Student Guide to Manipal

Mini Project Report -Database Lab (DSE 2241)

Department of Data Science & Computer Applications

B. Tech Data Science

4th Semester – Batch: A1 - Group: A13

Submitted By

Riddhika Rungta	230968068
Sthuthi V. Soans	230968047
Sohan Sanil	230968074
Rohit Vinod	230968045

Mentored By

Mr. Shrinidhi B

Assistant Professor-Senior

DSCA, MIT

Mrs. Archana H

Assistant Professor-Senior

DSCA, MIT

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

Date: 13 April 2025

CERTIFICATE

This is to certify that the Riddhika Rungta (230968068), Sthuthi V. Soans (230968047), Sohan Sanil (230968074) and Rohit Vinod (230968045), have successfully executed a mini project titled "Student Guide to Manipal" rightly bringing fore the competencies and skill sets they have gained during the course- Database Lab (DSE 2241), thereby resulting in the culmination of this project.

Mr. Shrinidhi B
Assistant Professor-Senior
DSCA, MIT

Mrs. Archana H
Assistant Professor-Senior
DSCA, MIT

ABSTRACT

Students moving to new places for higher education often struggle to navigate their surroundings and access essential services. In today's fast-paced world, quick and efficient access to such services is crucial. To address this challenge, the Student Guide to Manipal database is designed to provide students with comprehensive information about Manipal. The main objective of the project is to help students familiarize themselves with their new environment, enabling them to find all kinds of services and their locations in no time.

The database consists of multiple tables pertaining to various services provided in and around Manipal like health care, restaurants, postal services, police stations, laundry facilities, car rentals, tailoring, grocery stores, picnic spots, fashion outlets, theatres etc. Each table includes key details such as location/address, phone number, ratings, online delivery options, etc. The tool used to create the project is SQL Plus.

These are some of the basic necessities of a student in new surroundings. Thus, by implementing this database, we can create a student friendly environment which ensures that all their needs are one tap away, thus making their transition to Manipal smoother and more comfortable. Using SQL Plus, the database so created results in efficient data storage and easy retrieval of information. Security is a high priority in the present world, which is kept in mind while making the database, so that students can safely utilize the data without concerns.

In conclusion, the project is aimed at providing seamless access to important amenities for the students who are starting a new journey of education in a new place. The efficient database allows us to carry out business analysis so as to improve our project and expand services according to the new trends.

Contents

Chapter 1

Introduction

Relocating to a new city for higher education presents several challenges for students, especially when it comes to accessing essential daily services. For newcomers at Manipal Institute of Technology (MIT), navigating the local ecosystem—which includes everything from finding reliable food outlets and medical facilities to locating laundry services or grocery stores—can often be confusing and time-consuming. In a world where information access and service efficiency are vital, there is an increasing need for centralized platforms that guide students to the right resources in a timely and reliable manner. This project stems from that very need: to build a unified digital system that acts as a comprehensive service guide for students beginning their journey in Manipal.

Currently, students depend on fragmented sources such as social media recommendations, word-of-mouth suggestions, or scattered web searches to find everyday services. This lack of structure often leads to inefficiency, misinformation, and missed opportunities to engage with trusted local providers. Moreover, there is no existing institutional portal or system in place that helps students organize their service needs or offer reviews that could benefit others. These gaps in accessibility and transparency clearly highlight the demand for a system that is both reliable and student-centric.

The proposed solution is a structured backend database system titled *Student Guide to Manipal*. This database stores details of critical services such as hostels, restaurants, gyms, medical stores, transport providers, salons, and more. Through a well-designed schema comprising entities like Students, Services, Service Providers, Bookings, and Reviews, the system facilitates core functionalities like registration, service discovery, booking, and feedback. In addition to helping students, the system also empowers local service providers to register and maintain their information, thus enhancing mutual accessibility.

The advantages of implementing such a system are multifold. It simplifies students' day-to-day lives, reduces dependence on unverified information sources, and improves efficiency in accessing local resources. The project also ensures data consistency, scalability, and ease of retrieval. Once integrated with a frontend interface in the future, this database system could serve as a one-stop solution for MIT students to fulfill their everyday needs—securely, quickly, and with confidence. Furthermore, the use of Oracle SQL and PL/SQL allows for precise control over data integrity, performance, and transactional operations, ensuring a solid technical foundation for potential expansion.

Chapter 2

Synopsis

2.1 Proposed System

The Student Guide to Manipal database system is designed to provide an organized and centralized digital backend that enables students at Manipal Institute of Technology to easily access information about essential services available in and around the campus. This includes services such as hostels, restaurants, gyms, medical stores, grocery outlets, salons, transport, and more.

The problem addressed by this project is the lack of a unified system for new and existing students to explore, compare, and avail services quickly in a new environment. Currently, students rely on scattered platforms such as social media groups, personal references, or offline exploration, which are often inefficient and unreliable. The proposed system allows both students and service providers to register into the platform, maintain their profiles, and interact through features such as bookings, reviews, and service discovery.

The project builds a fully functional backend using Oracle SQL and PL/SQL. The database schema includes multiple interrelated entities like Students, Services, Service_Details, Avails, Writes_Reviews, Hostels, and Emergency_Contacts. It also includes procedures for registration, login validation, booking, feedback submission, service filtering, and automatic ID generation. This backend system lays the foundation for a future full-stack application that can serve as an official student portal for service accessibility and campus support.

2.2 Objectives

The main objectives of the project are:

• To create a comprehensive and structured backend database system for student services in Manipal.

• To allow new students to explore nearby services in categories such as food, health, transport, fitness, and essentials.

• To enable seamless student registration, login validation, and emergency contact mapping.

• To allow service providers to register their services and manage offerings including price, availability, and contact details.

• To allow students to avail services, view distance-based recommendations, and post verified reviews.

• To auto-generate student IDs, login credentials, service detail IDs, and emergency contact IDs dynamically using PL/SQL logic.

• To reduce manual effort in accessing student-relevant services and improve accessibility through structured information.

• To ensure secure and normalized data storage using Oracle SQL and PL/SQL, ensuring scalability and data integrity for future integration.

Chapter 3: Functional Requirements

This backend database project serves as a student-centric portal designed to simplify access to essential services in and around Manipal Institute of Technology. It supports core functionalities like student/service provider registration, login validation, service booking, review submission, and booking status updates. The system is implemented entirely in Oracle SQL and PL/SQL and simulates end-to-end flow of interaction between users and services using modular database logic.

3.1 User Registering/Login Module

This module allows students and service providers to register securely, log in using system-generated credentials, and reset passwords in case they forget them. It supports functionalities:

- New user registration
- Login
- Forgot password

3.1.1 New User Registration

A new student or service provider must provide their details. Unique IDs (Login_ID, Student_ID, Detail_ID, and Contact_ID) are auto-generated based on record count. Passwords must be secure and validated.

Table 3.1 Registration

INPUT	Role(Student/Service), Name, Email, Phone, Password, Additional profile fields
Processing	• Validate password strength • Auto-generate Login ID • Generate Student ID / Detail ID /

	Contact ID ● Insert into Login, Students, Service_Details, Emergency_Contacts as required
OUTPUT	Success message with generated IDs / Prompt to re-enter incorrect values

3.1.2 Login

Users (students or service providers) can log in with their generated credentials.

Table 3.2 Login

INPUT	Login_ID, Password
Processing	● Match Login_ID and Password in Login table ● Trigger trg_validate_login can also be used to simulate login attempt
OUTPUT	Login successful with role displayed / Error if credentials are incorrect

3.1.3 Forgot Password

If users forget their passwords, they can reset it by verifying their registered phone number.

Table 3.3 Forgot Password

INPUT	Login_ID, Phone Number, New Password

Processing	• Validate user-phone match • Validate new password length • Update password in Login table
OUTPUT	Password changed successfully / Error message on mismatch

3.2 Service Booking Module

This module enables students to view and book services based on service type, proximity, and ratings.

