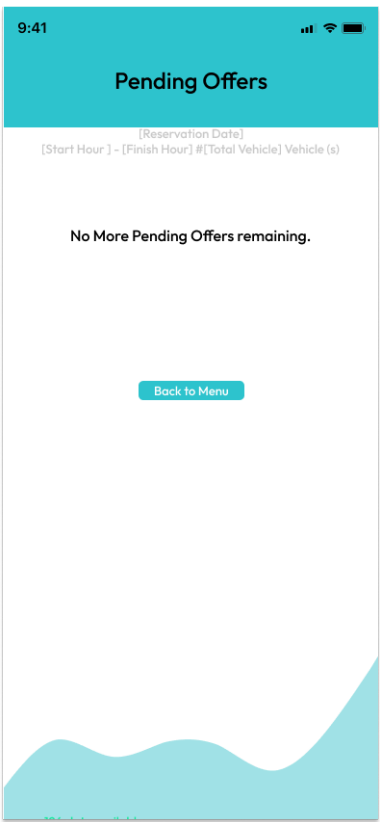
Interface 135:



Pending Offers Page

Object_ID	TYPE	LABEL*	Description**
8790	Button	Approve	If clicked, it will activate the Function ApproveOffer().
8789	Button	Deny	If clicked, it will activate the Function DenyOffer().
8987	Label	Pending Offers	Header which displays the text "Pending Offers"
8957	Label	Location	Displays the location of the parking slot
8975	Label	Payment	Displays the header and the amount

Interface 202:



No more pending offers remain Page

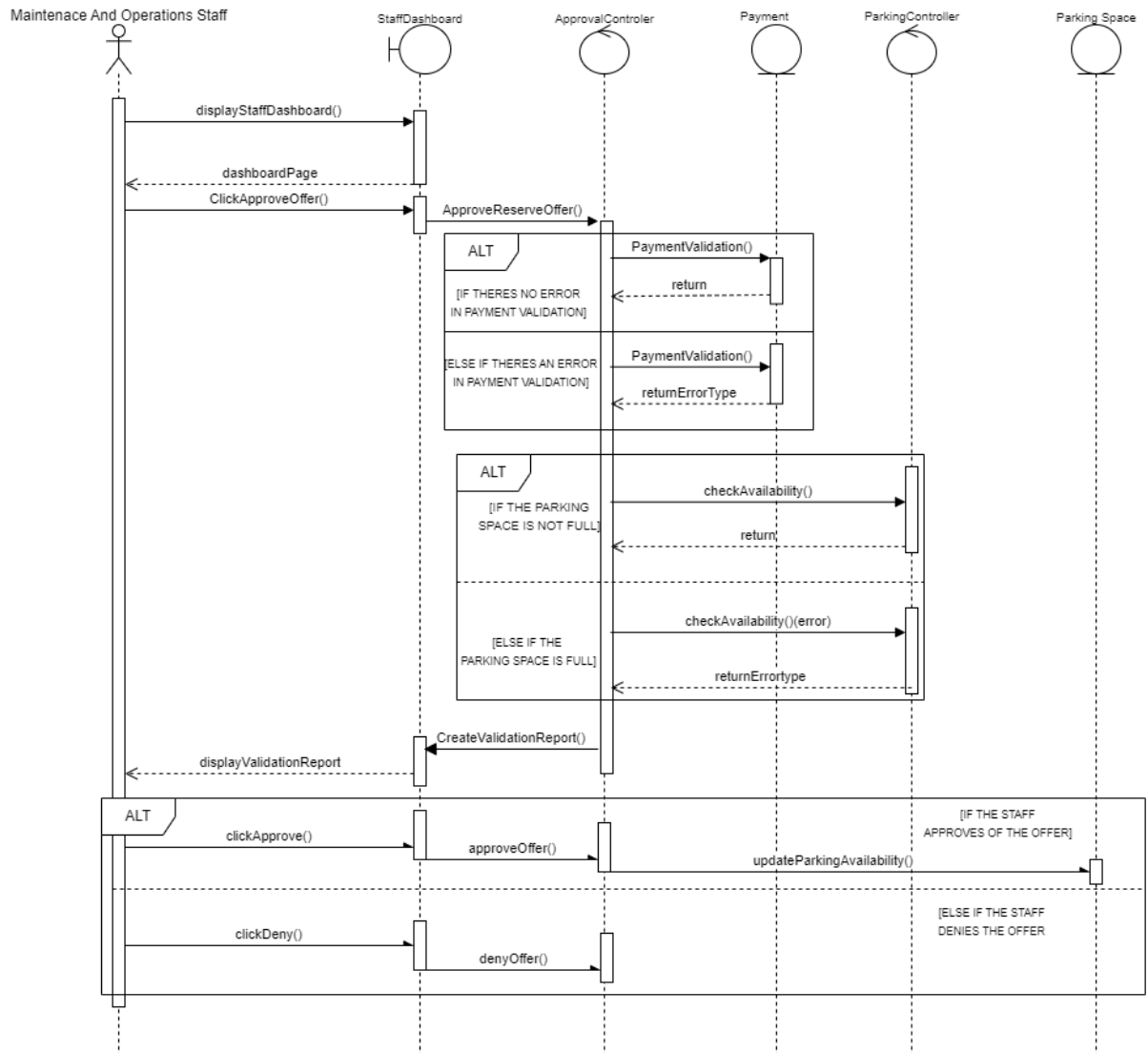
Object_ID	TYPE	LABEL*	Description**
8686	Button	Back to Menu	If clicked, it will activate the function displayStaffDashboard() and redirect to staff dashboard page
8486	Text	No more Pending Offers	Displays “No More pending offers remaining” when all the offers have been dealt with

