
**DTD3013 (Group B3)**

**Group assignment : Database System**

# KOLEJ HOSTEL REGISTRATION SYSTEM



| | |
|---|---|
| **WONG CHUN LIANG** | **D20231106337** |
| **GOH QUO TENG** | **D20231106372** |
| **NUR IZZATI BINTI MD RAJMI** | **D20231106358** |



**Professor Madya Dr. Aslina Bt Saad**

## 1.0 PROJECT INTRODUCTION

This group project is developed for the DTD3013 Database System course. The goal of this project is to design a database system named Kolej Hostel Registration System, which aims to digitize and improve the current manual hostel registration process used in university college environments.

Currently, students need to queue and fill out paper forms, and staff must manually check documents and allocate rooms. Our proposed system replaces this outdated method with a centralized database and web-based registration system that improves accuracy, efficiency, and user experience. It supports online applications, automatic room assignments, payment verification, and report generation, all while reducing paperwork and administrative workload.

This project applies core database concepts including entity-relationship modeling, normalization, and CRUD (Create, Read, Update, Delete) operations to ensure that the system is not only functional but scalable and maintainable in the long term. Furthermore, strict business rules such as gender-based and race-based room allocation are integrated into the design to reflect actual practices within Malaysian higher education institutions.

The following section will examine the identity of the problem in detail and outline the challenges encountered in the current hostel registration system, thereby establishing the foundation for the system we propose.

## 2.0 INDENTITY PROBLEM

The implementation of an effective hostel registration system is crucial to ensure that university students are accommodated in a timely and organized manner. However, the current process used by many educational institutions—including manual form submissions, physical queues, and paper-based records—has proven to be inefficient, error-prone, and difficult to manage, especially during peak registration periods. These outdated methods not only cause delays for students but also increase the administrative burden on hostel management staff.

Without a digital solution in place, issues such as double room bookings, misplaced forms, and a lack of centralized data access frequently occur. Students may not receive confirmation on time, while staff face difficulties in verifying payment records, checking room availability, and updating allocation statuses. Moreover, the absence of real-time data often results in inaccurate or outdated hostel information, which further complicates room assignment decisions.

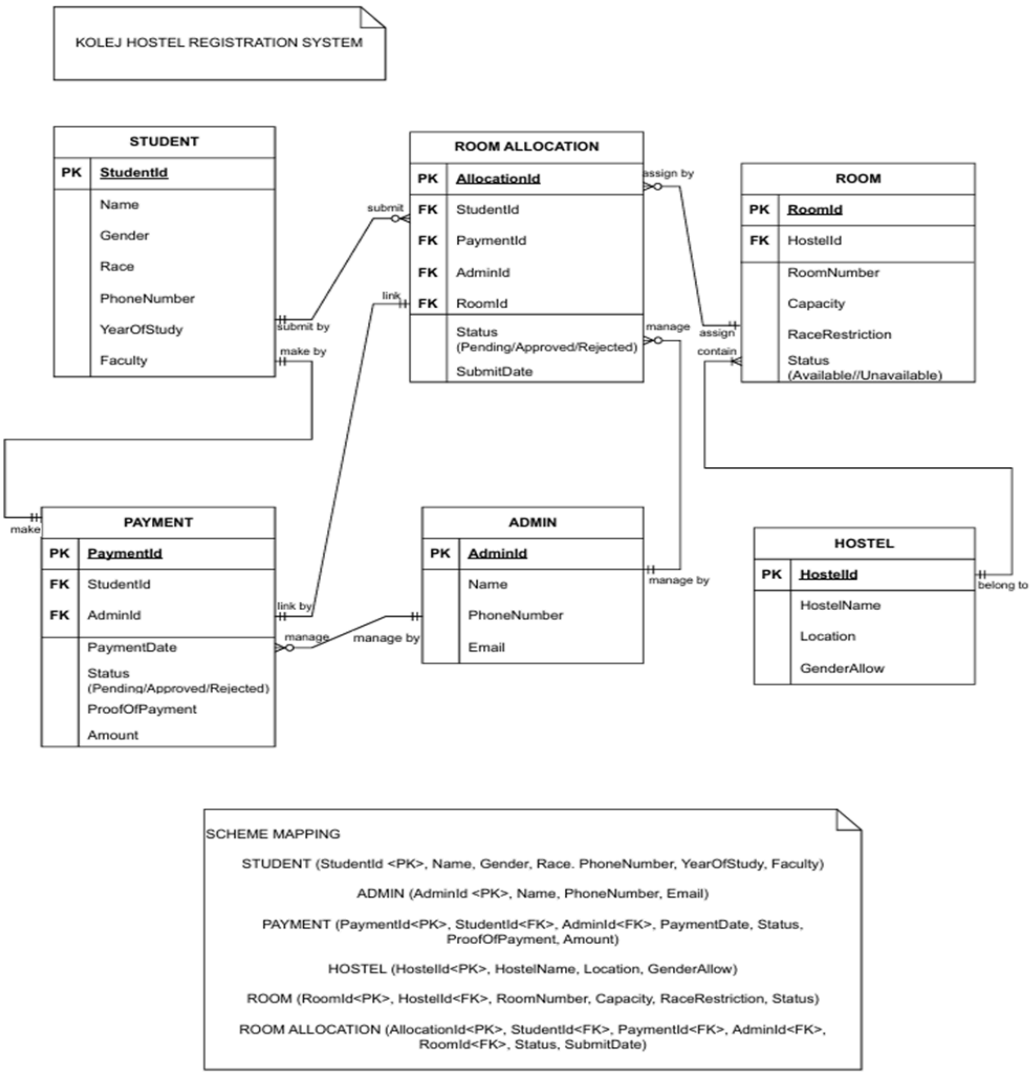
To better understand and address these challenges, this section identifies the key problems of the current system by analyzing its background, abstraction of issues, system objectives, and the required data operations. The tables below summarize the essential elements that guided the design of our proposed Kolej Hostel Registration System.