Table 3.4 Service Booking

INPUT	Student_ID, Selected Detail_ID
Processing	• Use proc_show_service_options to list nearby services • Avail_ID is auto-generated
OUTPUT	Booking recorded in Avails table with “Pending” status

3.3 Booking Status Update Module

Allows service providers or system to update the status of a booking from "Pending" to either "Confirmed" or "Cancelled".

Table 3.5 Update Booking Status

INPUT	Avail_ID, New Status
Processing	• Validate that current status is "Pending"

	• Update the record using proc_update_avail_status
OUTPUT	Success message confirming update

3.4 Review and Feedback Module

Enables students to post reviews after availing a service. The system can also compute and update average ratings automatically.

Table 3.6 Review Submission

INPUT	Student_ID, Detail_ID, Rating, Comment
Processing	• Insert record into Writes_Reviews • Optionally call proc_update_rating using func_get_avg_rating to update service average rating
OUTPUT	Review saved / Service rating updated

3.5 Reporting and Analytics Module

This module includes useful queries for gaining insights about student engagement and service performance. These queries assist in identifying popular services, student activity, and cost trends, thereby helping improve service offerings.

Table 3.7 Reporting Queries

Functionality	**Description**
Top 5 highest rated services	Query to retrieve top-rated service providers using AVG()
Login IDs that have never used the system	Identifies users who have never booked or reviewed
Services with rating greater than 8	Shows high-quality services based on user reviews
Average cost of services by service type	Provides cost trend analysis per category
Number of services availed by each student	Shows student engagement and usage frequency
Reviews posted for a particular service provider	Lists feedback and ratings for a selected Detail_ID

Chapter 4
Detailed Design

4.1 ER Diagram

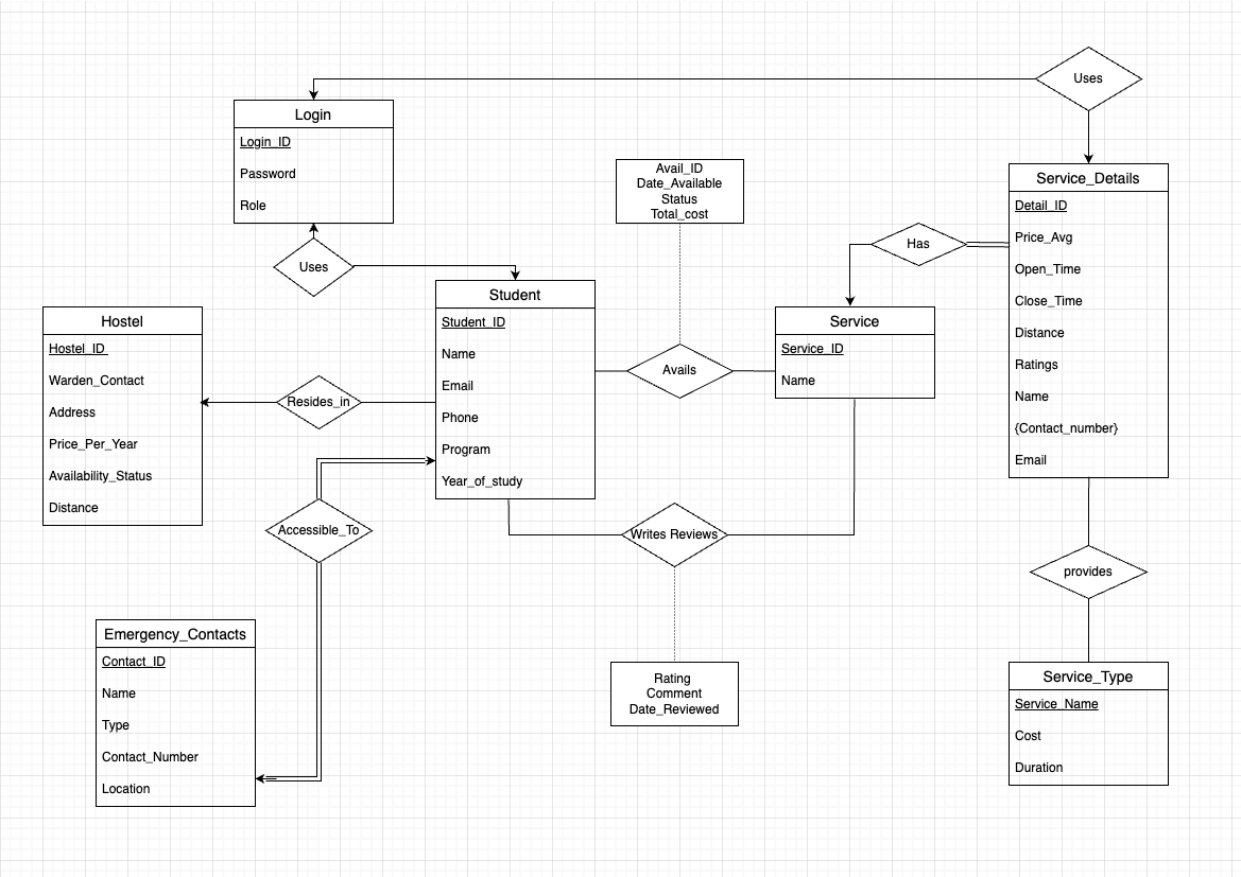

Figure 4.1 ER Diagram

4.2 Schema Diagram

Student(Student_ID, Name, Email, Phone, Program, Year_of_Study, Login_ID, Hostel_No, Contact_ID)

Login(Login_ID, Password, Role)

Hostel(Hostel_No, Warden_Contact, Address, Price_per_Year, Availabiliy_Status, Distance)

Emergency_Contacts(Contact_ID, Name, Type, Contact_Number, Location)

Service(Service_ID, Name)

Service_Details(Detail_ID, Price_Avg, Open_Time, Close_Time, Distance, Ratings, Name, Contact_Number, Email, Login_ID, Service_ID)

Service_Types(Service_Name, Cost, Duration)

Provides(Service_Name, Detail_ID)

Avails(Avail_ID, Student_ID, Service_ID, Date_Available, Status, Total_Cost)

Writes_Reviews(Student_ID, Service_ID, Rating, Comments, Date_Reviewed)

