**Temasek Polytechnic**

**School of Informatics and IT**

**Diploma in Information Technology (IT)**

Software Design Specifications (SDS)

**Project Particulars**

|  |  |
| --- | --- |
| **Tutor** | Mel Goh |
| **Class** | P03 |
| **Project Title** | Delonix Regia Hotel Management System |

**Project Team’s Particulars**

|  |  |
| --- | --- |
| **Matric Number** | **Student Name** |
| 1403530G | Shannon Sim Jun Hao |
| 1405591G | Joycelyn Nge Jia Qi |
| 1401811C | Vivian Neo Wen Ting |

**1. DISTRIBUTION OF WORKLOAD**

|  |  |
| --- | --- |
| **Design** | **Members** |
| Architecture Design | Vivian |
| User Interface Design | Vivian |
| Program Design | Joycelyn |
| Database Design | Shannon |

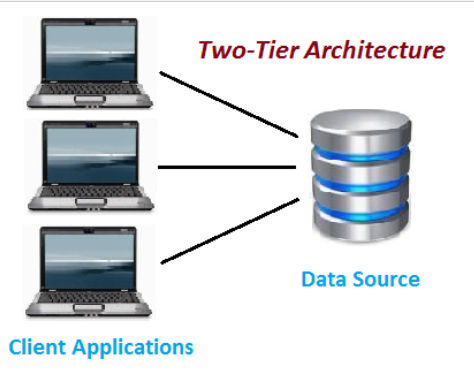
# **2. ARCHITECTURE DESIGN**

## **2-Tier Architecture**

A two-tier architecture is a software architecture in which a presentation layer or interface runs on a client, and a data layer or data structure gets stored on a server. Separating these two components into different locations represents a two-tier architecture, as opposed to a single-tier architecture.

2-Tier Architectures supply a basic network between a client and a server.

For example, the basic web model is a 2-Tier Architecture. A web browser makes a request from a web server, which then processes the request and returns the desired response, in this case, web pages. This approach improves scalability and divides the user interface from the data layers. However, it does not divide application layers so they can be utilized separately. This makes them difficult to update and not specialized. The entire application must be updated because the layers are not separated.



**Figure 1 - 2-Tier Architecture**

**3-Tier Architecture**



**Figure 2 - 3-Tier Architecture**

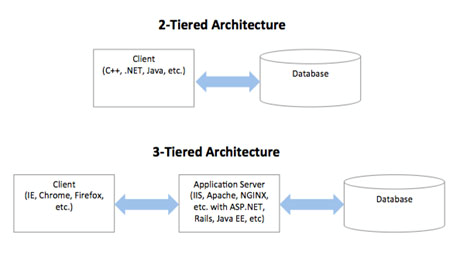
A three-tier architecture is a client-server architecture in which the functional process logic, data access, computer data storage and user interface are developed and maintained as independent modules on separate platforms. Three-tier architecture is a software design pattern and a well-established software architecture.

Three-tier architecture allows any one of the three tiers to be upgraded or replaced independently. The user interface is implemented on a desktop PC and uses a standard graphical user interface with different modules running on the application server. The relational database management system on the database server contains the computer data storage logic. The middle tiers are usually multi-tiered.

The three tiers in a three-tier architecture are:

1. Presentation Tier: Occupies the top level and displays information related to services available on a website. This tier communicates with other tiers by sending results to the browser and other tiers in the network.
2. Application Tier: Also called the middle tier, logic tier, business logic or logic tier, this tier is pulled from the presentation tier. It controls application functionality by performing detailed processing.
3. Data Tier: Houses database servers where information is stored and retrieved. Data in this tier is kept independent of application servers or business logic.