| **Background (Current situation/scenario)** | **Problem (Abstraction)** | **Objectives (Abstraction)** | **CRUD (Pattern recognition)** | **New Scenario with complete business rules** |
|---|---|---|---|---|
| - Students need to line up to fill out physical forms for kolej hostel registration.<br><br>- Staff must manually verify student details and payments.<br><br>- Lack of real-time data for available rooms and occupancy.<br><br>- Difficulty in tracking student records and payment status. | - Slow processing time for registrations.<br><br>- Inaccurate or outdated room availability information.<br><br>- Inefficient tracking of student records and payments.<br><br>- Lack of centralized access to kolej hostel data.<br><br>- Risk of data | - Allow students to apply for hostel rooms online.<br><br>- Automatic or assisted room assignments based on race and gender.<br><br>- Easily store, view, and update student and room records.<br><br>- Track current occupants, vacant rooms, and generate reports. | - Students can **create** a new account.<br><br>- Students can **read** student details and room assignments.<br><br>- Students can **update** student information or change rooms.<br><br>- Students can **delete** a student from the system (e.g., when they leave | -Student must register online using their detail in order to use the Upsi hostel registration system.<br>-This system saves time and facilitates the management of business activities.<br>-The Upsi hostel registration system make it simple to update student report and handle hostel administration |

| | | | | |
|---|---|---|---|---|
| - No centralized database for hostel information.<br><br>- High dependency on paperwork increases risk of loss or damage.<br><br>- Allocation of rooms is done manually, which may lead to double bookings. | loss or misplaced physical forms.<br><br>-Increased chances of double-booking rooms. | - Enable hostel staff to manage student info, room availability, and application approvals/rejections. | the hostel).<br><br>- Admin can **create** a new admin/staff user account.<br><br>- Admin can **read** the list of staff/admins.<br><br>- Admin can **update** roles or permissions.<br><br>- Admin can **delete** an admin/staff account. | supply.<br><br>-Each student must makes one payment<br><br>-Each student must submit one or many room allocation(registration form)<br><br>-Each room may assigned in many room allocation<br><br>-Each hostel must contain many room |

### 3.0 Entity Relationship Diagram（ERD）

The Entity Relationship Diagram (ERD) is a crucial component in the planning and development of a well-structured database system. It visually represents the entities involved in the Kolej Hostel Registration System and outlines how these entities relate to each other. This diagram serves as the foundation for understanding the logical structure of the system before actual implementation. In our system, key entities include Student, Admin, Hostel, Room, Payment, and Room Allocation. Each entity has its own attributes and is uniquely identified by a primary key. Relationships between entities are represented through foreign keys, ensuring that data across tables remains connected and consistent. For instance, a single student may be linked to one or more payments and room allocations, and each room allocation must be approved by an admin. The ERD enables us to map real-world processes such as student registration, room booking, and payment verification into a digital framework. It also ensures that normalization rules are followed, avoiding data redundancy and maintaining data integrity throughout the system. The diagram below presents the complete ERD for the Kolej Hostel Registration System.




SCHEME MAPPING

STUDENT (StudentId <PK>, Name, Gender, Race. PhoneNumber, YearOfStudy, Faculty)

ADMIN (AdminId <PK>, Name, PhoneNumber, Email)

PAYMENT (PaymentId<PK>, StudentId<FK>, AdminId<FK>, PaymentDate, Status, ProofOfPayment, Amount)

HOSTEL (HostelId<PK>, HostelName, Location, GenderAllow)

ROOM (RoomId<PK>, HostelId<FK>, RoomNumber, Capacity, RaceRestriction, Status)

ROOM ALLOCATION (AllocationId<PK>, StudentId<FK>, PaymentId<FK>, AdminId<FK>, RoomId<FK>, Status, SubmitDate)

## 4.0 RELATIONAL MODEL

Following the design of the ERD, the next step in building the database is translating the entities and their relationships into relational tables. This relational model provides a detailed structure of how data will be stored, including table names, attributes, data types, and relationships between tables.