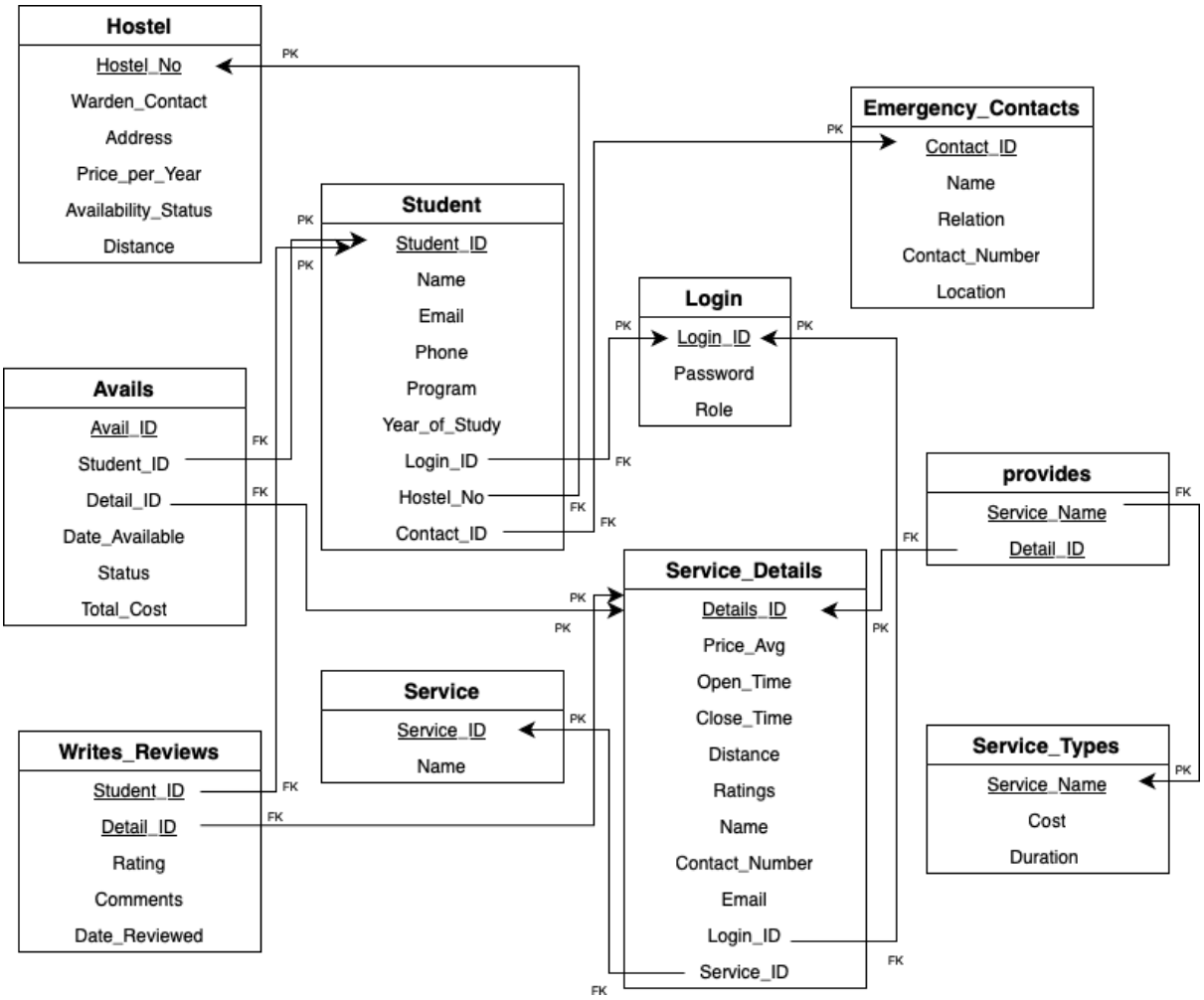

Figure 4.2 Schema Diagram

4.3 Data Dictionary

LOGIN

Column	Data Type	Constraints	Constraint_name
Login_ID	varchar2(9)	Primary Key	
Password	varchar2(15)	unique	unq_pass
Role	varchar2(10)	valid-Student,Admin,Service	const_role

EMERGENCY_CONTACTS

Column	Data Type	Constraints	Constraint_name
Contact_ID	varchar2(5)	Primary Key, starts with 'C'	
Name	varchar2(20)	Unique	
Relation	varchar2(20)		
Contact_Number	number(10)	valid- '_____'	const_contact
Location	varchar2(50)		

SERVICE

Column	Data Type	Constraint	Constraint_name
Service_ID	varchar2(6)	Primary Key, valid-'S_____'	const_service_ID
Name	varchar2(30)	Unique	

SERVICE_DETAILS

Column	Data Type	Constraints	Constraint_Name
Detail_ID	varchar2(6)	Primary Key, valid- 'D____'	const_detail_ID
Price_avg	number(5)	>0	const_price_avg
Open_Time	Date		
Close_Time	Date		
Distance	number(5,2)		
Ratings	number(2)	>0 and <=10	Rating_lim
Name	varchar2(30)		
Contact_num	number(10)	unique,'_____'	unique_no
Email	varchar2(30)	contains'%@%.com'	email_const

Login_ID		references Login	
Service_ID		references Service	

PROVIDES

Column	Data Type	Constraints	Constraint_name
Service_name		References Service_Types	
Detail_ID		References Service_Details	

SERVICE_TYPES

Column	Data Type	Constraints	Constraint_name
Service_Name	varchar2(20)	Primary Key	
Cost	number(5)		
Duration	number(5)		

HOSTEL

Column	Data Type	Constraint	Constraint_name
Hostel_No	varchar2(6)	Primary Key, Valid- 'B__'	const_hostel_no
Warden_contact	number(10)	Unique, '_____'	
Address	varchar2(100)		
Price_per_Year	number(10,2)	>0	
Availability_Status	varchar2(3)	Valid-Yes,No	
Distance	number(5,2)	>0	

STUDENT

Column	Data Type	Constraints	Constraint_name
Student_ID	varchar2(6)	Primary Key, Valid- 'M_____'	const_stud_ID
Name	varchar2(40)		
Email	varchar2(40)	contains '@manipal.edu'	const_stud_email
Phone	number(10)	unique, '_____'	
Program	varchar2(6)	valid-BTECH, MTECH, MBA, BBA, BCA, MCA, MMBS	const_prog
Year_of_Study	number(4)		
Login_ID		references Login	
Hostel_No		references Hostel	
Contact_ID		References Emergency Contacts	

AVAILS

Column	Data Type	Constraint	Constraint_name
Avail_ID	varchar2(6)	Primary Key, Starts with 'A'	
Student_ID		references Student	
Detail_ID		references Service Details	
Data_Avails	Date		

Status	varchar2(20)	valid-Confirmed, Pending, Cancelled	const_status
Total cost	number(7)	>0	

WRITES_REVIEWS

Column	Data Type	Constraint	Constraint_name
Student_ID		Primary Key, references Student	
Detail_ID		Primary Key, references Service Details	
Rating	number(2)	>0 and <=10	
Comments	varchar2(50)		
Date_Reviewed	Date		

4.4 Relational Model Implementation

```
CREATE TABLE LOGIN (LOGIN_ID VARCHAR2(9) PRIMARY KEY, PASSWORD VARCHAR2(15) CONSTRAINT UNQ_PASS UNIQUE, ROLE VARCHAR2(10) CONSTRAINT CONST_ROLE CHECK(ROLE IN ('STUDENT','ADMIN','SERVICE')));

CREATE TABLE EMERGENCY_CONTACTS(CONTACT_ID VARCHAR2(5) PRIMARY KEY CHECK(CONTACT_ID LIKE 'C%'), NAME VARCHAR2(20) UNIQUE, RELATION VARCHAR2(20) , CONTACT_NUMBER NUMBER(10) CHECK(CONTACT_NUMBER LIKE '__________'), LOCATION VARCHAR2(50));

CREATE TABLE SERVICE(SERVICE_ID VARCHAR2(6) CONSTRAINT ONST_SERVICE_ID CHECK(SERVICE_ID LIKE 'S_____') PRIMARY KEY, NAME VARCHAR2(30) UNIQUE);

CREATE TABLE SERVICE_TYPES( SERVICE_NAME VARCHAR2(20) PRIMARY KEY, COST NUMBER(5), DURATION NUMBER(5));

CREATE TABLE SERVICE_DETAILS(DETAIL_ID VARCHAR2(6) CONSTRAINT CONST_DETAIL_ID CHECK(DETAIL_ID LIKE 'D_____') PRIMARY KEY, PRICE_AVG NUMBER(5) CONSTRAINT CONST_PRICEAVG CHECK(PRICE_AVG >0), OPEN_TIME DATE,
```

```
CLOSE_TIME DATE, DISTANCE NUMBER(5,2), RATINGS NUMBER(2) CONSTRAINT
RATING_LIM CHECK(RATINGS BETWEEN 0 AND 10), NAME VARCHAR2(30),
CONTACT_NUM NUMBER(10) CONSTRAINT UNIQUE_NUM UNIQUE CHECK(CONTACT_NUM
LIKE '__________'), EMAIL VARCHAR2(30) CONSTRAINT EMAIL_CONST
CHECK(EMAIL LIKE '%@%.COM'), LOGIN_ID VARCHAR2(9) REFERENCES LOGIN,
SERVICE_ID VARCHAR2(6) REFERENCES SERVICE);

CREATE TABLE HOSTEL( HOSTEL_ID VARCHAR2(6) PRIMARY KEY CONSTRAINT
CONST_HOSTEL_NO CHECK(HOSTEL_ID LIKE 'B_____'), WARDEN_CONTACT
NUMBER(10) UNIQUE CHECK(WARDEN_CONTACT LIKE '__________'), ADDRESS
VARCHAR2(100), PRICE_PER_YEAR NUMBER(10,2) CHECK(PRICE_PER_YEAR>0),
AVAILABILITY_STATUS VARCHAR2(3) CHECK(AVAILABILITY_STATUS IN
('YES','NO')), DISTANCE NUMBER(5,2) CHECK(DISTANCE>0));

CREATE TABLE PROVIDES( SERVICE_NAME VARCHAR2(20) REFERENCES
SERVICE_TYPES, DETAIL_ID VARCHAR2(6) REFERENCES SERVICE_DETAILS);

CREATE TABLE STUDENTS( STUDENT_ID VARCHAR2(6) PRIMARY KEY CONSTRAINT
CONST_STUD_ID CHECK(STUDENT_ID LIKE 'M_____'), NAME VARCHAR2(40), EMAIL
VARCHAR2(40) CONSTRAINT CONST_STUD_EMAIL CHECK(EMAIL LIKE
'%@MANIPAL.EDU'), PHONE NUMBER(10) UNIQUE CHECK(PHONE LIKE
'__________'), PROGRAM VARCHAR2(6) CONSTRAINT CONST_PROG CHECK(PROGRAM
IN ('BTECH','MTECH','MBA','BBA','MBBS','BCA','MCA')), YEAR_OF_STUDY
NUMBER(4), LOGIN_ID VARCHAR2(9) REFERENCES LOGIN, HOSTEL_NO VARCHAR2(6)
REFERENCES HOSTEL, CONTACT_ID VARCHAR2(5) REFERENCES
EMERGENCY_CONTACTS);

CREATE TABLE AVAILS ( AVAIL_ID VARCHAR2(6) PRIMARY KEY, STUDENT_ID
VARCHAR2(6) REFERENCES STUDENTS, DETAIL_ID VARCHAR2(6) REFERENCES
SERVICE_DETAILS, DATE_AVAILS DATE, STATUS VARCHAR2(20) CHECK(STATUS IN
('CONFIRMED','PENDING','CANCELLED')), TOTAL_COST NUMBER(7)
CHECK(TOTAL_COST > 0));

CREATE TABLE WRITES_REVIEWS( STUDENT_ID VARCHAR2(6) REFERENCES STUDENTS,
DETAIL_ID VARCHAR2(6) REFERENCES SERVICE_DETAILS, RATING NUMBER(2)
CHECK(RATING>0 AND RATING<=10), COMMENTS VARCHAR2(50), DATE_REVIEWED
DATE, PRIMARY KEY(STUDENT_ID, DETAIL_ID));
```

