BITP 3123 DISTRIBUTED APPLICATION DEVELOPMENT

SEMESTER 2 2024/2025

PROJECT TITLE: SMART PET GROOMING AND DELIVERY SYSTEM

LECTURER'S NAME: DR. HARIZ BIN MOHD NAIM @ MOHAYAT

STUDENT NAME	MATRIC NO
1. Muna Hanani binti Mohamad Fouzi	B032310628
2. Nabila Raqiqah binti Mohamad Rahman Chin	B032310538
3. Nur Insyirah Ayuni binti Abu Bakar	B032310598

1.0 INTRODUCTION

1.1 Project Overview

This project is a Smart Pet Grooming Booking System designed to simplify the grooming service process for both pet owners and staff. The system enables users (pet owners) to register, add their pets, make grooming bookings, and choose between walk-in or pickup service modes. Meanwhile, staff can view, update, and manage the bookings efficiently through a separate interface.

The system solves the common problems of manual appointment scheduling, overbooking, and miscommunication by providing a centralized, real-time platform for booking and service tracking.

1.2 Commercial Value

The Smart Pet Grooming and Delivery System offers strong potential for commercialization in today's growing pet care industry. As more pet owners seek convenience and professional grooming services, a centralized, user-friendly platform like this can fulfill key market needs.

Key Benefits and Market Relevance

Value	**Explanation**
Increasing Pet Ownership	With rising pet ownership, there is growing demand for accessible grooming services.
Convenience for Busy Pet Owners	The system allows booking appointments anytime, from anywhere, via the desktop GUI.
Pickup & Delivery Service	The option for home pickup adds convenience, appealing to elderly or working users.
Better Staff Management	Grooming businesses can monitor bookings and pickup statuses in real-time.
Centralized Booking System	Reduces human error and double-booking, improving operational efficiency.

1.2.1 Potential Business Adoption

Veterinary clinics, grooming salons, and pet care startups can use this system to automate their appointment processes and track service requests more efficiently.

- This project can be extended into a full commercial SaaS (Software-as-a-Service) platform with support for:
    - Mobile apps (for pet owners),
    - Admin dashboards (for business owners),
    - Online payment integration (FPX, credit card).

2.0 SYSTEM ARCHITECTURE

2.1 High-Level Diagram

The system architecture includes:

Frontend Application 1: A desktop interface for pet owners to register, add pet details, and make bookings.

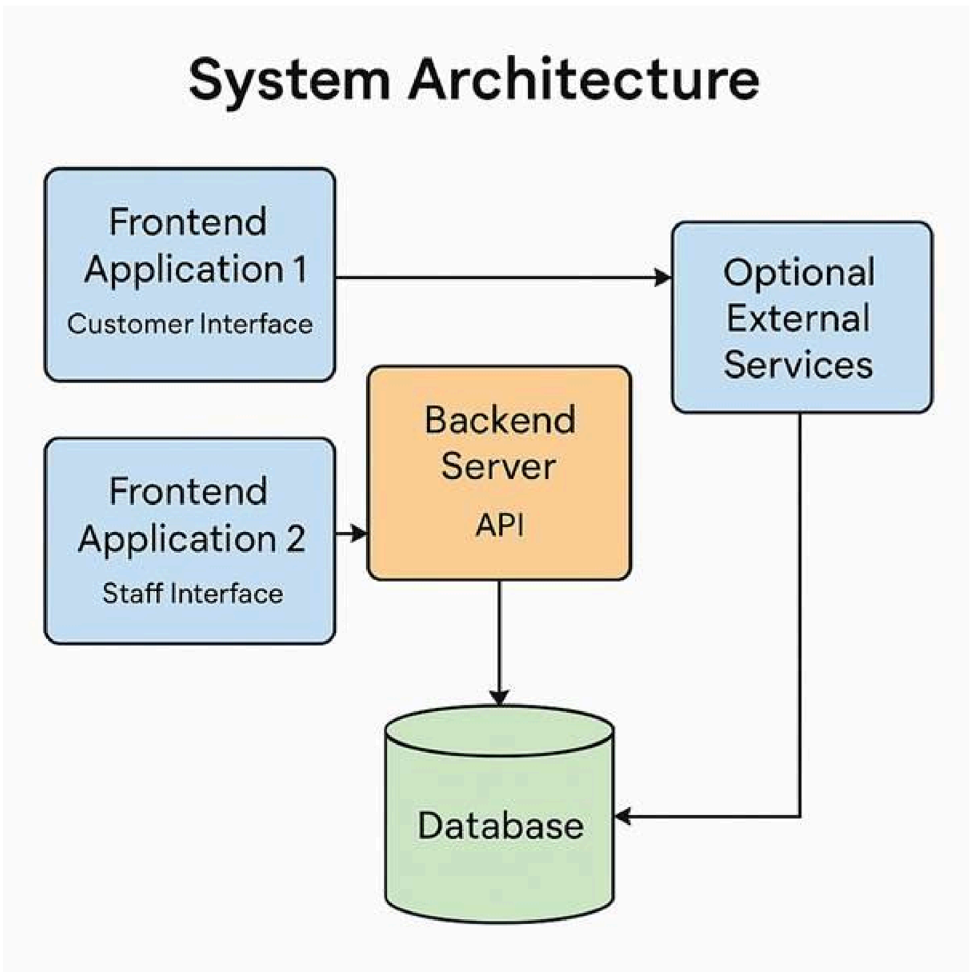
Frontend Application 2: A desktop interface for staff to view, manage, and update booking statuses.