Se

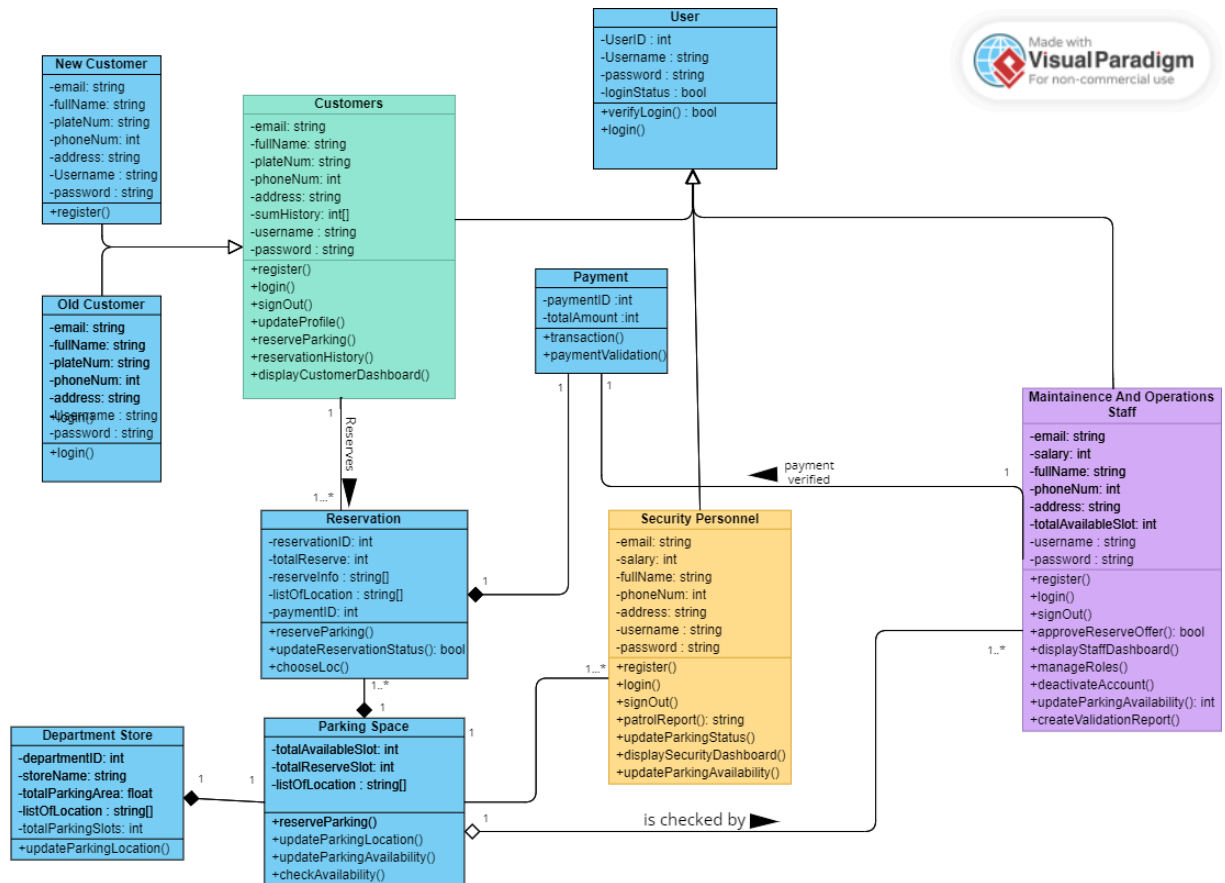
3.1.5.3 Object Identification and Class Type #5 <Accept Offer>

No	New Object Name	Class Type
1	StaffDashboard	Boundary(interface)
2	ApprovalController	Controller
3	PaymentController	Controller
4	Payment	Entity(Database)
5	Parking Space	Entity(Database)