5. Implementation

5.1 Queries

5.1.1 Show top 5 highest rated service providers

```
SELECT sd.Name, AVG(r.Rating) AS Avg_Rating
FROM Service_Details sd
JOIN Writes_Reviews r ON sd.Detail_ID = r.Detail_ID
GROUP BY sd.Name
ORDER BY Avg_Rating DESC
FETCH FIRST 5 ROWS ONLY;
```

5.1.2 Login IDs who have never posted a review or availed a service

```
SELECT Login_ID
FROM Students
WHERE Login_ID NOT IN (
    SELECT DISTINCT s.Login_ID
    FROM Students s
    JOIN Avails a ON s.Student_ID = a.Student_ID
    UNION
    SELECT DISTINCT s.Login_ID
    FROM Students s
    JOIN Writes_Reviews w ON s.Student_ID = w.Student_ID
);
```

5.1.3 Find services with rating greater than 8

```
SELECT Name, Ratings, Price_Avg
FROM Service_Details
WHERE Ratings > 8;
```

5.1.4 Display average cost of services by type

```
SELECT s.Name AS Service_Category, AVG(sd.Price_Avg) AS Avg_Price
FROM Service s
JOIN Service_Details sd ON s.Service_ID = sd.Service_ID
GROUP BY s.Name;
```

5.1.5 Count of services availed by each student

```
SELECT s.Student_ID, s.Name, COUNT(*) AS Services_Availed
FROM Students s
JOIN Avails a ON s.Student_ID = a.Student_ID
GROUP BY s.Student_ID, s.Name;
```

5.1.6 Show reviews for a particular service provider

```
SELECT wr.Student_ID, wr.Rating, wr.Comments
FROM Writes_Reviews wr
WHERE wr.Detail_ID = 'D00005';
```

5.2 Triggers

5.2.1 trg_validate_login (for login check)

Validates login credentials and throws error if password doesn't match. Only valid during a login attempt. Normally done in app layer, but can simulate in DB.

```
CREATE OR REPLACE TRIGGER trg_validate_login
BEFORE INSERT ON Login
FOR EACH ROW
DECLARE
  v_password VARCHAR2(50);
BEGIN
  SELECT Password INTO v_password FROM Login WHERE Login_ID =
:NEW.Login_ID;
  IF v_password != :NEW.Password THEN
    RAISE_APPLICATION_ERROR(-20002, 'Invalid password.');
  END IF;
END;
/
```

```
SET SERVEROUTPUT ON;


DECLARE
  v login id    VARCHAR2(10) := '&login id';
  v_password    VARCHAR2(50) := '&password';
  v_stored_pass VARCHAR2(50);
  v_role        VARCHAR2(10);
BEGIN
```

```
  SELECT Password, Role INTO v_stored_pass, v_role
  FROM Login
  WHERE Login_ID = v_login_id;

  IF v_password = v_stored_pass THEN
    DBMS_OUTPUT.PUT_LINE('Login successful! Role: ' || v_role);
  ELSE
    DBMS_OUTPUT.PUT_LINE('Login failed: Incorrect password.');
  END IF;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Login failed: User does not exist.');
END;
/
```

5.3 Stored Procedures

5.3.1 proc_submit_review

Inserts a review for a service into the Writes_Reviews table along with the current date.

```
CREATE OR REPLACE PROCEDURE proc_submit_review (
  p_stud_id IN VARCHAR2,
  p_detail_id IN VARCHAR2,
  p_rating IN NUMBER,
  p_comment IN VARCHAR2
) AS
BEGIN
  INSERT INTO Writes_Reviews (Student_ID, Detail_ID, Rating, Comments,
Date_Reviewed)
  VALUES (p_stud_id, p_detail_id, p_rating, p_comment, SYSDATE);
END;
/
```

Procedure call:

```
SET SERVEROUTPUT ON;

DECLARE
  v_student_id VARCHAR2(10) := '&student_id';
  v_detail_id  VARCHAR2(10) := '&detail_id';
  v_rating     NUMBER := &rating;
  v_comment    VARCHAR2(200) := '&comment';
BEGIN
  proc_submit_review(v_student_id, v_detail_id, v_rating, v_comment);
END;
```

```
/
```

5.3.2 proc_register_new_student

Registers a new student along with emergency contact and login details, generating IDs automatically.

```
CREATE OR REPLACE PROCEDURE proc_register_new_student (
  p_stud_name       IN VARCHAR2,
  p_stud_email      IN VARCHAR2,
  p_stud_phone      IN NUMBER,
  p_stud_program    IN VARCHAR2,
  p_stud_year       IN NUMBER,
  p_stud_hostel     IN VARCHAR2,
  p_contact_name    IN VARCHAR2,
  p_contact_relation IN VARCHAR2,
  p_contact_number  IN NUMBER,
  p_contact_address IN VARCHAR2,
  p_password        IN VARCHAR2
) IS
  v_login_id     VARCHAR2(10);
  v_student_id   VARCHAR2(6);
  v_contact_id   VARCHAR2(5);
  v_count        NUMBER;
BEGIN
  SELECT COUNT(*) + 1 INTO v_count FROM Login;
  v_login_id := 'L' || LPAD(v_count, 8, '0');
  INSERT INTO Login (Login_ID, Password, Role)
  VALUES (v_login_id, p_password, 'STUDENT');


  SELECT COUNT(*) + 1 INTO v_count FROM Emergency_Contacts;
  v_contact_id := 'C' || LPAD(v_count, 3, '0');
  INSERT   INTO   Emergency_Contacts   (Contact_ID,   Name,   Relation,
Contact_Number, Location)
  VALUES     (v_contact_id,     p_contact_name,     p_contact_relation,
p_contact_number, p_contact_address);


  SELECT COUNT(*) + 1 INTO v_count FROM Students;
  v_student_id := 'M' || LPAD(v_count, 5, '0');
  INSERT INTO Students (
    Student_ID, Name, Email, Phone, Program, Year_of_Study, Login_ID,
Hostel_No, Contact_ID
  )
  VALUES (
    v_student_id,      p_stud_name,      p_stud_email,      p_stud_phone,
p_stud_program,
    p_stud_year, v_login_id, p_stud_hostel, v_contact_id
  );
```

```
  DBMS_OUTPUT.PUT_LINE('Student registered successfully!');
  DBMS_OUTPUT.PUT_LINE('Student_ID: ' || v_student_id);
  DBMS_OUTPUT.PUT_LINE('Login_ID: ' || v_login_id);
  DBMS_OUTPUT.PUT_LINE('Contact_ID: ' || v_contact_id);
END;
/
```