Backend Server: Handles API requests, database interactions, and business logic.

Database: Stores all user, pet, booking, and service information.

Optional External Services: APIs for authentication, address validation, and notifications.

All components communicate over HTTP using structured API calls and JSON data exchange.



3.0 BACKEND APPLICATION

3.1 Technology Stack

Layer	Technology	Description
Programming Language	PHP 8+	Used for writing the RESTful backend logic.
Web Server	Apache (via XAMPP)	Hosts and runs PHP scripts locally
Database	MySQL/ MariaDB	Stores all application data like users, pets, bookings, services and deliveries.
API Architecture	REST	Backend exposes RESTful API endpoints for frontend access.
Development IDE	Eclipse	Used for writing and managing backend and frontend code.
Tunneling Tool	ngrok	Exposes local API (localhost) to a public URL for frontend connection and testing.

3.2 API Documentation

API Endpoints:

Base URL (for testing with ngrok):

“https://<ngrok-id>.ngrok.io/smart_petgrooming_backend/api/”

- This URL changes every time restart ngrok. Replace “<ngrok-id>”.

▪ API endpoint, method and description stated as below:

Endpoint	Method	Description
/registerOwner.php	POST	Register a new pet owner and new staff.
/login.php	POST	Log in a user using username and password and redirected to staff window or customer window based on user type.
/addPetAndBoking.php	POST	Add a new pet and create a booking
/pickupDetail.php	POST	Add new pickup address and pickup time for customer who choose “Pickup & Delivery”.
/getBookingDetail.php	POST	Get booking details for all users.
/getPickupDetail.php	POST	Retrieve pickup or delivery info by all bookings.
/updateBookingStatus.php	POST	Update booking status made by register staff.
/updatePickupStatus.php	POST	Update pickup status made by register staff.

Endpoint: /registerOwner.php

Method: POST

- Required Request Parameters

Field Name	Type	Required	Description
username	string	Yes	Desired username for the new account.
password	string	Yes	Password in plain text.
email	string	Yes	User's email address.
user_type	string	Yes	Type of user: "Pet Owner" or "Staff".
phone	string	No	User’s contact number.
address	string	No	User’s home address.

- Example Request Body (JSON)
    - To use in Postman: set method to POST . choose “ raw” , JSON.

```
{
  "username": "Muna",
  "email": "Muna@gmail.com",
  "password": "09876",
  "user_type": "Pet Owner",
  "phone": "0123456789",
  "address": "Taman Indah,KL"
}
```

- Success Response:
  - o Status Code: 200 OK

```
{"message": "Registration successful","user_id":"O009"}
```

- Error Responses

❖ Case 1- Missing require fields
❖ Status Code: 400 Bad Request

```
{ "message": "Missing required fields"}
```

❖ Case 2- Email already registered
❖ Status Code: 409 Conflict

```
{"message": "Email already registered"}
```

❖ Case 3- Invalid user type
❖ Status Code: 400 Bad Request

```
{"message": "Invalid user type"}
```

Endpoint: /login.php

Method: POST

- Required Request Parameters

Field Name	Type	Required	Description
username	string	Yes	User's login username.
password	string	Yes	User's password.