Each table is designed with a primary key to ensure the uniqueness of records. Foreign keys are used to establish logical connections between related tables. For example, the Payment table includes references to both Student and Admin, ensuring that every payment is properly linked to the individual who made it and the staff member who processed it.

This relational model also ensures that each table adheres to normalization principles, which eliminates data duplication and simplifies future maintenance. The tables defined in our system collectively form a reliable and scalable database structure capable of handling real hostel registration scenarios. The following section outlines the relational tables that have been created for this system.

**STUDENT**

| ***StudentId*<PK>** | **Name** | **Gender** | **Race** | **PhoneNumber** | **YearOf Study** | **Faculty** |
|---|---|---|---|---|---|---|
| D20231106001 | AHMAD FAIZ BIN ROSLAN | Male | Malay | 011-1111111 | 2 | FKMT |
| D20231106002 | TAN WEI LING | Female | Chinese | 011-1111112 | 1 | FPE |
| D20231106003 | ARUN KUMAR A/L RAJ | Male | Indian | 011-1111113 | 2 | FKMT |
| D20231106004 | NURUL AIN BINTI ZAINAL | Female | Malay | 011-1111114 | 4 | FSK |
| D20231106005 | LIM KAI WEN | Male | Chinese | 011-1111115 | 1 | FBK |

**ADMIN**

| AdminId<PK> | Name | PhoneNumber | Email |
|---|---|---|---|
| A001 | LEE WEI MING | 012-2222221 | weiming.lee@hostel.com |
| A002 | MUHAMMAD HAZIQ BIN AZMI | 012-2222222 | haziq.azmi@hostel.com |
| A003 | NORAINA BINTI ZAKARIA | 012-2222223 | noraina@hostel.com |
| A004 | KAVITHA A/P SELVAN | 012-2222224 | kavitha.s@hostel.com |
| A005 | SITI NUR BALQIS BINTI MOHD | 012-2222225 | balqis.mohd@hostel.com |

**PAYMENT**

| PaymentId <PK> | StudentId <FK> | AdminId <FK> | PaymentDate | Status | ProofOfPayment | Amount |
|---|---|---|---|---|---|---|
| P001 | D20231106001 | A001 | 01/06/2025 | Approved | proof_p001.jpg | 506.00 |
| P002 | D20231106002 | A001 | 01/06/2025 | Approved | proof_p002.jpg | 506.00 |
| P003 | D20231106003 | A001 | 01/06/2025 | Rejected | proof_p003.jpg | 506.00 |
| P004 | D20231106004 | A003 | 01/06/2025 | Pending | proof_p004.jpg | 506.00 |
| P005 | D20231106005 | A004 | 01/06/2025 | Pending | proof_p005.jpg | 506.00 |

**HOSTEL**

| HostelId<PK> | HostelName | Location | GenderAllow |
|---|---|---|---|
| H001 | Kolej Khar | Jalan Khar 1 | Male |
| H002 | Kolej Ksas | Jalan Ksas 1 | Male |
| H003 | Kolej Kuo | Jalan Kuo 1 | Male |
| H004 | Kolej Kab | Jalan Kab 1 | Female |
| H005 | Kolej Ksjas | Jalan Ksjas 1 | Female |

**ROOM**

| RoomId <PK> | HostelID <FK> | RoomNumber | Capacity | RaceRestriction | Status |
|---|---|---|---|---|---|
| R001 | H001 | 001 | 2 | Chinese | Available |
| R002 | H001 | 002 | 2 | Chinese | Available |
| R003 | H001 | 101 | 2 | Malay | Available |
| R004 | H001 | 102 | 2 | Indian | Available |
| R005 | H001 | 201 | 2 | Malay | Available |

**ROOM ALLOCATION**

| AllocationId <PK> | StudentId <FK> | PaymentId <FK> | AdminId <FK> | RoomId <FK> | Status | SubmitDate |
|---|---|---|---|---|---|---|
| RA001 | D20231106001 | P001 | H001 | R001 | PENDING | 01/06/2025 |
| RA002 | D20231106002 | P002 | A001 | R002 | APPROVED | 01/06/2025 |

| | | | | | | |
|---|---|---|---|---|---|---|
| RA003 | D20231106003 | P003 | A001 | R003 | APPROVED | 01/06/2025 |
| RA004 | D20231106004 | P004 | A003 | R004 | APPROVED | 01/06/2025 |
| RA005 | D20231106005 | P005 | A004 | R005 | PENDING | 01/06/2025 |

## 5.0 DATA DICTIONARY

To support clarity and transparency in system development, a comprehensive data dictionary is included as part of our documentation. The data dictionary lists all the fields used in each table, alongside their data types, size limits, descriptions, and constraints.