Procedure call:

```
SET SERVEROUTPUT ON;

DECLARE
  v_stud_name       VARCHAR2(50) := '&stud_name';
  v_stud_email      VARCHAR2(50) := '&stud_email';
  v_stud_phone      NUMBER       := &stud_phone;
  v_stud_program    VARCHAR2(10) := '&stud_program';
  v_stud_year       NUMBER       := &stud_year;
  v_stud_hostel     VARCHAR2(6)  := '&stud_hostel';
  v_contact_name    VARCHAR2(50) := '&contact_name';
  v_contact_relation VARCHAR2(20) := '&contact_relation';
  v_contact_number  NUMBER       := &contact_number;
  v_contact_address VARCHAR2(100):= '&contact_address';
  v_password        VARCHAR2(50) := '&password';
BEGIN
  proc_register_new_student(
    p_stud_name        => v_stud_name,
    p_stud_email       => v_stud_email,
    p_stud_phone       => v_stud_phone,
    p_stud_program     => v_stud_program,
    p_stud_year        => v_stud_year,
    p_stud_hostel      => v_stud_hostel,
    p_contact_name     => v_contact_name,
    p_contact_relation => v_contact_relation,
    p_contact_number   => v_contact_number,
    p_contact_address  => v_contact_address,
    p_password         => v_password
  );
END;
/
```

5.3.3 proc_register_new_service

Registers a new service provider and their service details while generating Login_ID and Detail_ID and inserts into Login and Service_Details.

```
Select Service_ID, Name from Service;
CREATE OR REPLACE PROCEDURE proc_register_new_service (
  p_name          IN VARCHAR2,
  p_price         IN NUMBER,
  p_open_time     IN VARCHAR2,
  p_close_time    IN VARCHAR2,
  p_distance      IN NUMBER,
  p_rating        IN NUMBER,
  p_contact       IN NUMBER,
  p_email     IN    VARCHAR2,
  p_password   IN   VARCHAR2,
  p_service_id  IN VARCHAR2
) IS
  v_login_id      VARCHAR2(10);
  v_detail_id     VARCHAR2(6);
  v_count         NUMBER;
BEGIN

  SELECT COUNT(*) + 1 INTO v_count FROM Login;
  v_login_id := 'L' || LPAD(v_count, 8, '0');
  INSERT INTO Login (Login_ID, Password, Role) VALUES (v_login_id,
p_password, 'Service');

  SELECT COUNT(*) + 1 INTO v_count FROM Service_Details;
  v_detail_id := 'D' || LPAD(v_count, 5, '0');

  INSERT INTO Service_Details (
    Detail_ID, Price_Avg, Open_Time, Close_Time, Distance, Ratings,
    Name, Contact_Num, Email, Login_ID, Service_ID
  ) VALUES (
    v_detail_id,
    p_price,
    TO_DATE(p_open_time, 'HH24:MI'),
    TO_DATE(p_close_time, 'HH24:MI'),
    p_distance,
    p_rating,
    p_name,
    p_contact,
    p_email,
    v_login_id,
    p_service_id
  );

  DBMS_OUTPUT.PUT_LINE('Service registered successfully!');
  DBMS_OUTPUT.PUT_LINE('Detail_ID: ' || v_detail_id);
  DBMS_OUTPUT.PUT_LINE('Login_ID: ' || v_login_id);
END;
/
```

Procedure call:

```
SET SERVEROUTPUT ON;

DECLARE
  v_name          VARCHAR2(50) := '&name';
  v_price         NUMBER       := &price;
  v_open_time     VARCHAR2(10) := '&open_time';
  v_close_time    VARCHAR2(10) := '&close_time';
  v_distance      NUMBER       := &distance;
  v_rating        NUMBER       := &rating;
  v_contact       NUMBER       := &contact;
  v_email         VARCHAR2(50) := '&email';
  v_password      VARCHAR2(50) := '&password';
  v_service_id    VARCHAR2(6)  := '&service_id';
BEGIN
  proc_register_new_service(
    p_name          => v_name,
    p_price         => v_price,
    p_open_time     => v_open_time,
    p_close_time    => v_close_time,
    p_distance      => v_distance,
    p_rating        => v_rating,
    p_contact       => v_contact,
    p_email         => v_email,
    p_password      => v_password,
    p_service_id    => v_service_id
  );
END;
/
```

5.3.4 proc_show_services_by_type

Displays service providers matching a specific service type, showing price and rating.

```
CREATE OR REPLACE PROCEDURE proc_show_services_by_type (
  p_service_name IN VARCHAR2
) AS
BEGIN
  FOR rec IN (
    SELECT sd.Name, sd.Price_Avg, sd.Ratings
    FROM Provides p
    JOIN Service_Details sd ON p.Detail_ID = sd.Detail_ID
    WHERE p.Service Name = p service name
```

```
  ) LOOP
    DBMS_OUTPUT.PUT_LINE('Name:  '  ||  rec.Name  ||  ',  Price:  '  ||
rec.Price_Avg || ', Rating: ' || rec.Ratings);
  END LOOP;
END;
/
```

Procedure call:

```
SET SERVEROUTPUT ON;

DECLARE
  v_service_name VARCHAR2(50) := '&service_name';
BEGIN
  proc_show_services_by_type(v_service_name);
END;
/
```

5.3.5 proc_show_service_options

Shows nearby service options sorted by distance from the student's hostel for a given service type.

```
CREATE OR REPLACE PROCEDURE proc_show_service_options (
    p_student_id IN VARCHAR2,
    p_service_name IN VARCHAR2,
    p_service_type IN VARCHAR2
) AS
    v_hostel_no     VARCHAR2(6);
    v_student_dist NUMBER;
    CURSOR c_service_options IS
        SELECT  sd.Detail_ID,  sd.Price_Avg,  sd.Distance,  sd.Ratings,
sd.Open_Time, sd.Close_Time
        FROM Service_Details sd
        JOIN Provides p ON sd.Detail_ID = p.Detail_ID
        WHERE p.Service_Name = p_service_type
        ORDER BY ABS(sd.Distance - v_student_dist);
BEGIN
    SELECT Hostel_No INTO v_hostel_no FROM Students WHERE Student_ID =
p_student_id;
    SELECT Distance INTO v_student_dist FROM Hostel WHERE Hostel_No =
v_hostel_no;

    DBMS_OUTPUT.PUT_LINE('Available   Service   Details   for   '   ||
p_service_name || ' (' || p_service_type || '):');
    FOR rec IN c_service_options LOOP
```

```
        DBMS_OUTPUT.PUT_LINE('Detail ID: ' || rec.Detail_ID || ', Price:
' || rec.Price_Avg ||
                             ', Distance: ' || rec.Distance || ' km,
Rating: ' || rec.Ratings ||
                             ', Open: ' || TO_CHAR(rec.Open_Time,
'HH24:MI') || ', Close: ' || TO_CHAR(rec.Close_Time, 'HH24:MI'));
    END LOOP;
END;
/
```