- Example Request Body (JSON)
  - o To use in Postman: set method to POST . choose " raw" , JSON.

```
{
  "username": "Muna",
  "password": "09876"
```

- Success Response:
    - Status Code: 200 OK

```
{"status":"success","message":"Login
successful","user_id":"O009","username":"Muna","user_type":"Pet Owner"}
```

- Error Responses

❖ Case 1- Missing username or password
❖ Status Code: 400 Bad Request

```
{"status": "error", "message": "Missing username or password"}
```

❖ Case 2- Username not found
❖ Status Code: 404 Not Found

```
{"status": "error", "message": "User not found"}
```

❖ Case 3- Incorrect password
❖ Status Code: 401 Unauthorized

```
{"status": "error", "message": "Invalid password"}
```

Endpoint: /addPetAndBooking.php

Method: POST

- Required Request Parameters

Field Name	Type	Required	Description
user_id	string	Yes	ID of the pet owner.
name	string	Yes	Pet’s name

type	string	Yes	Pet type (dog,cat)
breed	string	No	Breed of pet
age	integer	No	Age of pet
special_notes	string	No	Any notes about the pet.
service_type	string	Yes	Type of grooming service: "Basic Grooming" or "Full Grooming".
booking_date	string (date)	Yes	Desired booking date (format: YYYY-MM-DD).
booking_time	string (time)	No	Desired booking time
special_instructions	String	No	Instructions for groomers.
service_mode	String	Yes	Either "Walk-in" or "Pickup & Delivery".

- Example Request Body (JSON)
    - o To use in Postman: set method to POST . choose “ raw” , JSON.

```
{
  "user_id": "O009",
  "name": "Gabriela",
  "type": "Cat",
  "breed": "Persian",
  "age": 1,
  "special_notes": "Hates loud noises",
  "service_type": "Full Grooming",
  "booking_date": "2025-08-01",
  "booking_time": "10:30:00",
  "special_instructions": "-",
  "service_mode": "Pickup & Delivery"
}
```

- Success Response:
  - Status Code: 200 OK

```
{
  "user_id": "O009",
  "name": "Gabriela",
  "type": "Cat",
  "breed": "Persian",
  "age": 1,
  "special_notes": "Hates loud noises",
  "service_type": "Full Grooming",
  "booking_date": "2025-08-01",
  "booking_time": "10:30:00",
  "special_instructions": "-",
  "service_mode": "Pickup & Delivery"
```

```
{ "message": "Missing required fields"}
```

- ❖ Case 2- Invalid service mode
- ❖ Status Code: 400 Bad Request

```
{"message": "Invalid service mode"}
```

- ❖ Case 3- Owner not found
- ❖ Status Code: 404 Not Found

```
{"message": "Owner ID not found. Please register first."}
```

- ❖ Case 4- Invalid service type
- ❖ Status Code: 400 Bad Request

```
{"message": "Invalid service type"}
```

- ❖ Case 5- Database error (during insert)
- ❖ Status Code: 500 Internal Server Error

```
{"success": false, "message": "Error inserting booking: [detailed DB error]"}
```

Endpoint: /pickupDetail.php

Method: POST

- Required Request Parameters

Field Name	Type	Required	Description
Booking_id	string	Yes	Booking ID related to the pickup
Pickup_address	string	Yes	Customer’s pickup address.
Pickup_time	String(time)	Yes	Desired pickup time

- Example Request Body (JSON)
    - o To use in Postman: set method to POST . choose “ raw” , JSON.

```
{
  "booking_id": "B005",
  "pickup_address": "123 Pet Lane, KL",
  "pickup_time": "14:00:00"
}
```

- Success Response:
    - Status Code: 200 OK

```
{
  "success": true,
  "message": "Pickup details updated successfully!",
  "delivery": {
    "delivery_id": "D005",
    "booking_id": "B005",
    "service_mode": "Pickup & Delivery",
    "pickup_address": "123 Pet Lane, KL",
    "pickup_time": "14:00:00",
    "status": "Pending",
    "created_at": "2025-07-13 23:25:48"
  }
}
```

- ❖ Case 1- Missing required fields
- ❖ Status Code: 400 Bad Request

```
{"success": false, "message": "Missing required fields. Please provide booking
ID, pickup address, and pickup time."}
```

- ❖ Case 2- No delivery record found
- ❖ Status Code: 404 Not Found

```
{"success": false, "message": "Delivery record not found for this booking."}
```

- ❖ Case 3- Not a Pickup & Delivery booking
- ❖ Status Code: 400 Bad Request

```
{"success": false, "message": "This booking is not for pickup & delivery
service."}
```

- ❖ Case 4- Database error (during insert)
- ❖ Status Code: 500 Internal Server Error

```
{"success": false, "message": "message": "Error updating delivery: [error
details]"}
```