Each field is carefully defined to match the system's data requirements. For instance, the StudentId field uses the VARCHAR type to accommodate alphanumeric IDs, while the PaymentDate field uses the DATE type for proper chronological tracking. Constraints such as Primary Key (PK) and Foreign Key (FK) are applied to ensure data consistency and integrity across the entire system.

In addition to improving understanding for developers and users, the data dictionary also plays a vital role in future system maintenance, troubleshooting, and expansion. It acts as a reference guide for anyone interacting with the database, making it easier to update or integrate with other systems. The following tables provide detailed definitions for each data entity used in the Kolej Hostel Registration System.

**Student**

| DATA NAME | DATA TYPE | SIZE | DESCRIPTION | CONSTRAINT |
|---|---|---|---|---|
| **StudentId** | VARCHAR | 12 | Unique ID for each student | Primary Key (PK) |
| **Name** | VARCHAR | 100 | Full name of the student | Null |
| **Gender** | VARCHAR | 6 | Gender of the student<br>(Male/Female) | Null |
| **Race** | VARCHAR | 10 | Race of Student<br>(Chinese/Malay/Indian/Bumiputera/ Other) | Null |
| **PhoneaNumber** | VARCHAR | 15 | Contact number of the student | Null |
| **YearOfStudy** | INT | - | Current year of study | Null |
| **Faculty** | VARCHAR | 10 | Faculty of the student | Null |

**Admin**

| DATA NAME | DATA TYPE | SIZE | DESCRIPTION | CONSTRAINT |
|---|---|---|---|---|
| AdminId | VARCHAR | 5 | Unique ID for each admin | Primary Key(PK) |
| Name | VARCHAR | 100 | Full name of the admin | Null |
| PhoneNumber | VARCHAR | 15 | Contact number of the admin | Null |
| Email | VARCHAR | 100 | Email address | Null |

**Payment**

| DATA NAME | DATA TYPE | SIZE | DESCRIPTION | CONSTRAINT |
|---|---|---|---|---|
| PaymentId | VARCHAR | 5 | Unique payment transaction ID | Primary Key(PK) |
| StudentId | VARCHAR | 12 | Student who made the payment | Foreign Key(FK) |
| AdminId | VARCHAR | 5 | Admin who processed the payment | Foreign Key(FK) |
| PaymentDate | DATE | - | Date when payment was made | Null |
| Status | VARCHAR | 10 | Status of the payment | Null |
| ProofOfPayment | FILE (PATH) | 255 | File path or URL to uploaded payment proof | Null |
| Amount | DECIMAL | 10,2 | Amount paid | Null |

**Hostel**

| DATA NAME | DATA TYPE | SIZE | DESCRIPTION | CONSTRAINT |
|---|---|---|---|---|
| HostelId | VARCHAR | 5 | Unique hostel ID | Primary Key(PK) |
| HostelName | VARCHAR | 50 | Name of the hostel | Null |
| Location | VARCHAR | 100 | Location/address of the hostel | Null |
| GenderAllow | VARCHAR | 6 | Gender allow<br><br>(Male/Female) | Null |

**Room**

| DATA NAME | DATA TYPE | SIZE | DESCRIPTION | CONSTRAINT |
|---|---|---|---|---|
| RoomId | VARCHAR | 5 | Unique room ID | Primary Key(PK) |
| HostelId | VARCHAR | 5 | Hostel to which the room belongs | Foreign Key(FK) |
| RoomNumber | VARCHAR | 5 | Room number | Null |
| Capacity | INT | - | Maximum number of occupants | Null |
| RaceRestriction | VARCHAR | 10 | Room restriction for room | Null |
| Status | VARCHAR | 10 | Room availability status | Null |

**Room Allocation**

| DATA NAME | DATA TYPE | SIZE | DESCRIPTION | CONSTRAINT |
|---|---|---|---|---|
| AllocationId | VARCHAR | 6 | Unique ID for the allocation | Primary Key(PK) |
| StudentId | VARCHAR | 12 | Student who is allocated the room | Foreign Key(FK) |
| PaymentId | VARCHAR | 5 | Payment associated with the allocation | Foreign Key(FK) |
| AdmintId | VARCHAR | 5 | Admin who approved the allocation | Foreign Key(FK) |
| RoomId | VARCHAR | 5 | Allocated room | Foreign Key(FK) |
| Stautus | VARCHAR | 10 | Allocation status (Pending, Approved, Rejected) | Null |
| SubmitDate | Date | - | Date of submit | Null |

## 6.0 SQL STATEMENT

This section provides the full set of SQL statements used to implement the Kolej Hostel Registration System database. It includes the creation of tables (Data Definition Language - DDL), insertion of sample data (Data Manipulation Language - DML), and the application of constraints, keys, and relationships to ensure data consistency and accuracy. These SQL scripts are essential for translating the system's design into a working database, allowing structured storage of student, admin, hostel, payment, and room allocation information. By executing these statements, we ensure that the database not only reflects the real-world needs of the hostel registration process but also supports the core functionalities of the system such as online registration, room assignment, and payment verification. This structured approach improves efficiency, minimizes redundancy, and enforces referential integrity across all records in the system.