Procedure call:

```
SET SERVEROUTPUT ON;
DECLARE
  v_student_id   VARCHAR2(10) := '&student_id';
  v_service_name VARCHAR2(50) := '&service_name';
  v_service_type VARCHAR2(50) := '&service_type';
BEGIN
  proc_show_service_options(v_student_id,               v_service_name,
v_service_type);
END;
/
```

5.3.6 proc_book_selected_service

Books a selected service for a student, generates an Avail_ID, and inserts a pending booking.

```
CREATE OR REPLACE PROCEDURE proc_book_selected_service (
    p_student_id IN VARCHAR2,
    p_detail_id IN VARCHAR2
) AS
    v_cost NUMBER;
    v_count NUMBER;
    v_avail_id VARCHAR2(10);
BEGIN

    SELECT Price_Avg INTO v_cost FROM Service_Details WHERE Detail_ID =
p_detail_id;


    SELECT COUNT(*) INTO v_count FROM Avails;

    v_avail_id := 'A' || LPAD(v_count + 1, 5, '0');
```

```
    INSERT INTO Avails (Avail_ID, Student_ID, Detail_ID, Date_Avails,
Status, Total_cost)
    VALUES (v_avail_id, p_student_id, p_detail_id, SYSDATE, 'Pending',
v_cost);

    DBMS_OUTPUT.PUT_LINE('Service booked with Avail_ID ' || v_avail_id
|| ' for Student ID: ' || p_student_id || ' and Detail ID: ' ||
p_detail_id);
END;
/
```

Procedure call:

```
SET SERVEROUTPUT ON;
DECLARE
  v_student_id VARCHAR2(10) := '&student_id';
  v_detail_id  VARCHAR2(10) := '&detail_id';
BEGIN
  proc_book_selected_service(v_student_id, v_detail_id);
END;
/
```

5.3.7 proc_update_avail_status

Updates the status of an existing booking (e.g., from Pending to Confirmed/Cancelled).

```
CREATE OR REPLACE PROCEDURE proc_update_avail_status (
  p_avail_id IN VARCHAR2,
  p_new_status IN VARCHAR2
) AS
BEGIN
  UPDATE Avails
  SET Status = p_new_status
  WHERE Avail_ID = p_avail_id AND Status = 'Pending';

  DBMS_OUTPUT.PUT_LINE('Updated Avail_ID ' || p_avail_id || ' to status:
' || p_new_status);
END;
/
```

Procedure call:

```
SET SERVEROUTPUT ON;

DECLARE
  v_avail_id    VARCHAR2(10) := '&avail_id';
  v_new_status  VARCHAR2(20) := '&new_status';
BEGIN
  proc_update_avail_status(
    p_avail_id    => v_avail_id,
    p_new_status  => v_new_status
  );
END;
/
```

5.3.8 proc_forgot_password

Resets the user's password after validating phone number and login ID match.

```
CREATE OR REPLACE PROCEDURE proc_forgot_password (
  p_login_id     IN VARCHAR2,
  p_phone_number IN NUMBER,
  p_new_password IN VARCHAR2
) AS
  v_stored_phone NUMBER;
  v_role         VARCHAR2(10);
BEGIN
  SELECT s.Phone, l.Role INTO v_stored_phone, v_role
  FROM Students s
  JOIN Login l ON s.Login_ID = l.Login_ID
  WHERE l.Login_ID = p_login_id AND l.Role = 'Student';

  IF v_stored_phone = p_phone_number THEN
    IF LENGTH(p_new_password) >= 8 THEN
      UPDATE Login
      SET Password = p_new_password
      WHERE Login_ID = p_login_id;

      DBMS_OUTPUT.PUT_LINE('Password changed successfully for Login ID:
' || p_login_id);
    ELSE
      DBMS_OUTPUT.PUT_LINE('New password must be at least 8 characters
long.');
    END IF;
  ELSE
    DBMS_OUTPUT.PUT_LINE('Phone number does not match the registered
phone.');
```

```
  END IF;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Login ID not found or role mismatch.');
END;
/
SET SERVEROUTPUT ON;

DECLARE
  v_login_id     VARCHAR2(10) := '&login_id';
  v_phone_number NUMBER       := &phone_number;
  v_new_password VARCHAR2(50) := '&new_password';
BEGIN
  proc_forgot_password(
    p_login_id     => v_login_id,
    p_phone_number => v_phone_number,
    p_new_password => v_new_password
  );
END;
/
```

5.4 Stored Functions

5.4.1 func_get_avg_rating

Calculates the average rating for a given service based on entries in Writes_Reviews.

```
CREATE OR REPLACE FUNCTION func_get_avg_rating(p_detail_id VARCHAR2)
RETURN NUMBER IS
  avg_rating NUMBER;
BEGIN
  SELECT AVG(Rating) INTO avg_rating FROM Writes_Reviews WHERE Detail_ID
= p_detail_id;
  RETURN avg_rating;
END;
/
```

Calls func_get_avg_rating and updates the Ratings column in the Service_Details table.

```
CREATE OR REPLACE PROCEDURE proc_update_rating (
  p_detail_id IN VARCHAR2
) AS
  v_avg_rating NUMBER;
BEGIN
  v_avg_rating := func_get_avg_rating(p_detail_id);
  UPDATE Service_Details
  SET Ratings = ROUND(v_avg_rating)
  WHERE Detail_ID = p_detail_id;

  DBMS_OUTPUT.PUT_LINE('Updated average rating for Detail_ID ' ||
p_detail_id || ' to ' || ROUND(v_avg_rating));
END;
/
```

```
SET SERVEROUTPUT ON;
DECLARE
  v_detail_id VARCHAR2(10) := '&detail_id';
BEGIN
  proc_update_rating(v_detail_id);
END;
/
```

5.4.2 func_total_services_by_student

Returns the total number of services availed by a particular student.

```
CREATE OR REPLACE FUNCTION func_total_services_by_student(p_student_id
VARCHAR2)                        RETURN NUMBER IS
count_services NUMBER;
BEGIN
SELECT COUNT(*) INTO count_services FROM Avails WHERE Student_ID =
p_student_id;
      RETURN count_services;
      END;
      /
```

Displays the count and list of services availed by a student along with date and booking status.

```
CREATE OR REPLACE PROCEDURE proc_list_services_by_student(p_student_id
VARCHAR2) AS
  v_count NUMBER;
```

```
BEGIN
  v_count := func_total_services_by_student(p_student_id);
  DBMS_OUTPUT.PUT_LINE('Total  services  availed  by  student      '   ||
p_student_id || ': ' || v_count);


  FOR rec IN (
    SELECT sd.Name AS Service_Name, a.Date_Avails, a.Status
    FROM Avails a
    JOIN Service_Details sd ON a.Detail_ID = sd.Detail_ID
    WHERE a.Student_ID = p_student_id
  ) LOOP
    DBMS_OUTPUT.PUT_LINE(
      ' - ' || rec.Service_Name ||
      ' | Date: ' || TO_CHAR(rec.Date_Avails, 'YYYY-MM-DD') ||
      ' | Status: ' || rec.Status
    );
  END LOOP;
END;
/
```

```
SET SERVEROUTPUT ON;

DECLARE
  v_student_id VARCHAR2(10) := '&student_id';
BEGIN
  proc_list_services_by_student(v_student_id);
END;
/
```