## **Comparison between 2-Tier Architecture and 3-Tier Architecture**



|  |  |  |
| --- | --- | --- |
| **Architecture** | **Advantages** | **Disadvantages** |
| 2-Tier | * Less likely to lose data as system and database are independent from each other * Easier to make future upgrades like expanding database capacity * Communication is faster * Easy to maintain | * Connections to database server are very expensive * Only connects a few number of users to a server * Expensive to deploy HMS system * Low security * Intranet application * Low level of processing requirements |
| 3-Tier | * Limit communications from web server to just the system servers * Application servers can be deployed on many machines * Database does not require any connection from every client * Reusable * Improved data integrity * Improved security * Reduced distribution * Improved availability * High level of processing requirements * Connects large number of people to a server * High degree of flexibility in deployment platform and configuration | * More complex * Point of communication are doubled * Difficult to set up and maintain |

**Conclusion**

Based on the information above, we have decided to propose deploying the HMS system using the 3-Tier Architecture. As seen above, the advantages of 3-Tier Architecture outweighs that of the 2-Tier Architecture.

The 3-Tier Architecture is more scalable as it allows the application servers to be deployed on many machines. The database also do not require a connection from every client. Since it is more scalable, it provides a flexible, dynamic back-end environment that can be adjusted as the needs and requirements changes.

The 3-Tier Architecture offers better and improved data security as the clients do not have direct access to the database. This is because the database is stored away from the HMS system preventing the database from being affected if any kind of virus has been attacked on the HMS system. It also offers better availability as the mission critical applications can make use of the redundant applications servers thus being able to recover from network of server failures.

# **3. USER INTERFACE (UI) DESIGN**

**Consistency**

The Navigation bar and background are the same throughout by creating a master page.

If UI controls are consistent, the user are able to navigate to their desired page easily.

**Prevention of errors**

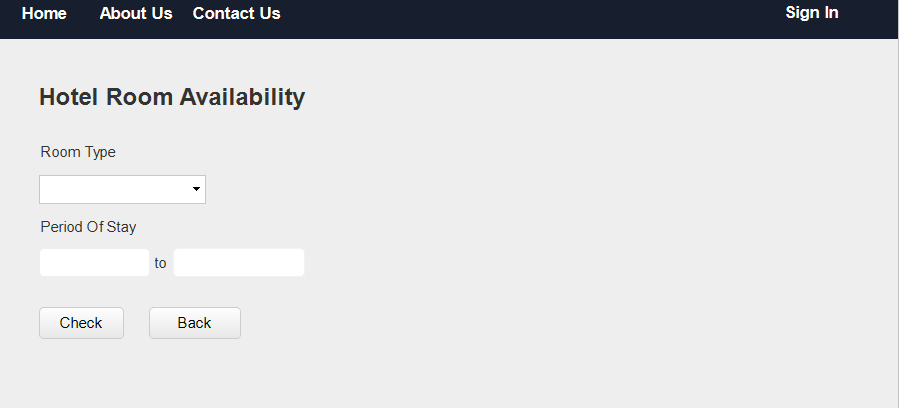
Users might make errors in inputting the wrong details or the wrong type of details format. The label will display an error message to the user instead of showing an error page with all the coding and this will improves user satisfaction and simplify the use of our User interface.

**User Friendly Interface**

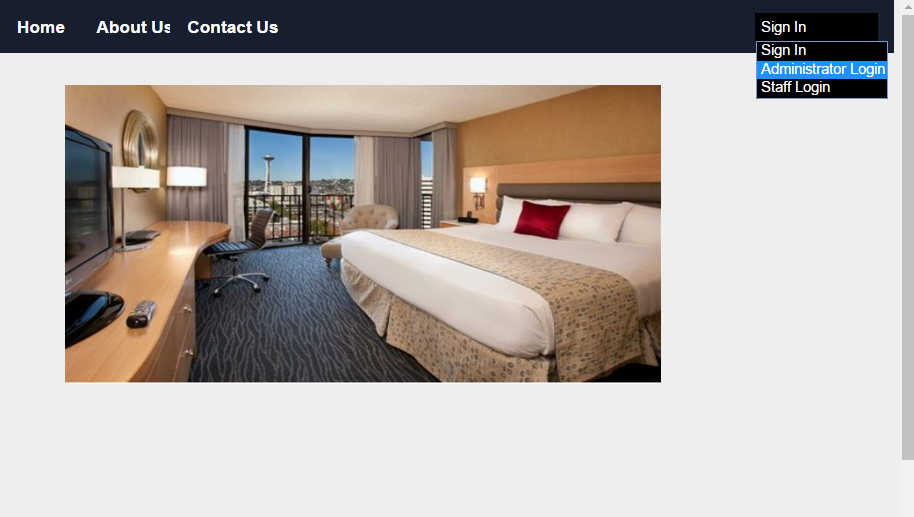
Website is placed and organized in textboxes for data entry and dropdown for date. It is easy for user to navigate to different web pages as the website have the same menu header.