Endpoint: /getPickupDetail.php

Method: POST

- ❖ Nome required in the request body
- ❖ This endpoint retrieves all service_mode: Pickup & Delivery records from the database.
- ▪ Success Response:
    - o Status Code: 200 OK

```
{
  "success": true,
  "data": [
    {
      "delivery_id": "D003",
      "booking_id": "B003",
      "pet_name": "Kitty",
      "pickup_address": "",
      "pickup_time": "",
      "status": "Pending",
      "service_mode": "Pickup & Delivery"
    },
    {
      "delivery_id": "D038",
      "booking_id": "B038",
      "pet_name": "Gabriela",
      "pickup_address": "",
      "pickup_time": "",
      "status": "Pending",
      "service_mode": "Pickup & Delivery"
    }
  ]
```

▪ Error Responses

- ❖ Case 1- No pickup delivery records found
- ❖ Status Code: 400 Not Found

```
{"success": false, "message": "No pickup & delivery records found"}
```

- ❖ Case 2- Database error (during insert)
- ❖ Status Code: 500 Internal Server Error

```
{"success": false, "message": "message": "Error: [error message]"}
```

Endpoint: /getBookingDetail.php

Method: POST

- ❖ Nome required in the request body
- ❖ This endpoint retrieves a list of all bookings along with pet name, service, booking date, and service mode.

- Success Response:
    - Status Code: 200 OK

```
{
  "success": true,
  "bookings": [
    {
      "booking_id": "B001",
      "username": "sakura",
      "pet_name": "Fluffy",
      "service_type": "Basic Grooming",
      "booking_status": "Completed",
      "booking_date": "2023-12-15",
      "booking_time": "14:00:00",
      "service_mode": "Walk-in"
    },
    {
      "booking_id": "B038",
      "username": "Muna",
      "pet_name": "Gabriela",
      "service_type": "Full Grooming",
      "booking_status": "Pending",
      "booking_date": "2025-08-01",
      "booking_time": "10:30:00",
      "service_mode": "Pickup & Delivery"
    }
  ]
```

- Error Responses

- ❖ Case 1- No bookings found
- ❖ Status Code: 400 Not Found

```
{"success": false, "message": " "message": "No bookings found"}
```

- ❖ Case 2- Database error (during insert)
- ❖ Status Code: 500 Internal Server Error

```
{"success": false, "message": "message": "Error: [error message]"}
```

Endpoint: /updateBookingStatus.php

Method: POST

- ❖ This endpoint is used to update the status of an existing booking (e.g., from "Pending" to "Confirmed").
- Required Request Parameters

Field Name	Type	Required	Description
booking_id	string	Yes	ID of the booking to update
new_status	string	Yes	New status (e.g., "Confirmed", "Completed")

- Example Request Body (JSON)
    - o To use in Postman: set method to POST . choose " raw" , JSON.

```
{
  "booking_id": "B005",
  "new_status": "Completed"
}
```

- Success Response:
    - o Status Code: 200 OK

```
{ "success": true,   "message": "Status updated"}
```

- Error Responses

❖ Case 1- Missing parameters
❖ Status Code: 400 Bad Request

```
{"success": false, "message": "Missing parameters"}
```

❖ Case 2- SQL/database failure
❖ Status Code: 500 Internal Server Error

```
{"success": false, "message": "Update failed: [error message]"}
```

Endpoint: /updatePickupStatus.php

Method: POST

❖ This endpoint is used to update the status of an existing booking (e.g., from "Pending" to "Confirmed").
- Required Request Parameters

Field Name	Type	Required	Description
delivery_id	string	Yes	ID of the booking to update
new_status	string	Yes	New status (e.g., "Confirmed", "Completed")

- Example Request Body (JSON)
    - o To use in Postman: set method to POST . choose “ raw” , JSON.

```
{
  "delivery_id": "D005",
  "new_status": "Completed"
}
```

- Success Response:
    - Status Code: 200 OK

```
{
  "success": true,
  "message": "Status updated successfully",
  "delivery_id": "D005",
  "new_status": "Completed"
}
```

- Error Responses

- Case 1- delivery_id not found / no changes made
- Status Code: 404 Not Found

```
"success": false, "message": "No records updated - check delivery_id."}
```

- Case 2- Missing fields
- Status Code: 400 Bad Request

```
"success": false, "message": "Missing delivery_id or new_status."}
```

- Case 3- SQL/database failure
- Status Code: 500 Internal Server Error

```
{"success": false, "message": "Update failed: [error message]"}
```

Security:

- The system currently uses basic username and password authentication.
- Input validation is implemented to prevent missing or invalid fields.
- All requests are sent over HHTPS using ngrok, which encrypts data in transit.
- Paaword are stired in plain text for now.In production,we will be securely hashed using ‘password_hash()’.
- For future improvements, the system can implement JWT authentication to protect API endpoints.