6. Result

6.1 Registration of Student (5.3.2)

A new student can register into the Student Guide to Manipal by entering the details like Name, Email ID, Phone Number, Program, Year of Admission, Hostel, Emergency Contact Details as well as the Password for future use. The Login ID, Student ID and Contact ID will be auto-generated by the system.

```
Enter value for stud_name: Vaishanvi Rai
old   2:   v_stud_name       VARCHAR2(50) := '&stud_name';
new   2:   v_stud_name       VARCHAR2(50) := 'Vaishanvi Rai';
Enter value for stud_email: vaish@manipal.edu
old   3:   v_stud_email      VARCHAR2(50) := '&stud_email';
new   3:   v_stud_email      VARCHAR2(50) := 'vaish@manipal.edu';
Enter value for stud_phone: 7899854210
old   4:   v_stud_phone      NUMBER       := &stud_phone;
new   4:   v_stud_phone      NUMBER       := 7899854210;
Enter value for stud_program: BTECH
old   5:   v_stud_program    VARCHAR2(10) := '&stud_program';
new   5:   v_stud_program    VARCHAR2(10) := 'BTECH';
Enter value for stud_year: 2023
old   6:   v_stud_year       NUMBER       := &stud_year;
new   6:   v_stud_year       NUMBER       := 2023;
Enter value for stud_hostel: B13
old   7:   v_stud_hostel     VARCHAR2(6)  := '&stud_hostel';
new   7:   v_stud_hostel     VARCHAR2(6)  := 'B13';
Enter value for contact_name: Prakash Rai
old   8:   v_contact_name    VARCHAR2(50) := '&contact_name';
new   8:   v_contact_name    VARCHAR2(50) := 'Prakash Rai';
Enter value for contact_relation: Father
old   9:   v_contact_relation VARCHAR2(20) := '&contact_relation';
new   9:   v_contact_relation VARCHAR2(20) := 'Father';
Enter value for contact_number: 9564421033
old  10:   v_contact_number  NUMBER       := &contact_number;
new  10:   v_contact_number  NUMBER       := 9564421033;
Enter value for contact_address: 12, Kalyan Nagar, Udupi, Karnataka
old  11:   v_contact_address VARCHAR2(100):= '&contact_address';
new  11:   v_contact_address VARCHAR2(100):= '12, Kalyan Nagar, Udupi, Karnataka';
Enter value for password: g24^%HHGgh
old  12:   v_password        VARCHAR2(50) := '&password';
new  12:   v_password        VARCHAR2(50) := 'g24^%HHGgh';
```

```
Student registered successfully!
Student_ID: M00021
Login_ID: L00000042
Contact_ID: C021
```

6.2 Registration of Service Provider (5.3.3)

A new service provider can register into the database by entering their details like the Name of the service, Average Price, Open and Close Time, Ratings, Distance from a fixed point (MIT), Contact Details as well as the Password for future use. The System generates the Detail_ID and Login_ID for the provider.

```
Enter value for name: Manipal Eats
old   2:   v_name         VARCHAR2(50) := '&name';
new   2:   v_name         VARCHAR2(50) := 'Manipal Eats';
Enter value for price: 1000
old   3:   v_price        NUMBER       := &price;
new   3:   v_price        NUMBER       := 1000;
Enter value for open_time: 05:00
old   4:   v_open_time    VARCHAR2(10) := '&open_time';
new   4:   v_open_time    VARCHAR2(10) := '05:00';
Enter value for close_time: 22:00
old   5:   v_close_time   VARCHAR2(10) := '&close_time';
new   5:   v_close_time   VARCHAR2(10) := '22:00';
Enter value for distance: 0.5
old   6:   v_distance     NUMBER       := &distance;
new   6:   v_distance     NUMBER       := 0.5;
Enter value for rating: 10
old   7:   v_rating       NUMBER       := &rating;
new   7:   v_rating       NUMBER       := 10;
Enter value for contact: 9542100365
old   8:   v_contact      NUMBER       := &contact;
new   8:   v_contact      NUMBER       := 9542100365;
Enter value for email: m1eats@gmail.com
old   9:   v_email        VARCHAR2(50) := '&email';
new   9:   v_email        VARCHAR2(50) := 'm1eats@gmail.com';
Enter value for password: mEats2#top
old  10:   v_password     VARCHAR2(50) := '&password';
new  10:   v_password     VARCHAR2(50) := 'mEats2#top';
Enter value for service_id: S00001
old  11:   v_service_id   VARCHAR2(6)  := '&service_id';
new  11:   v_service_id   VARCHAR2(6)  := 'S00001';
Service registered successfully!
Detail_ID: D00021
Login_ID: L00000043
```

6.3 Login (The usual method for logging in to the site) (5.2.1)

```
Enter value for login_id: L0000002
old   2:   v_login_id    VARCHAR2(10) := '&login_id';
new   2:   v_login_id    VARCHAR2(10) := 'L0000002';
Enter value for password: vz7SyOn@XtlD
old   3:   v_password    VARCHAR2(50) := '&password';
new   3:   v_password    VARCHAR2(50) := 'vz7SyOn@XtlD';
Login successful! Role: Student

PL/SQL procedure successfully completed.

SQL> /
Enter value for login_id: L0000003
old   2:   v_login_id    VARCHAR2(10) := '&login_id';
new   2:   v_login_id    VARCHAR2(10) := 'L0000003';
Enter value for password: sdghvhsfrb
old   3:   v_password    VARCHAR2(50) := '&password';
new   3:   v_password    VARCHAR2(50) := 'sdghvhsfrb';
Login failed: Incorrect password.

PL/SQL procedure successfully completed.
```

6.4 Change the old password to a new one if user forgets their old password (5.3.8)

```
Enter value for login_id: L0000005
old   2:   v_login_id     VARCHAR2(10) := '&login_id';
new   2:   v_login_id     VARCHAR2(10) := 'L0000005';
Enter value for phone_number: 9376148890
old   3:   v_phone_number NUMBER       := &phone_number;
new   3:   v_phone_number NUMBER       := 9376148890;
Enter value for new_password: nj32$@JKD#
old   4:   v_new_password VARCHAR2(50) := '&new_password';
new   4:   v_new_password VARCHAR2(50) := 'nj32$@JKD#';
Password changed successfully for Login ID: L0000005

PL/SQL procedure successfully completed.
```

6.5 Displaying the different types of Services Available

```
SQL> SELECT Service_ID, Name FROM Service
  2  ;

SERVIC NAME
------ ----------------------------
S00001 Food and Delivery
S00002 Laundry Services
S00003 Medical and Pharmacy
S00004 Grocery Shop
S00005 Printing and Stationery
S00006 Gym and Fitness
S00007 Cab Booking

7 rows selected.
```

6.6 Searching specific services based on the type of Service (5.3.4)

```
SQL> /
Enter value for service_name: Dry Cleaning
old   2:   v_service_name VARCHAR2(50) := '&service_name';
new   2:   v_service_name VARCHAR2(50) := 'Dry Cleaning';
Name: Dhobimate, Price: 294, Rating: 6

PL/SQL procedure successfully completed.
```

6.7 Searching service sorted by distance from the Student's Location (MIT) (5.3.5)

```
Enter value for student_id: M00020
old   2:   v_student_id   VARCHAR2(10) := '&student_id';
new   2:   v_student_id   VARCHAR2(10) := 'M00020';
Enter value for service_name: Laundry Services
old   3:   v_service_name VARCHAR2(50) := '&service_name';
new   3:   v_service_name VARCHAR2(50) := 'Laundry Services';
Enter value for service_type: Dry Cleaning
old   4:   v_service_type VARCHAR2(50) := '&service_type';
new   4:   v_service_type VARCHAR2(50) := 'Dry Cleaning';
Available Service Details for Laundry Services (Dry Cleaning):
Detail ID: D00002, Price: 294, Distance: .75 km, Rating: 6, Open: 09:00, Close:
23:00
```

6.8 Finding average ratings of the Services based of Student reviews (5.1.1)

```
SQL> SELECT sd.Name, AVG(r.Rating) AS Avg_Rating
  2  FROM Service_Details sd
  3  JOIN Writes_Reviews r ON sd.Detail_ID = r.Detail_ID
  4  GROUP BY sd.Name
  5  ORDER BY Avg_Rating DESC
  6  FETCH FIRST 5 ROWS ONLY;

NAME                           AVG_RATING
------------------------------ ----------
Om Xerox                                9
Campus Store                            9
Dollops                                 9
Wellness Forever                        9
Vitos                                   9
```