3.1.5.4 Sequence Diagram #5 Accept Offer

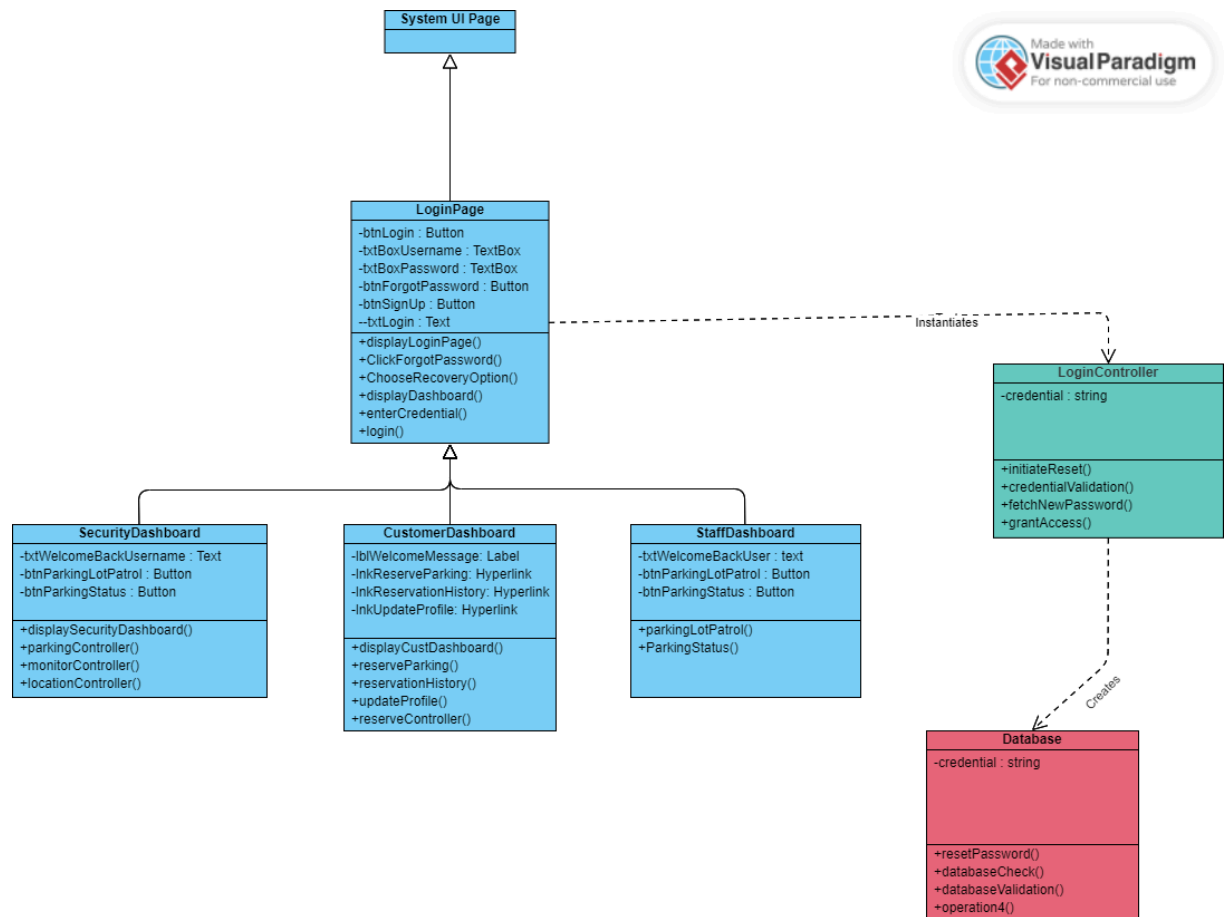


3.2 Overall Class Diagram

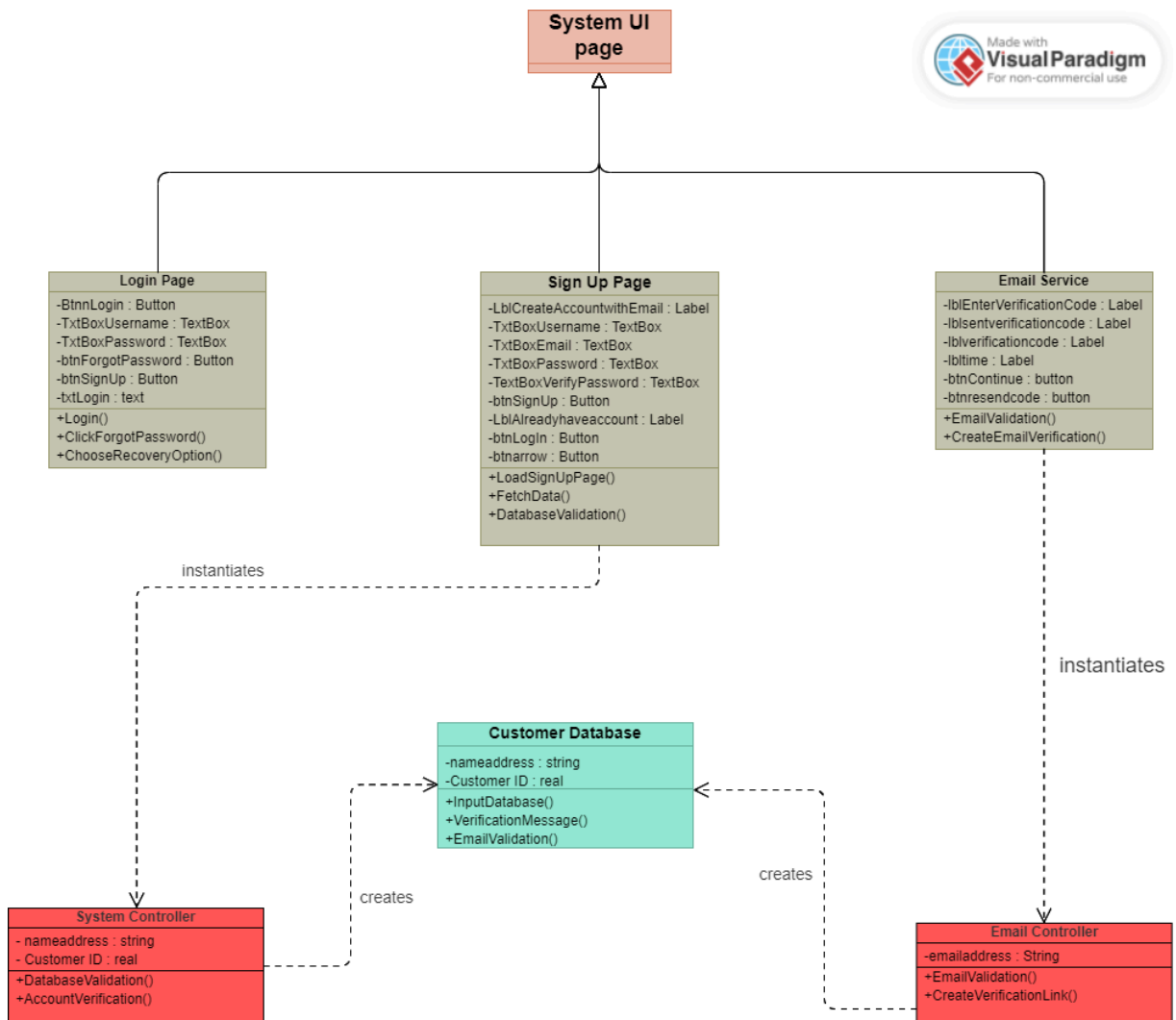


3.3 Detailed Class Design