5.0 FRONTEND APPLICATION

5.1 Customer Application

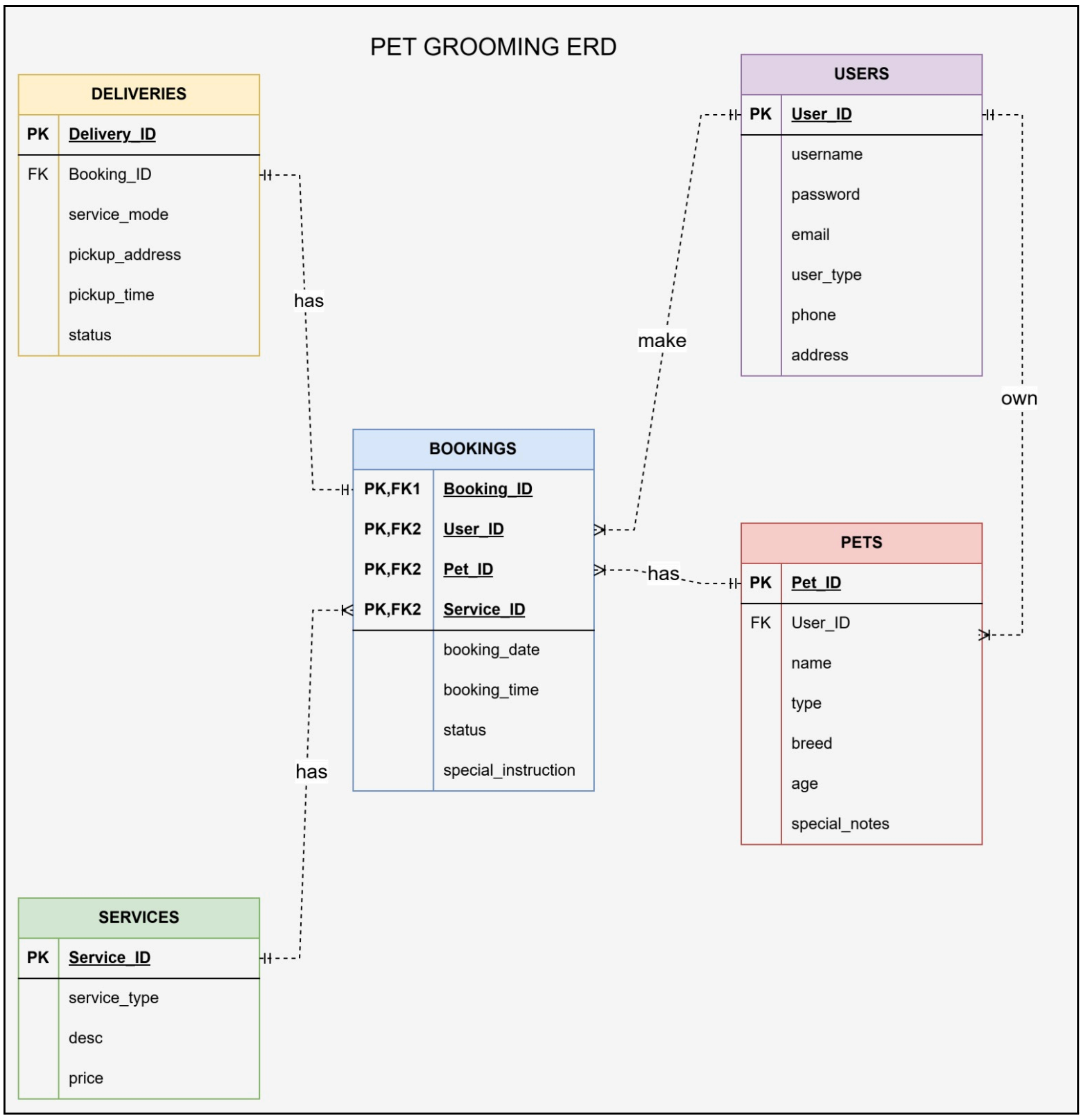
- ➔ **Purpose:** Allows pet owners to register, log in, add pet details, book grooming services, and choose pickup or walk-in.
- ➔ **Technology Stack:** Java Swing (GUI application)
- ➔ **API Integration:** Sends JSON-formatted data using HTTP POST requests to backend APIs like for registration, booking submission

5.2 Admin/Staff Application

- ➔ **Purpose:** Allows staff to log in, view all bookings, and update statuses such as booking confirmation or completion.
- ➔ **Technology Stack:** Java Swing
- ➔ **API Integration:** Loads booking data from backend API and updates it based on staff actions