6.9 Booking service by using their Detail ID (5.3.6)

```
Enter value for student_id: M00020
old   2:   v_student_id VARCHAR2(10) := '&student_id';
new   2:   v_student_id VARCHAR2(10) := 'M00020';
Enter value for detail_id: D00015
old   3:   v_detail_id  VARCHAR2(10) := '&detail_id';
new   3:   v_detail_id  VARCHAR2(10) := 'D00015';
Service booked with Avail_ID A00019 for Student ID: M00020 and Detail ID: D00015

PL/SQL procedure successfully completed.
```

6.10 Checking status of booking made by the student

```
SQL> select * from Avails where Avail_ID='A00019';

AVAIL_ STUDEN DETAIL DATE_AVAI STATUS               TOTAL_COST
------ ------ ------ --------- -------------------- ----------
A00019 M00020 D00015 13-APR-25 Pending                     168
```

6.11 Confirming the booking (updating booking status) of the student which is carried out by the service provider (5.3.7)

```
Enter value for avail_id: A00019
old   2:   v_avail_id    VARCHAR2(10) := '&avail_id';
new   2:   v_avail_id    VARCHAR2(10) := 'A00019';
Enter value for new_status: Confirmed
old   3:   v_new_status  VARCHAR2(20) := '&new_status';
new   3:   v_new_status  VARCHAR2(20) := 'Confirmed';
Updated Avail_ID A00019 to status: Confirmed
```

6.12 Writing reviews for the service provided to the student (5.3.1)

```
Enter value for student_id: M00003
old   2:   v_student_id VARCHAR2(10) := '&student_id';
new   2:   v_student_id VARCHAR2(10) := 'M00003';
Enter value for detail_id: D00012
old   3:   v_detail_id  VARCHAR2(10) := '&detail_id';
new   3:   v_detail_id  VARCHAR2(10) := 'D00012';
Enter value for rating: 9
old   4:   v_rating     NUMBER := &rating;
new   4:   v_rating     NUMBER := 9;
Enter value for comment: Good service and sweet staff
old   5:   v_comment    VARCHAR2(200) := '&comment';
new   5:   v_comment    VARCHAR2(200) := 'Good service and
 sweet staff';

PL/SQL procedure successfully completed.
```

6.13 Checking the number of bookings made by the students (specifically those who have made any bookings) (5.1.5)

```
SQL> SELECT s.Student_ID, s.Name, COUNT(*) AS Services_Availed
  2  FROM Students s
  3  JOIN Avails a ON s.Student_ID = a.Student_ID
  4  GROUP BY s.Student_ID, s.Name;

STUDEN NAME                                 SERVICES_AVAILED
------ ---------------------------------------- ---------------
M00002 Priya Desai                                         1
M00003 Rohan Mehta                                         2
M00004 Ishita Nair                                         1
M00006 Megha Bhatia                                        2
M00008 Aanya Rao                                           1
M00009 Aditya Pillai                                       3
M00014 Tanvi Singh                                         1
M00015 Ankit Joshi                                         1
M00016 Shreya Patel                                        1
M00017 Manav Kapoor                                        3
M00019 Harsh Vora                                          2

STUDEN NAME                                 SERVICES_AVAILED
------ ---------------------------------------- ---------------
M00020 Diya Jain                                           1
```

6.14 Checking the number of bookings made by a specific student using their student id (5.4.2)

```
Enter value for student_id: M00003
old   2:   v_student_id VARCHAR2(10) := '&student_id';
new   2:   v_student_id VARCHAR2(10) := 'M00003';
Total services availed by student M00003: 2
- Crystal Clean Laundry | Date: 2025-03-13 | Status: Cancelled
- Kasturba Medical Hospital | Date: 2025-03-24 | Status: Confirmed

PL/SQL procedure successfully completed.
```

6.15 Show reviews for a particular service provider (5.1.6)

```
SQL> SELECT wr.Student_ID, wr.Rating, wr.Comments
  2  FROM Writes_Reviews wr
  3  WHERE wr.Detail_ID = 'D00005';

STUDEN     RATING COMMENTS
------ ---------- --------------------------------------------------
M00005          6 Affordable and reliable service near hostel.
```

6.16 Lists out all the students who have never submitted a review (5.1.2)

```
SQL> SELECT Login_ID
  2  FROM Students
  3  WHERE Login_ID NOT IN (
  4      SELECT DISTINCT s.Login_ID
  5      FROM Students s
  6      JOIN Avails a ON s.Student_ID = a.Student_ID
  7      UNION
  8      SELECT DISTINCT s.Login_ID
  9      FROM Students s
 10      JOIN Writes_Reviews w ON s.Student_ID = w.Student_ID
 11  );

LOGIN_ID
---------
L0000019
L00000042
```

6.17 Displays all the services that have their ratings above 8 (5.1.3)

```
SQL> SELECT Name, Ratings, Price_Avg
  2  FROM Service_Details
  3  WHERE Ratings > 8;

NAME                          RATINGS  PRICE_AVG
----------------------------- ---------- ----------
Manipal Eats                          10       1000
Om Xerox                               9        184
Marena                                 9        369
Campus Store                           9        128
Radha Medicals                        10        319
Dollops                               10        364
Kasturba Medical Hospital             10        291
Wellness Forever                       9        228
FreshMart                              9        376
Pai Tiffins                           10        242

10 rows selected.
```

6.18 Displays the average cost of the different types of services (5.1.4)

```
SQL> SELECT s.Name AS Service_Category, AVG(sd.Price_Avg) AS Avg_Price
  2  FROM Service s
  3  JOIN Service_Details sd ON s.Service_ID = sd.Service_ID
  4  GROUP BY s.Name;

SERVICE_CATEGORY               AVG_PRICE
------------------------------ ----------
Food and Delivery                   458.8
Printing and Stationery               209
Laundry Services                      231
Gym and Fitness                     270.5
Grocery Shop                   226.666667
Medical and Pharmacy           279.333333
Cab Booking                    273.666667

7 rows selected.
```

6.19 Updates the rating of a service with the average rating given for that service detail (5.4.1)

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2    v_detail_id VARCHAR2(10) := '&detail_id';
  3  BEGIN
  4    proc_update_rating(v_detail_id);
  5  END;
  6  /
Enter value for detail_id: D00004
old   2:   v_detail_id VARCHAR2(10) := '&detail_id';
new   2:   v_detail_id VARCHAR2(10) := 'D00004';
Updated average rating for Detail_ID D00004 to 9

PL/SQL procedure successfully completed.
```

7. Conclusion and Future Work

7.1 Conclusion

The Student Guide to Manipal project successfully implements a robust and efficient backend database system that caters to the needs of students relocating to a new environment. It provides structured access to essential services such as food, laundry, gyms, medical stores, grocery outlets, hostels, and more. The system ensures seamless interaction between students and service providers by offering modules for secure registration, login authentication, service booking, review submission, and data analytics.

With features like auto-generation of unique IDs, login verification, distance-based service filtering, and modular procedures for various interactions, the system demonstrates the effectiveness of SQL and PL/SQL in handling real-world service management use cases. The triggers and functions ensure data integrity, while reports and queries aid in analyzing usage trends and service quality. The project achieves its objective of making student transitions smoother and improving service accessibility in and around the Manipal Institute of Technology campus.

7.2 Scope for Future Work

While the current system operates as a backend database, it opens up several possibilities for future enhancements:

- A user-friendly frontend web or mobile application can be developed to interact with the database, enabling live service booking and real-time updates.
- A payment gateway module can be integrated for online service payments and order tracking.
- Geolocation features can be added to dynamically calculate distance from the user's current location instead of hostel blocks.
- Admin panel functionalities can be extended to manage user accounts, deactivate outdated service providers, and view feedback reports.

- AI/ML integration could be used for personalized service recommendations based on past booking behavior.

By expanding the current structure and incorporating these features, the Student Guide to Manipal system can evolve into a complete digital assistant for students and campus service providers alike.