### 6.1 DATA DEFINITION LANGUAGE (DDL)

1. Create database

The following SQL statement creates a database called kolej_hostel_registration_system to store all related tables and data for the hostel registration system.

```
CREATE DATABASE kolej_hostel_registration_system;
```

2. Delete database

The following SQL statement deletes the entire kolej_hostel_registration_system database, including all its tables and stored data.

```
DROP DATABASE kolej_hostel_registration_system;
```

3. Create table

The following SQL statement creates a table with specific columns and data types to store structured information in the database.

```sql
CREATE TABLE Student(
    StudentId VARCHAR(12) NOT NULL,
    Name VARCHAR(100),
    Gender VARCHAR(6),
    Race VARCHAR(10),
    PhoneNumber VARCHAR(15),
    YearOfStudy INT,
    Faculty VARCHAR(10)
);
```

```sql
CREATE TABLE Admin(
    AdminId VARCHAR(5) NOT NULL,
    Name VARCHAR(100),
    PhoneNumber VARCHAR(15),
    Email VARCHAR(100)
);
```

```sql
CREATE TABLE Payment (
    PaymentId VARCHAR(5) NOT NULL,
    StudentId VARCHAR(12),
    AdminId VARCHAR(5),
    PaymentDate DATE,
    Status VARCHAR(10),
    ProofOfPayment VARCHAR(255),
    Amount DECIMAL(10,2)
);
```

```sql
CREATE TABLE Hostel (
    HostelId VARCHAR(5) NOT NULL,
    HostelName VARCHAR(50),
    Location VARCHAR(100),
    GenderAllow VARCHAR(6)
);
```

```sql
CREATE TABLE Room (
    RoomId VARCHAR(5) NOT NULL,
    HostelId VARCHAR(5),
    RoomNumber VARCHAR(5),
    Capacity INT,
    RaceRestriction VARCHAR(10),
    Status VARCHAR(10)
);
```

```sql
CREATE TABLE RoomAllocation (
    AllocationId VARCHAR(6) NOT NULL,
    StudentId VARCHAR(12),
    PaymentId VARCHAR(5),
    AdminId VARCHAR(5),
    RoomId VARCHAR(5),
    Status VARCHAR(10),
    SubmitDate DATE
);
```

4. Modify table

ADD PRIMARY KEY

The following SQL statement adds a primary key to a table, ensuring that each record can be uniquely identified.

```sql
ALTER TABLE student ADD PRIMARY KEY (StudentId);
```

```sql
ALTER TABLE admin ADD PRIMARY KEY (AdminId);
```

```sql
ALTER TABLE payment ADD PRIMARY KEY (PaymentId);
```

```sql
ALTER TABLE hostel ADD PRIMARY KEY (HostelId);
```

```sql
ALTER TABLE room ADD PRIMARY KEY (RoomId);
```

```sql
ALTER TABLE roomallocation ADD PRIMARY KEY (AllocationId);
```

ADD FOREIGN KEY

The following SQL statement adds a foreign key to a table to establish a relationship between two tables, enforcing referential integrity.

```sql
ALTER TABLE payment
ADD FOREIGN KEY (StudentId) REFERENCES student (StudentId),
ADD FOREIGN KEY (AdminId) REFERENCES admin (AdminId);
```

```sql
ALTER TABLE room ADD FOREIGN KEY (HostelId) REFERENCES hostel (HostelId);
```

```sql
ALTER TABLE roomallocation
ADD FOREIGN KEY (StudentId) REFERENCES student (StudentId),
ADD FOREIGN KEY (PaymentId) REFERENCES payment (PaymentId),
ADD FOREIGN KEY (AdminId) REFERENCES admin (AdminId),
ADD FOREIGN KEY (RoomId) REFERENCES room (RoomId);
```

5. Delete table

The following SQL statement deletes a table from the database along with all its stored data.

```sql
DROP TABLE admin;
```