6.0 DATABASE DESIGN

6.1 Entity_Relationship Diagram (ERD)



6.2 Schema Justification

6.2.1 users Table

Field Name	**Data Type**	**Constraints / Description**
user_id	VARCHAR(10)	Primary Key – Unique ID (O001, S001).
username	VARCHAR(50)	Required – For login
password	VARCHAR(255)	Required – Encrypted password
email	VARCHAR(100)	Optional – For communication
user_type	ENUM('Pet Owner','Staff')	Required – Distinguishes role
phone	VARCHAR(20)	Optional – For contact info
address	TEXT	Optional – Useful for pickup/delivery
created_at	TIMESTAMP	Default – Record creation time

6.2.2 pets Table

Field Name	**Data Type**	**Constraints / Description**
pet_id	VARCHAR(10)	Primary Key – Unique ID per pet
user_id	VARCHAR(10)	Foreign Key → users.user_id (pet owner)
name	VARCHAR(50)	Required – Pet's name
type	VARCHAR(50)	Optional – Pet type (cat, dog)
breed	VARCHAR(50)	Optional – Pet breed
age	INT(11)	Optional – Pet age
special_notes	TEXT	Optional – Extra information (allergies)

6.2.3 services Table

Field Name	Data Type	Constraints / Description
service_id	INT(11)	Primary Key – Auto Increment
service_type	ENUM('Basic Grooming','Full Grooming')	Type of grooming service
description	TEXT	Optional – Explanation of services
price	DECIMAL(10,2)	Required – Cost of the service

6.2.4 bookings Table

Field Name	Data Type	Constraints / Description
booking_id	VARCHAR(10)	Primary Key – Unique booking reference
pet_id	VARCHAR(10)	Foreign Key → pets.pet_id
service_id	INT(11)	Foreign Key → services.service_id
user_id	VARCHAR(10)	Foreign Key → users.user_id
booking_date	DATE	Required – Date selected by user
booking_time	TIME	Required – Grooming time
status	ENUM('Pending','Completed','Cancelled','In Progress')	Tracks booking progress
special_instructions	TEXT	Optional – Extra requests by pet owner
created_at	TIMESTAMP	Default – Timestamp of booking creation

6.2.5 deliveries Table

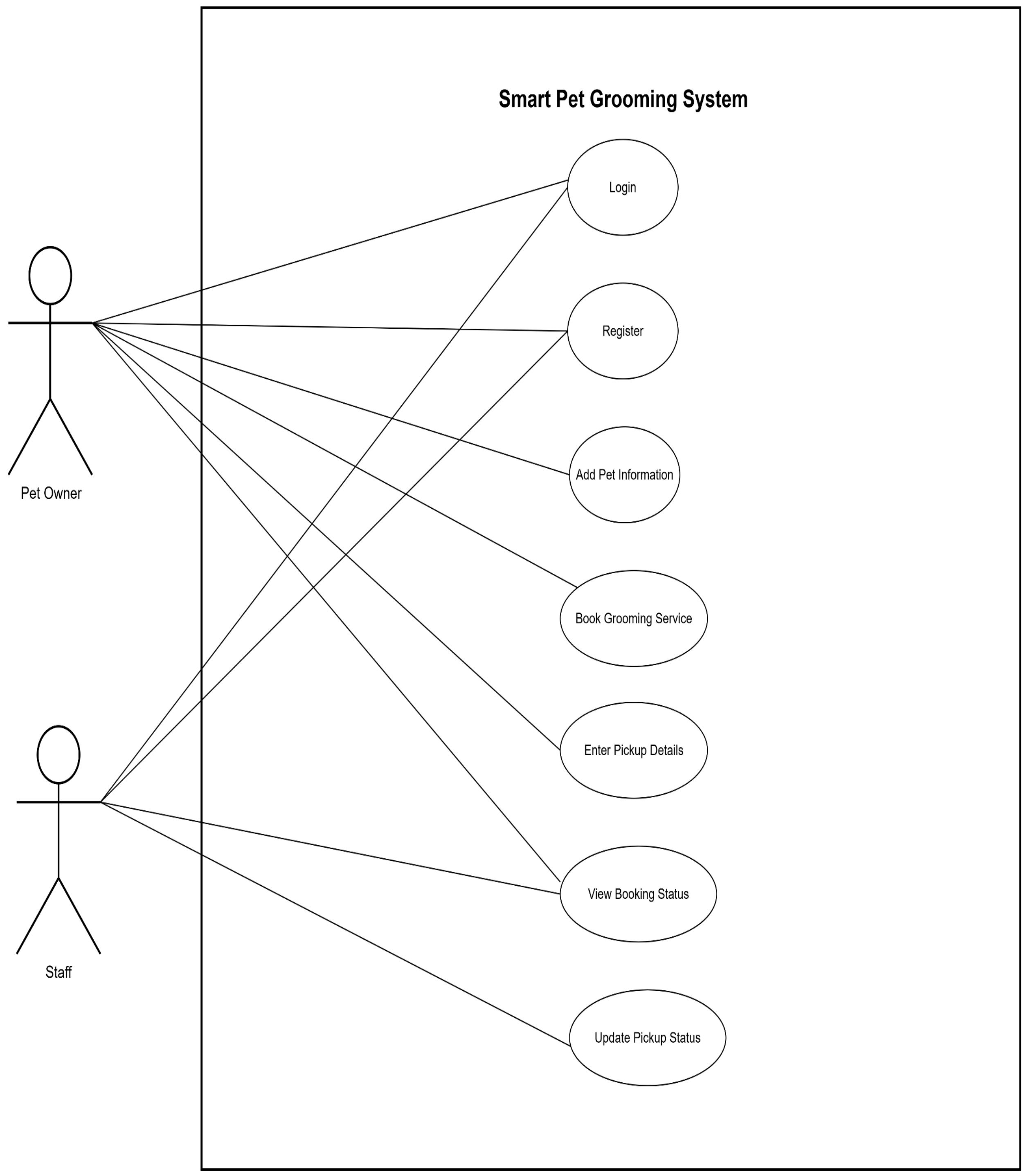
Field Name	Data Type	Constraints / Description
delivery_id	VARCHAR(10)	Primary Key – Unique booking reference
booking_id	VARCHAR(10)	Foreign Key → bookings.booking_id
service_mode	ENUM('Walk-in','Pickup & Delivery')	Type of service used
pickup_address	TEXT	Required if using pickup service
pickup_time	TIME	Required for pickup scheduling
status	ENUM('Assigned','Picked Up','Completed')	Status of delivery
created_at	TIMESTAMP	Default – Timestamp of delivery record creation