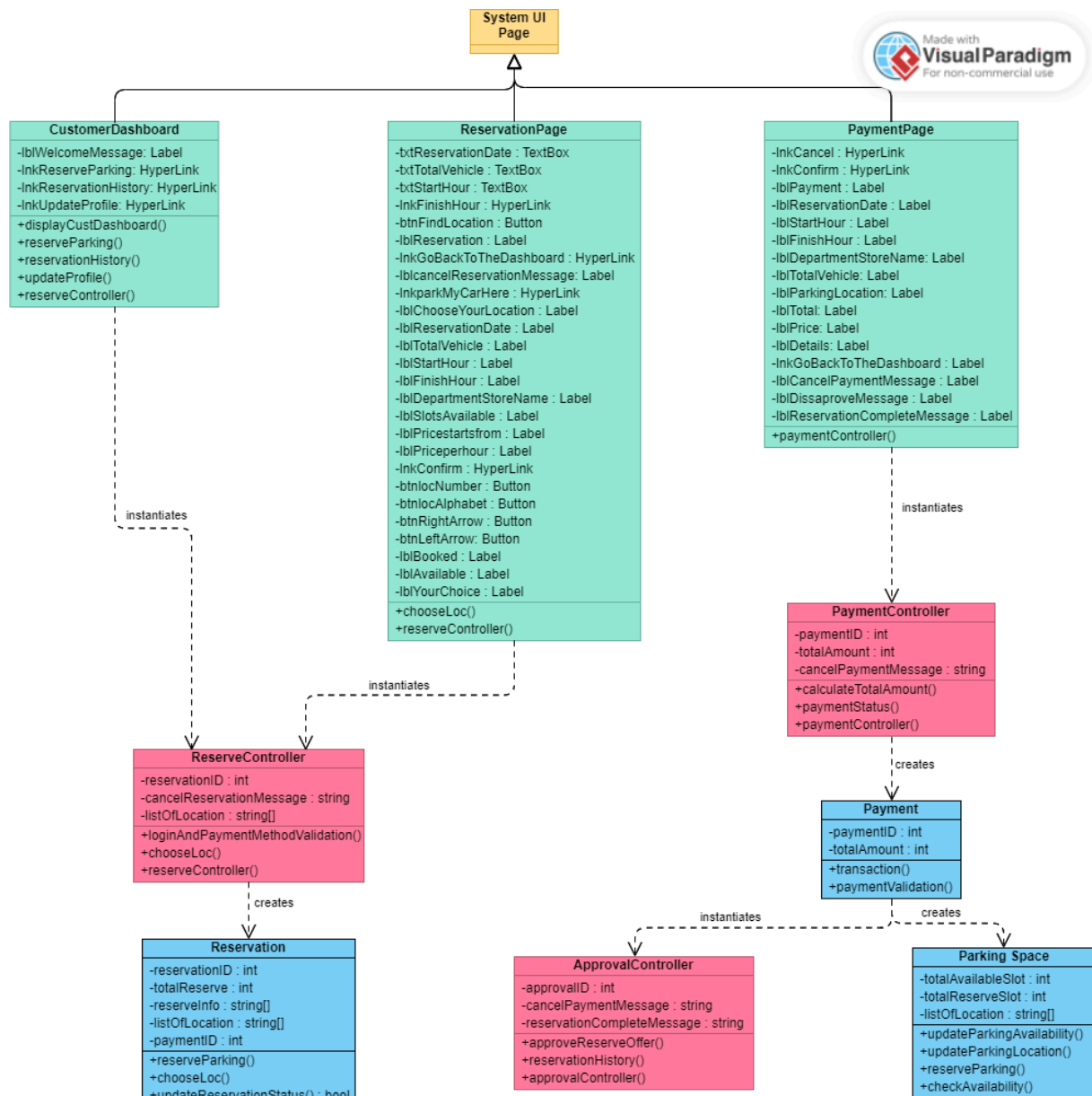
3.3.1 Class Diagram #1 Login - Muhammad Akmal Mutohar



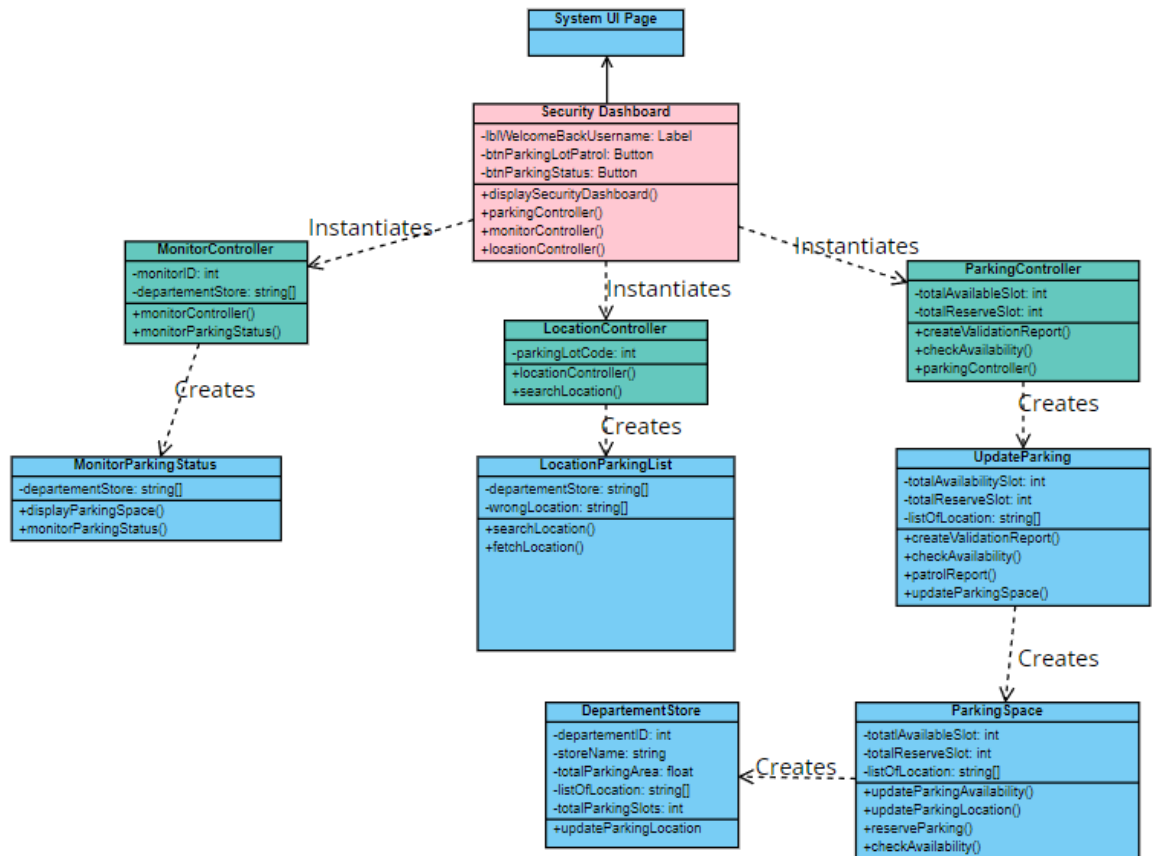
3.3.2 Class Diagram #2 Sign Up - Agam Marichal