### 6.2 DATA MANIPULATION LANGUAGE (DML)

Compulsory

1. Create data

The following SQL statement inserts new data (a record) into a specified table in the database.

```sql
INSERT INTO Student (StudentId, Name, Gender, Race, PhoneNumber, YearOfStudy,
Faculty) VALUES
('D20231106001', 'AHMAD FAIZ BIN ROSLAN', 'Male', 'Malay', '011-11111111', 2,
'FKMT'),
('D20231106002', 'TAN WEI LING', 'Female', 'Chinese', '011-11111112', 1, 'FPE'),
('D20231106003', 'ARUN KUMAR A/L RAJ', 'Male', 'Indian', '011-11111113', 2,
'FKMT'),
('D20231106004', 'NURUL AIN BINTI ZAINAL', 'Female', 'Malay', '011-11111114', 4,
'FSK'),
('D20231106005', 'LIM KAI WEN', 'Male', 'Chinese', '011-11111115', 1, 'FBK');
```

```sql
INSERT INTO admin (AdminId, Name, PhoneNumber, Email) VALUES
('A001', 'LEE WEI MING', '012-2222221', 'weiming.lee@hostel.com'),
('A002', 'MUHAMMAD HAZIQ BIN AZMI', '012-2222222', 'haziq.azmi@hostel.com'),
('A003', 'NORAINA BINTI ZAKARIA', '012-2222223', 'noraina@hostel.com'),
('A004', 'KAVITHA A/P SELVAN', '012-2222224', 'kavitha.s@hostel.com'),
('A005', 'SITI NUR BALQIS BINTI MOHD', '012-2222225', 'balqis.mohd@hostel.com');
```

```sql
INSERT INTO Payment (PaymentId, StudentId, AdminId, PaymentDate, Status,
ProofOfPayment, Amount) VALUES
('P001', 'D20231106001', 'A001', '2025-06-01', 'Approved', 'proof_p001.jpg',
506.00),
('P002', 'D20231106002', 'A001', '2025-06-01', 'Approved', 'proof_p002.jpg',
506.00),
('P003', 'D20231106003', 'A001', '2025-06-01', 'Rejected', 'proof_p003.jpg',
506.00),
('P004', 'D20231106004', 'A004', '2025-06-01', 'Pending', 'proof_p004.jpg',
506.00),
('P005', 'D20231106005', 'A004', '2025-06-01', 'Pending', 'proof_p005.jpg',
506.00);
```

```sql
INSERT INTO Hostel (HostelId, HostelName, Location, GenderAllow) VALUES
('H001', 'Kolej Khar', 'Jalan Khar 1', 'Male'),
('H002', 'Kolej Ksas', 'Jalan Ksas 1', 'Male'),
('H003', 'Kolej Kuo', 'Jalan Kuo 1', 'Male'),
('H004', 'Kolej Kab', 'Jalan Kab 1', 'Female'),
('H005', 'Kolej Ksjas', 'Jalan Ksjas 1', 'Female');
```

```sql
INSERT INTO room (RoomId, HostelID, RoomNumber, Capacity, RaceRestriction, Status)
VALUES
('R001', 'H001', '001', 2, 'Chinese', 'Available'),
('R002', 'H001', '002', 2, 'Chinese', 'Available'),
('R003', 'H001', '101', 2, 'Malay', 'Available'),
('R004', 'H001', '102', 2, 'Indian', 'Available'),
('R005', 'H001', '201', 2, 'Malay', 'Available');
```

```sql
INSERT INTO roomallocation (AllocationId, StudentId, PaymentId, AdminId, RoomId,
Status, SubmitDate) VALUES
('RA001', 'D20231106001', 'P001', 'A001', 'R001', 'PENDING', '2025-06-01'),
('RA002', 'D20231106002', 'P002', 'A001', 'R001', 'APPROVED', '2025-06-01'),
('RA003', 'D20231106003', 'P003', 'A002', 'R003', 'APPROVED', '2025-06-01'),
('RA004', 'D20231106004', 'P004', 'A003', 'R004', 'APPROVED', '2025-06-01'),
('RA005', 'D20231106005', 'P005', 'A004', 'R005', 'PENDING', '2025-06-01');
```

2. Read data

The following SQL statement retrieves data from a table so that it can be viewed or processed.

```sql
SELECT StudentId, Name, Faculty
FROM student;
```

```sql
SELECT payment.PaymentId, student.Name, payment.PaymentDate
FROM payment
INNER JOIN student ON payment.StudentId = student.StudentId;
```

```sql
SELECT payment.PaymentId, student.Name, admin.Name
FROM payment
INNER JOIN student ON payment.StudentId = student.StudentId
INNER JOIN admin ON payment.AdminId = admin.AdminId;
```

3. Update data

The following SQL statement updates existing data in a table, modifying specific values based on a given condition.

```sql
UPDATE student
SET name = 'Goh Quo Teng', Gender = 'Male'
WHERE StudentId = 'D20231106002';
```

4. Delete data

The following SQL statement removes one or more records from a table based on a specified condition.

```sql
DELETE FROM student WHERE Name = 'Goh Quo Teng';
```