Below are the screen shots of some of our web page’s User Interface.

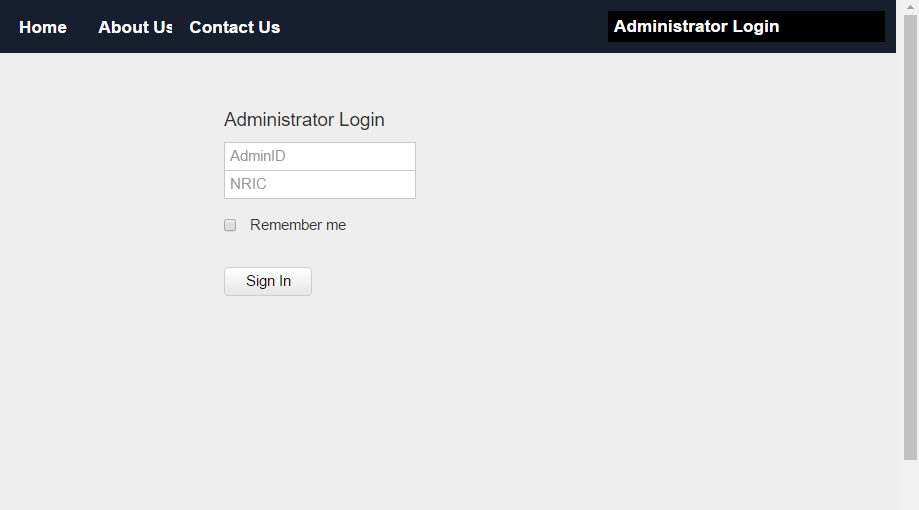
**Check Room Availability UI**

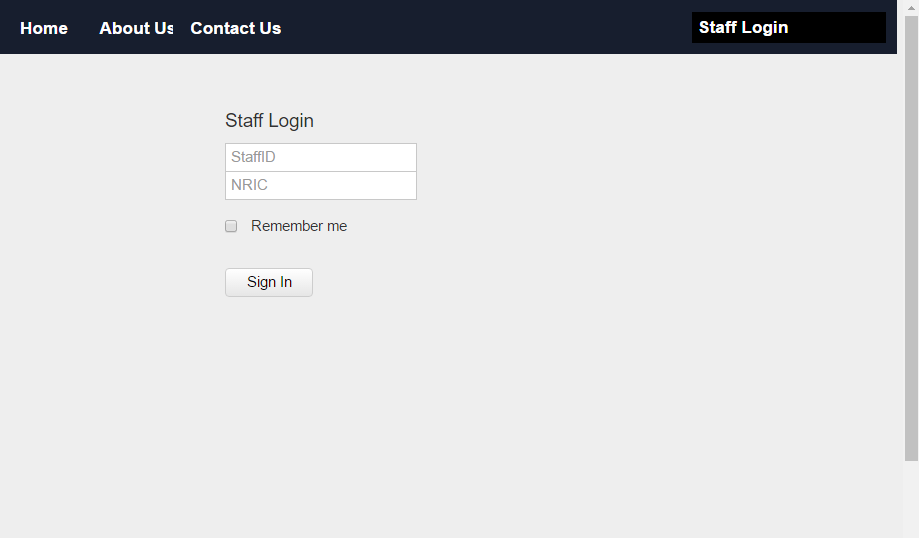


**Home Page UI**



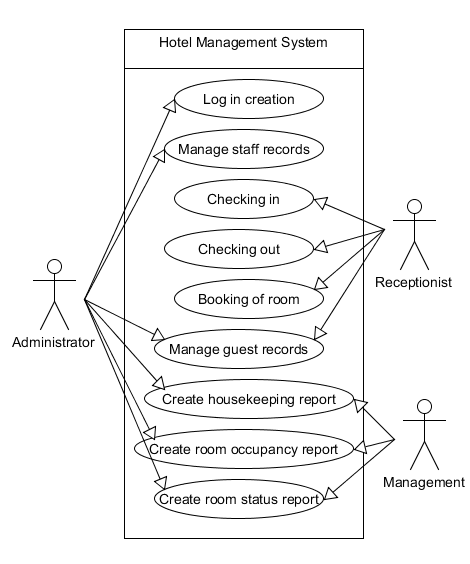
**Administrator & Staff Login UI**





**4. PROGRAM DESIGN**

**4.1 Use Case Diagram**



**4.2 Use Cases**

**Use Case 1: Log in creation**

**Brief description:** This use case allows the hotel administrator to register new accounts for new staffs

**Actors:** Hotel administrator

**Flow of events:**

**Main flow:**

1. The use case starts when there are new staffs in the hotel
2. The administrator will key in the staff’s personal details like their name, mobile no and address etc. and create an account for the new staff
3. The staff will choose a set of unique StaffID and password that will be used to access the system
4. The administrator will grant certain access of the system to the staff depending on their position in the hotel

**Alternative Flow:**

**Use Case 2: Manage staff records**

**Brief description:** This use case allows the hotel administrator to keep the latest record of the staffs

**Actors:** Hotel Administrator

**Flow of events:**

**Main flow:**

1. This use case starts when the staff in the hotel wants to update their personal particulars
2. The administrator will log in to the system
3. The administrator will click on the “update particulars” button
4. The administrator will select the particular staff that they wish to update
5. The system will direct the admin to the update form
6. After keying in the data, the admin will click on the submit button
7. The system will process the new data and save it into the database

**Alternative Flow 1: Delete staff records**

1. The alternative flow starts when the staff resigns or is being fired from the hotel
2. The administrator will log in to the system
3. The administrator will select the particular staff that they wish to delete
4. The administrator click on the delete button
5. The system processes the action and remove the staff from the database

**Use Case 3: Manage guest records**

**Brief description:** This use case allows the administrator to create, update, edit and delete the guest records

**Actors:** Administrator, receptionist

**Flow of events:**

**Main Flow:**

1. This use case starts when there are guests who are new to the hotel that wants to make a booking