3.3.3 Class Diagram #3 Reserve Parking Slot - Jauza Zahra



3.3.4 Class Diagram #4 Monitoring Parking Slot - Grizelda Audria Wijaya



3.3.5 Class Diagram #5

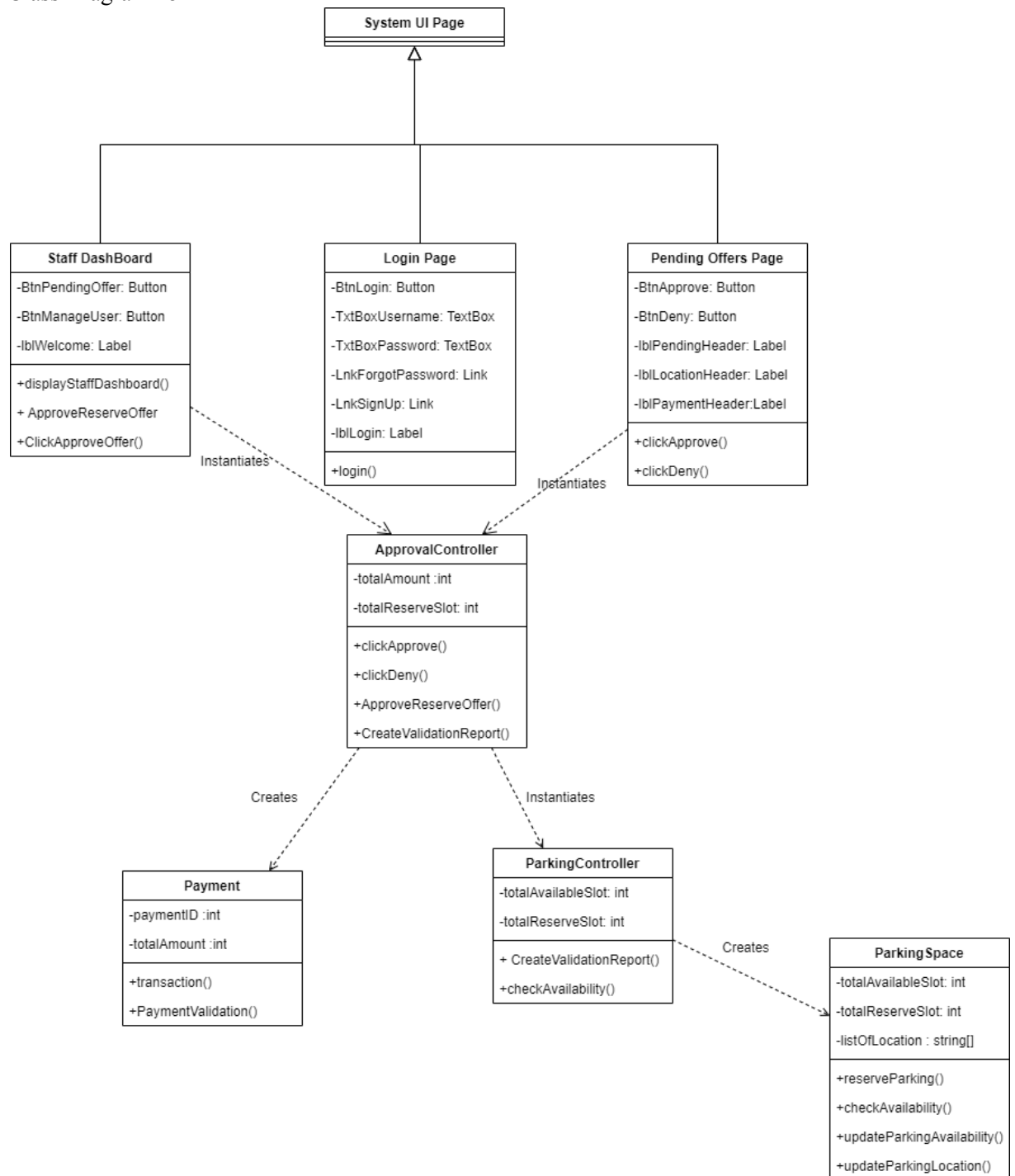


TABLE OF CLASSES :

Class ID	Name of Design Class	Atribute (visibility)	Method / Operation
006	LoginPage	-btnLogin : Button -txtBoxUsername : TextBox -txtBoxPassword : TextBox -btnForgotPassword : Button -btnSignUp : Button -txtLogin : Text	+displayLoginPage() +clickForgotPassword() +chooseRecoveryOption() +displayDashboard() +enterCredential() +login()