7.0 BUSINESS LOGIC AND DATA VALIDATION

7.1 Use Case Diagram

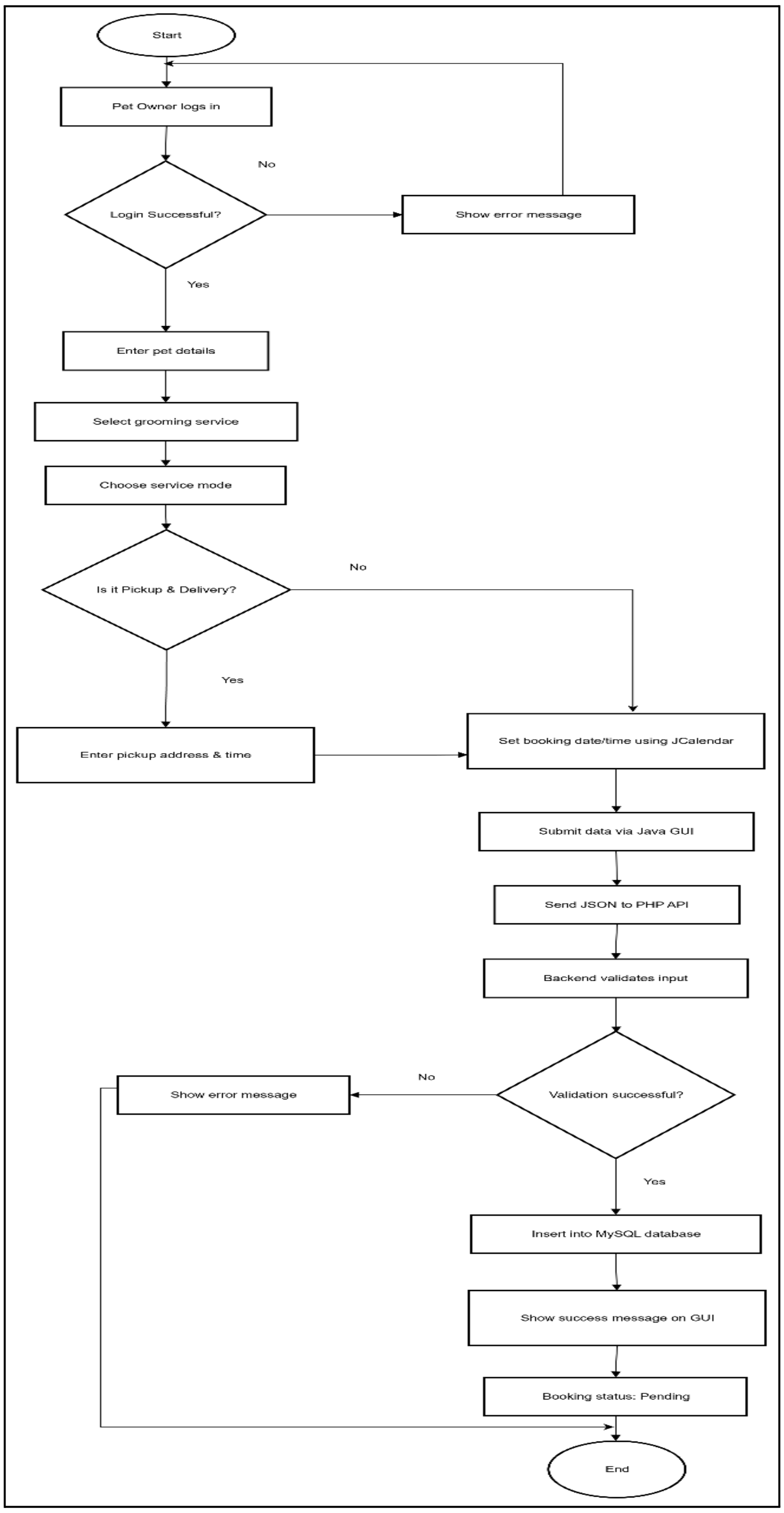
The use case diagram shows the main interactions between the users (Pet Owner and Staff) and the Pet Grooming System. Pet Owners can register, log in, add pet details, book grooming services, enter pickup information, and view booking status. Staff can register, log in, view bookings, and update the pickup/delivery status. This helps to understand what each user role is allowed to do in the system.