5. Sort data

The following SQL statement sorts the retrieved data in either ascending or descending order according to one or more columns.

```sql
SELECT * FROM student
ORDER BY name ASC;
```

```sql
SELECT * FROM student
ORDER BY name DESC;
```

6. Group by

The following SQL statement groups rows that have the same values in specified columns into summary rows.

```sql
SELECT Gender, COUNT(StudentId) AS NumberOfStudents FROM student
GROUP BY Gender;
```

7. Max

The following SQL statement finds the maximum value from a column in the table.

```sql
SELECT MAX(YearOfStudy) AS LongestYearOfStudy FROM student;
```

8. Min

The following SQL statement finds the minimum value from a column in the table.

```sql
SELECT MIN(YearOfStudy) AS ShortestYearOfStudy FROM student;
```

9. Average

The following SQL statement calculates the average value of a numeric column in the table.

```sql
SELECT AVG(YearOfStudy) FROM student;
```

### 6.3 Option

1. = equal to

The following SQL statement uses the = operator to filter and return rows that match an exact value.

```
SELECT * FROM roomallocation WHERE AdminId = 'A001';
```

2. > greater than

The following SQL statement uses the > operator to return rows where the column value is greater than the specified number.

```
SELECT * FROM student WHERE YearOfStudy > 2;
```

3. >= greater than or equal to

The following SQL statement uses the >= operator to return rows where the column value is greater than or equal to a given number.

```
SELECT * FROM student WHERE YearOfStudy >= 2;
```

4. less than

The following SQL statement uses the < operator to return rows where the column value is less than a specified value.

```
SELECT * FROM student WHERE YearOfStudy < 2;
```

5. <= less than or equal to

The following SQL statement uses the <= operator to return rows where the column value is less than or equal to a given value.

```sql
SELECT * FROM student WHERE YearOfStudy <= 2;
```

6. BETWEEN

The following SQL statement uses BETWEEN to filter data within a specified range.

```sql
SELECT * FROM student WHERE YearOfStudy BETWEEN 1 and 3;
```

7. …AND

The following SQL statement uses the AND operator to combine multiple conditions in a WHERE clause, returning rows that meet all specified criteria.

```sql
SELECT * FROM roomallocation WHERE AdminId = 'A001' AND Status = 'APPROVED';
```

8. IN

The following SQL statement uses the IN operator to check if a value exists in a list of specified values.

```sql
SELECT * FROM roomallocation WHERE RoomId IN ('R001', 'R005');
```

9. LIKE

The following SQL statement uses the LIKE operator to search for a specified pattern in a column, often using wildcard characters.

```sql
SELECT * FROM student WHERE Name LIKE 'A%';
```

## 7.0 Database (Kolej Hostel Registration System)

After designing the database structure and preparing the necessary SQL statements, the next step is to implement the Kolej Hostel Registration System using XAMPP. XAMPP is a popular local web server environment that includes MySQL (MariaDB), Apache, and PHP, making it ideal for developing and testing database-driven applications.

In this section, we demonstrate how the database is created and managed within the XAMPP environment using phpMyAdmin, a graphical user interface for MySQL. The process involves setting up the database, executing the SQL scripts to create tables and insert data, and verifying the relationships and data integrity. This hands-on implementation allows the system to be tested in a controlled, offline environment before any deployment, and helps ensure that the database functions as expected in terms of structure, relationships, and data handling.

All Table

| | Table | Action | Rows | Type | Collation | Size | Overhead |
|---|---|---|---|---|---|---|---|
| ☐ | admin | Browse Structure Search Insert Empty Drop | 5 | InnoDB | utf8mb4_general_ci | 16.0 KiB | - |
| ☐ | hostel | Browse Structure Search Insert Empty Drop | 5 | InnoDB | utf8mb4_general_ci | 16.0 KiB | - |
| ☐ | payment | Browse Structure Search Insert Empty Drop | 5 | InnoDB | utf8mb4_general_ci | 48.0 KiB | - |
| ☐ | room | Browse Structure Search Insert Empty Drop | 5 | InnoDB | utf8mb4_general_ci | 32.0 KiB | - |
| ☐ | roomallocation | Browse Structure Search Insert Empty Drop | 5 | InnoDB | utf8mb4_general_ci | 80.0 KiB | - |
| ☐ | student | Browse Structure Search Insert Empty Drop | 5 | InnoDB | utf8mb4_general_ci | 16.0 KiB | - |

Table Admin

| | AdminId | Name | PhoneNumber | Email |
|---|---|---|---|---|
| ☐ Edit Copy Delete | A001 | LEE WEI MING | 012-2222221 | weiming.lee@hostel.com |
| ☐ Edit Copy Delete | A002 | MUHAMMAD HAZIQ BIN AZMI | 012-2222222 | haziq.azmi@hostel.com |
| ☐ Edit Copy Delete | A003 | NORAINA BINTI ZAKARIA | 012-2222223 | noraina@hostel.com |
| ☐ Edit Copy Delete | A004 | KAVITHA A/P SELVAN | 012-2222224 | kavitha.s@hostel.com |
| ☐ Edit Copy Delete | A005 | SITI NUR BALQIS BINTI MOHD | 012-2222225 | balqis.mohd@hostel.com |