0077	LoginController	-credential : string	+initiateReset() +credentialValidation() +fetchNewPassword() +grantAccess()
008	Database	-credential	+resetPassword() +databaseCheck() +databaseValidation()
009	Sign Up Page	-LblCreateAccountwithEmail : Label -TxtBoxUsername : TextBox -TxtBoxEmail : TextBox -TxtBoxPassword : TextBox -TextBoxVerifyPassword : TextBox -btnSignUp : Button -LblAlreadyhaveaccount : Label -btnLogIn : Button -btnarrow : Button	+LoadSignUpPage() +FetchData() +DatabaseValidation()
010	StaffDashboard	-txtWelcomeBackUser : Text -btnParkingLot : Button -btnParkingStatus : Button	+ParkingLotPatrol() +ParkingStatus()
011	Email Service	-lblEnterVerificationCode : Label -lblsentverificationcode : Label -lblverificationcode : Label -lbltime : Label	+EmailValidation() +CreateEmailVerification()
012	Email Controller	-emailaddress : String	+EmailValidation() +CreateVerificationLink()
013	Customer Database	-nameaddress : string -Customer ID : real	+InputDatabase() +VerificationMessage() +EmailValidation()
014	System Controller	- nameaddress : string - Customer ID : real	DatabaseValidation() AccountVerification()
015	CustomerDashboard	-lblWelcomeMessage: Label -lnkReserveParking: HyperLink -lnkReservationHistory: HyperLink -lnkUpdateProfile: HyperLink	+displayCustDashboard() +reserveParking() +reservationHistory() +updateProfile() +reserveController()
016	ReservationPage	-txtReservationDate : TextBox -txtTotalVehicle : TextBox -txtStartHour : TextBox -lnkFinishHour : HyperLink -btnFindLocation : Button -lblReservation : Label -lnkGoBackToTheDashboard : HyperLink -lblcancelReservationMessage: Label -lnkparkMyCarHere : HyperLink -lblChooseYourLocation : Label -lblReservationDate : Label -lblTotalVehicle : Label -lblStartHour : Label -lblFinishHour : Label -lblDepartmentStoreName : Label -lblSlotsAvailable : Label -lblPricestartsfrom : Label -lblPriceperhour : Label -lnkConfirm : HyperLink -btnlocNumber : Button -btnlocAlphabet : Button -btnRightArrow : Button -btnLeftArrow: Button -lblBooked : Label	+chooseLoc() +reserveController()

		-lblAvailable : Label -lblYourChoice : Label	
017	PaymentPage	-lnkCancel : HyperLink -lnkConfirm : HyperLink -lblPayment : Label -lblReservationDate : Label -lblStartHour : Label -lblFinishHour : Label -lblDepartmentStoreName: Label -lblTotalVehicle: Label -lblParkingLocation: Label -lblTotal: Label -lblPrice: Label -lblDetails: Label -lnkGoBackToTheDashboard : Label -lblCancelPaymentMessage : Label -lblDisapproveMessage : Label -lblReservationCompleteMessage : Label	+paymentController()
018	ReserveController	-reservationID : int -cancelReservationMessage : string -listOfLocation : string[]	+loginAndPaymentMethod Validation() +chooseLoc() +reserveController()
004	Reservation	-reservationID : int -totalReserve : int -reserveInfo : string[] -listOfLocation : string[] -paymentID : int	+reserveParking() +chooseLoc() +updateReservationStatus() : bool
020	PaymentController	-paymentID : int -totalAmount : int -cancelPaymentMessage : string	+calculateTotalAmount() +paymentStatus() +paymentController()
021	Payment	-paymentID : int -totalAmount : int	+transaction() +paymentValidation()
022	ApprovalController	-approvalID : int -cancelPaymentMessage : string -reservationCompleteMessage : string	+approveReserveOffer() +reservationHistory() +approvalController()
023	SecurityDashboard	-lblWelcomeBackUsername: Label -btnParkingLotPatrol: Button -btnParkingStatus: Button	+displaySecurityDashboard () +parkingController() +monitorController() +locationController()
024	MonitorController	-monitorID: int -departementStore: string[]	+monitorController() +monitorParkingStatus()
025	MonitorParkingStatus	-departementStore: string[]	+displayParkingSpace() +monitorParkingStatus()
026	LocationController	-parkingLotCode: int	+locationController() +searchLocation()
027	LocationParkingList	-departementStore: string[] -wrongLocation: string[]	+searchLocation() +fetchLocation()
028	ParkingController	-totalReserveSlot: int -totalAvailableSlot: int	+checkAvailability() +createValidationReport() +parkingController()
029	UpdateParking	-totalAvailabilitySlot: int -totalReserveSlot: int -listOfLocation: string[]	+createValidationReport() +checkAvailability() +patrolReport() +updateParkingSpace()
030	ParkingSpace	-totalAvailableSlot: int -totalReserveSlot: int -listOfLocation: string[]	+updateParkingLocation() +updateParkingAvailability() +reserveParking() +checkAvailability()
031	DepartementStore	-departementID: int -storeName: string -totalParkingArea: float -listOfLocation: string[]	+updateParkingLocation()

		-totalParkingSlots: int	
032	Staff DashBoard	-BtnPendingOffer: Button -BtnManageUser: Button -lblWelcome: Label	+ ApproveReserveOffer +ClickApproveOffer() +displayStaffDashboard()
033	Login Page	-BtnLogin: Button -TxtBoxUsername: TextBox -LnkForgotPassword: Link -TxtBoxPassword: TextBox -LnkSignUp: Link -lblLogin: Label	+login()
034	Pending Offers Page	-BtnApprove: Button -lblPendingHeader: Label -BtnDeny: Button -lblLocationHeader: Label	+clickApprove() +clickDeny()
035	Approval Controller	-totalReserveSlot: int -totalAmount :int	+clickApprove() +clickDeny() +ApproveReserveOffer() +CreateValidationReport()
036	Payment	-paymentID :int -totalAmount :int	+transaction() +PaymentValidation()
037	Parking Controller	-totalAvailableSlot: int -totalReserveSlot: int	+ CreateValidationReport() +checkAvailability()
038	Parking Space	-totalAvailableSlot: int -totalReserveSlot: int -listOfLocation : string[]	+reserveParking() +checkAvailability() +updateParkingAvailability() +updateParkingLocation()

3.4 Algorithm and/or Query Design

Algorithm #1 Login - Muhammad Akmal Mutohar

Class Name : Login

Operation Name : Login()

Algorithm : Python

```
#Allows user to login if they enter the valid credentials that are stored within the database

#Get user input for login
username = input("Enter a username: ")
password = input("Enter password: ")
#Database Validation
if username = existing_username AND username = username_password:
    print("Login successful")
else
    print("Invalid Password, Please Try Again")
    return none
#If login successful then redirect to dashboard
```

Query #1 Login - Muhammad Akmal Mutohar

Simple query to match a username and password that are stored in database

SELECT * FROM users

WHERE username = 'entered_username' AND password = 'entered_password';

Description : This query is to retrieve a user's information from the users table where the entered username and password match the values stored in the database. If there is a match, it means that the entered credentials are valid, and the user can be granted access.