Use Case	**Actor**	**Description**
Login	Pet Owner, Staff	Allows users to log in using their credentials.
Register	Pet Owner, Staff	New users can create an account.
Add Pet Information	Pet Owner	Enter pet name, type, breed, and notes.
Book Grooming Service	Pet Owner	Choose service, mode (Walk-In or Pickup&Delivery), and date/time.
Enter Pickup Details	Pet Owner	Only shown if Pickup & Delivery is selected.
View Booking Status	Pet Owner, Staff	Check current status (Pending, Completed).
Update Pickup Status	Staff	Staff updates status (Cancelled, Delivered)


Smart Pet Grooming System
Pet Owner
Staff
Login
Register
Add Pet Information
Book Grooming Service
Enter Pickup Details
View Booking Status
Update Pickup Status

7.2 Flowchart

The flowchart displays the step-by-step process of how a Pet Owner uses the system. It starts from logging in, entering pet and booking details, selecting date and time (via calendar), and finally submitting.If Pickup & Delivery is selected, the owner also fills in pickup details.The flow ensures that all necessary data is collected before the booking is completed.



7.3 Data Validation

In our Smart Pet Grooming System, data validation is implemented on both the frontend (Java GUI) and backend (PHP) to ensure that user inputs are accurate, complete, and secure before being processed or stored in the database.

7.3.1Frontend Validation (Java GUI)

The frontend uses basic validation checks before sending any data to the backend via HTTP requests. This helps catch user errors early and improves user experience.

Validation	GUI	Description
Empty Field Check	Registration & Login	Ensures that username, password, email, etc. are not left blank.
Password Match	Registration Window	Confirms that password and confirm password fields match.
Service Mode Selection	Booking Window	Users must select either "Walk-In" or "Pickup & Delivery" to proceed.
Date Selection	JCalendar (Booking GUI)	Ensures a date is selected for booking.
Time Format	Pickup Details Window	Pickup time must follow the HH:MM format.
Address Presence	Pickup Details Window	Pickup address must not be empty when service mode is Pickup & Delivery.

7.3.2 Backend Validation (PHP & MySQL)

The backend performs essential validation checks even after frontend validation, as a security measure and to maintain data integrity.

Validation	**PHP File**	**Description**
Email Uniqueness	register.php	Before inserting a new user, check if the email already exists in the database.
Required Fields	All endpoint files	Ensures all required JSON keys (username, booking details) are present.
Role Checking	login.php	Confirms the user role to redirect to correct interface (owner/staff).
Valid Foreign Keys	addPetAndBooking.php	Verifies that user_id and service_id exist in respective tables before booking.
Data Format Checks	PHP input validation	Basic sanitization for address, time, and special notes to prevent SQL injection.