Table Hostel

| | HostelId | HostelName | Location | GenderAllow |
|---|---|---|---|---|
| ☐ Edit Copy Delete | H001 | Kolej Khar | Jalan Khar 1 | Male |
| ☐ Edit Copy Delete | H002 | Kolej Ksas | Jalan Ksas 1 | Male |
| ☐ Edit Copy Delete | H003 | Kolej Kuo | Jalan Kuo 1 | Male |
| ☐ Edit Copy Delete | H004 | Kolej Kab | Jalan Kab 1 | Female |
| ☐ Edit Copy Delete | H005 | Kolej Ksjas | Jalan Ksjas 1 | Female |

Table Payment

| | PaymentId | StudentId | AdminId | PaymentDate | Status | ProofOfPayment | Amount |
|---|---|---|---|---|---|---|---|
| Edit Copy Delete | P001 | D20231106001 | A001 | 2025-06-01 | Approved | proof_p001.jpg | 506.00 |
| Edit Copy Delete | P002 | D20231106002 | A001 | 2025-06-01 | Approved | proof_p002.jpg | 506.00 |
| Edit Copy Delete | P003 | D20231106003 | A001 | 2025-06-01 | Rejected | proof_p003.jpg | 506.00 |
| Edit Copy Delete | P004 | D20231106004 | A003 | 2025-06-01 | Pending | proof_p004.jpg | 506.00 |
| Edit Copy Delete | P005 | D20231106005 | A004 | 2025-06-01 | Pending | proof_p005.jpg | 506.00 |

Table Room

| | RoomId | HostelId | RoomNumber | Capacity | RaceRestriction | Status |
|---|---|---|---|---|---|---|
| Edit Copy Delete | R001 | H001 | 001 | 2 | Chinese | Available |
| Edit Copy Delete | R002 | H001 | 002 | 2 | Chinese | Available |
| Edit Copy Delete | R003 | H001 | 101 | 2 | Malay | Available |
| Edit Copy Delete | R004 | H001 | 102 | 2 | Indian | Available |
| Edit Copy Delete | R005 | H001 | 201 | 2 | Malay | Available |

Table Room Allocation

| | AllocationId | StudentId | PaymentId | AdminId | RoomId | Status | SubmitDate |
|---|---|---|---|---|---|---|---|
| Edit Copy Delete | RA001 | D20231106001 | P001 | A001 | R001 | PENDING | 2025-06-01 |
| Edit Copy Delete | RA002 | D20231106002 | P002 | A001 | R001 | APPROVED | 2025-06-01 |
| Edit Copy Delete | RA003 | D20231106003 | P003 | A001 | R003 | APPROVED | 2025-06-01 |
| Edit Copy Delete | RA004 | D20231106004 | P004 | A003 | R004 | APPROVED | 2025-06-01 |
| Edit Copy Delete | RA005 | D20231106005 | P005 | A004 | R005 | PENDING | 2025-06-01 |

Table Student

| | StudentId | Name | Gender | Race | PhoneNumber | YearOfStudy | Faculty |
|---|---|---|---|---|---|---|---|
| Edit Copy Delete | D20231106001 | AHMAD FAIZ BIN ROSLAN | Male | Malay | 011-11111111 | 2 | FKMT |
| Edit Copy Delete | D20231106002 | TAN WEI LING | Female | Chinese | 011-11111112 | 1 | FPE |
| Edit Copy Delete | D20231106003 | ARUN KUMAR A/L RAJ | Male | Indian | 011-11111113 | 2 | FKMT |
| Edit Copy Delete | D20231106004 | NURUL AIN BINTI ZAINAL | Female | Malay | 011-11111114 | 4 | FSK |
| Edit Copy Delete | D20231106005 | LIM KAI WEN | Male | Chinese | 011-11111115 | 1 | FBK |

Designer View

The Designer View illustrates the relational structure of the Kolej Hostel Registration System database as implemented in phpMyAdmin. It visually represents the relationships between the six main tables: student, admin, payment, hostel, room, and roomallocation.

Each table is connected through primary and foreign key constraints to enforce referential integrity and ensure consistency across the system. For example, the payment table references both student and admin, while the roomallocation table acts as a central linkage between student applications, room assignments, payments, and administrative approval. Similarly, the room table is connected to the hostel table to reflect the location and type of accommodation.

This visual layout helps developers and administrators understand the overall schema structure, the flow of data, and how different entities in the hostel registration process are interrelated. It serves as a foundation for maintaining, extending, and debugging the system effectively.