Algorithm #2 Sign Up - Agam Marichal

Class Name : Sign Up
Operation Name: fetchData()
Algorithm : Python

```
# Function to fetch user sign-up data and perform database validation
def fetchData():
    # Simulated database of existing usernames (replace with actual database logic)
    existing_usernames = ["user1", "user2", "user3"]

    # Get user input for Login
    username = input("Enter a username: ")
    # Perform database validation
    if username in existing_usernames:
        print("Username already exists. Please choose a different username.")
        return None
    else:
        password = input("Enter a password: ")
        email = input("Enter your email: ")
        # Additional data collection as needed

    # Simulated database entry (replace with actual database insertion logic)
    # In this example, just printing the collected data
    print("User sign-up successful. Data collected:")
    print(f"Username: {username}")
    print(f>Password: {password}")
    print(f>Email: {email}")
    # Additional data processing or database insertion

    # Return collected data or perform further actions
    return {
        "username": username,
        "password": password,
        "email": email
    }
    # Add more collected data as needed

# Sample usage
user_data = fetchData()
if user_data:
    # Proceed with further actions using user_data if needed
    print("User data collected successfully.")
else:
    print("Sign-up failed. Please try again.")
```

Query #2 Sign Up - Agam Marichal

{If referring to a specific query, complete the query below}

Query :
simple query for check the username available

Query :
SELECT * FROM users
WHERE username = 'Andi';

Description :

This SQL query is designed to check if a given username ('Username') already exists in the 'users' table. If any rows are returned by the query, it indicates that the username is already taken, and further actions, such as rejecting the new registration attempt, can be taken based on the result.

Please note that the actual implementation might vary based on your database schema and the structure of the 'users' table. Replace 'Username' with the parameter or variable that holds the username you want to validate. Additionally, ensure that your application properly handles SQL injection vulnerabilities by using parameterized queries or prepared statements, depending on the database library you are using in your code.

Algoritma #3 Reserve Parking Slot - Jauza Zahra Ulaya

Class Name : Reservation Page

Operation Name: chooseLoc()

Algorithm : Python

```
# Sample data structure representing available parking slots
available_slots = {
    'DepartmentStore1': ['A1', 'A2', 'B1', 'B2'],
    'DepartmentStore2': ['C1', 'C2', 'D1', 'D2']
}

def input_reservation_details():
    print("Reservation Details:")
    reservation_date = input("Enter Reservation Date (YYYY-MM-DD): ")
    start_hour = input("Enter Start Hour (24-hour format, e.g., 14 for 2 PM): ")
    finish_hour = input("Enter Finish Hour (24-hour format, e.g., 18 for 6 PM): ")
    total_vehicles = input("Enter Total Number of Vehicles: ")

    # Additional input validation can be added as needed

    return reservation_date, start_hour, finish_hour, total_vehicles

def check_payment_method():
    # Simulate checking the validity of the payment method
    valid_payment_method = False # Replace this with your payment method validation logic

    return valid_payment_method

def choose_department_store():
    # Display available department stores
    print("Available Department Stores:")
    for department_store in available_slots.keys():
        print(department_store)

    # Get user input for department store
    department_store_choice = input("Choose a Department Store: ")

    # Validate user input
    while department_store_choice not in available_slots:
        print("Invalid choice. Please choose a valid Department Store.")
        department_store_choice = input("Choose a Department Store: ")

    return department_store_choice

def choose_parking_slot(department_store):
    # Display available parking slots for the selected department store
    print(f"Available Parking Slots for {department_store}:")
    for slot in available_slots[department_store]:
        print(slot)
```

```

# Get user input for parking slot
parking_slot_choice = input("Choose a Parking Slot: ")

# Validate user input
while parking_slot_choice not in available_slots[department_store]:
    print("Invalid choice. Please choose a valid Parking Slot.")
    parking_slot_choice = input("Choose a Parking Slot: ")

return parking_slot_choice

def chooseLoc():
    # Check payment method
    if not check_payment_method():
        print("Please update your payment method before you want to reserve parking!")
        return

    # Input reservation details
    reservation_date, start_hour, finish_hour, total_vehicles = input_reservation_details()

    # Choose department store
    chosen_department_store = choose_department_store()

    # Choose parking slot
    chosen_parking_slot = choose_parking_slot(chosen_department_store)

    # Reserve the selected parking slot
    print(f'Reserved Parking Slot: {chosen_parking_slot} at {chosen_department_store}')
    print(f'Reservation Details: Date - {reservation_date}, Start Hour - {start_hour}, Finish Hour - {finish_hour}, Total Vehicles - {total_vehicles}')

# Example usage
chooseLoc()

```

Query #3 Reserve Parking Slot - Jauza Zahra

Query:

-- Simple query to retrieve reservations for a specific date

SELECT

reservationID,
totalReserve,
reserveInfo,
listOfLocation,
paymentID

FROM

Reservation

WHERE

reservationDate = '2024-01-15'; -- Replace '2024-01-15' with the desired date

Description:

This SQL query is designed to fetch basic information about reservations made on a specific date. The SELECT statement specifies the columns to be retrieved from the Reservation table, including reservationID, totalReserve, reserveInfo, listOfLocation, and paymentID. The WHERE clause filters the results based on the reservationDate column, selecting only those entries where the reservation date matches the specified date ('2024-01-15' in this case). The query provides a straightforward way to obtain details about reservations made on a particular day, facilitating easy tracking and analysis of reservation activities within the system.

Algorithm #4 Monitoring Parking Status - Grizelda Audria Wijaya

Class Name : Update Parking

Operation Name: updateParkingSpace()

Algorithm : Python

```
class ParkingControlSystem:
    def __init__(self, total_slots):
        self.total_slots = total_slots
        self.parking_status = {slot: 'available' for slot in range(1, total_slots + 1)}

    def updateParkingSpace(self, slot_number, status):
        """
        Update the status of a parking slot.

        Parameters:
        - slot_number (int): The parking slot number to be updated.
        - status (str): The new status of the parking slot, either "available" or "booked".

        Returns:
        - dict: Information about the updated parking slot.
        """
        if slot_number not in self.parking_status:
            return {'error': 'Invalid parking slot number'}

        if status.lower() not in ['available', 'booked']:
            return {'error': 'Invalid status. Please use "available" or "booked"'}

        self.parking_status[slot_number] = status.lower()

        return {
            'slot_number': slot_number,
            'status': status.lower(),
            'message': f'Parking slot {slot_number} is now {status.lower()}'
        }

# Example Usage:
parking_system = ParkingControlSystem(total_slots=10)

# Update parking slot 3 to 'booked'
result = parking_system.updateParkingSpace(slot_number=3, status='booked')
print(result)

# Update parking slot 5 to 'available'
result = parking_system.updateParkingSpace(slot_number=5, status='available')
print(result)
```

Query #4 Monitoring Parking Status - Grizelda Audria Wijaya

Query: Show the availability parking slot at 10:00 - 14:00

```
SELECT
    slot_number,
    status
FROM
    parking_status
WHERE
    timestamp BETWEEN "10:00:00" AND "14:00:00" AND status = 'available';
```

Description:

The MySQL query retrieves information from a table named parking_status, selecting the slot_number and status columns. The query filters the results using a WHERE clause: specifically, it selects rows where the timestamp falls between '10:00:00' and '14:00:00', representing a time range from 10:00 to 14:00. Additionally, it includes

only rows with a parking slot status of 'available'. The result provides details about parking slots that are available during the specified time frame, offering insights into parking availability for the given period.

Algorihm #5 Accept Offer - Ebrahim Nakhwa

Class Name : Parking Space
Operation Name: CheckAvailability()
Algorithm : Python

```
from enum import Enum

# Enum to represent different error types
class ErrorType(Enum):
    SPOT_TAKEN = 1
    SPOT_UNDER_CONSTRUCTION = 2
    EMERGENCY = 3

# Function to check availability
def CheckAvailability(spotNumber):
    # Simulated method to retrieve spot status from the database
    # Replace this with your actual implementation to get spot status
    # For demonstration purposes, assume some hardcoded spot statuses
    spotStatus = getStatusFromDatabase(spotNumber)

    # Check spot status and return validation report or error type
    if spotStatus == "taken":
        return createValidationReport(spotNumber, False, ErrorType.SPOT_TAKEN)
    elif spotStatus == "under_construction":
        return createValidationReport(spotNumber, False,
ErrorType.SPOT_UNDER_CONSTRUCTION)
    elif spotStatus == "emergency":
        return createValidationReport(spotNumber, False, ErrorType.EMERGENCY)
    else:
        return createValidationReport(spotNumber, True, None)

# Method to simulate retrieving status from the database
def getStatusFromDatabase(spotNumber):
    # Simulated method to retrieve spot status from the database
    # Replace this with your actual implementation to get spot status
    # For demonstration purposes, assume some hardcoded spot statuses
    if spotNumber == 1:
        return "available"
    elif spotNumber == 2:
        return "taken"
    elif spotNumber == 3:
        return "under_construction"
    elif spotNumber == 4:
        return "emergency"
    else:
        return "unknown"

# Method to create validation report or error type
def createValidationReport(spotNumber, isAvailable, errorType):
    if isAvailable:
        return f"Spot {spotNumber} is available. You can park here."
    else:
        if errorType == ErrorType.SPOT_TAKEN:
            return f"Spot {spotNumber} is already taken."
        elif errorType == ErrorType.SPOT_UNDER_CONSTRUCTION:
```

```

        return f"Spot {spotNumber} is under construction."
    elif errorType == ErrorType.EMERGENCY:
        return f"Spot {spotNumber} is unavailable due to an emergency."
    else:
        return f"Error: Spot {spotNumber} status is unknown."

# Sample usage
spot_to_check = 3 # Change this to test different spot numbers
validation_report = CheckAvailability(spot_to_check)
print(validation_report)

```

Query #5

Query :UPDATE parking_slots
 SET status = 'approved', updated_by = 'admin_username', updated_at = CURRENT_TIMESTAMP
 WHERE slot_number = 'slot_number_to_be_approved';
 Description :The purpose of this query is to update the parking space database whenever the staff approves the reservation offer

4 Requirement Traceability Matrix

Fill in with the mapping between the requirement and the realized use case.

FR Code	Name of Functional Requirement	Name of Use Case
FR-01	The system shall allow users to login into their respective accounts by providing a valid username and password.	Login
FR-02	The system shall authenticate user based on their username and password and grant access to their accounts if they are valid.	Login
FR-03	This function shall prevent unauthorized access to user accounts by denying access to users who provide invalid credentials.	Login
FR-04	This function shall allow users to log out of their accounts at any time.	Login
FR-05	This function shall allow users to create new accounts by providing the required information such as their name/username, email, and password.	Sign up
FR-06	The system shall validate the user's information and ensure that it is complete and accurate. (i.e. email validation by sending codes to email)	Sign up
FR-07	The system shall create a new user account in the database if the user's information is valid.	Sign up
FR-15	The system shall allow users to cancel their reservation.	Reserve Parking Slot
FR-16	The system shall display the slot availability in real-time.	Reserve Parking Slot
FR-17	The user shall be able to view available parking slots in the particular parking location.	Reserve Parking Slot

FR-18	The system shall display the total amount and the total number of user reservation(s).	Reserve Parking Slot
FR-19	The user shall click the “payment” button in order to complete the reservation.	Reserve Parking Slot
FR-37	The system shall be able to allow security personnel to make a report about the parking lot whether it is available or not (Monitor parking status use case)	Monitor parking status
FR-38	The system shall regularly retrieve and display real-time parking status information.	Monitor parking status
FR-39	The system shall trigger alerts and notifications when the vehicle leave first before or after the reservation time runs out.	Monitor parking status
FR-40	The system shall allow security personnel to filter the parking space view by location.	Monitor parking status
FR-28	The System shall be able to allow operation staff to accept or reject offer that have been made by the customer	Accept offer
FR-29	The System shall be able to send payment information from the customer to operations staff to secure payment process	Accept offer