2. The receptionist that is attending the customer will first log in to the hotel management system
3. The receptionist will click on the “new guest” button
4. The guest will provide their personal details like their name, mobile no and address etc
5. After keying the data, the receptionist will click on the “save” button
6. The system will process the data and save to the Guest records system

**Alternative Flow 1: Edit guest records**

1. This alternative flow starts from Step 4 of the main flow, the guest will provide the updated version of their personal particulars to the receptionist
2. After keying in the data, the receptionist will click on the “update” button
3. The system will process the data and save to the Guest records system

**Alternative Flow 2: Delete guest records**

1. This alternative flow starts when the management wants to delete the guest records
2. The administrator will log in to the guest records system
3. The admin will select the particular data that they wish to delete and click on “Delete”
4. The system will process the data and remove the guest from the database

**Use case 4: Create housekeeping report**

**Brief description:** This use case allows the hotel to generate housekeeping schedule on a monthly basis

**Actor:** Management and administrator

**Flow of events**

**Main Flow:**

1. This use case starts when the housekeeping report needs to be generated
2. The management will log in to the system and check for the working schedules of the staffs
3. The management will allocate different duties to the Housekeeping staffs according to their working days
4. The staffs are being allocated into four different segments: general maintenance, room maintenance, estate maintenance and security maintenance

**Use case 5: Create Room Occupancy report**

**Brief description:** This use case generates statistic reports of the room occupancy

**Actor:** Management and administrator

**Flow of events:**

**Main flow:**

1. This use case starts when the management wants to track the stats of all the occupants in the room
2. The management will log in to the Booking system and extract the data from the check in page
3. After extracting, the management will put the data in an excel sheet.

**Use case 6: Create room status report**

**Brief description:** This use case lists all the rooms the hotel has and the respective room status

**Actor:** Management and administrator

**Flow of Events:**

**Main Flow:**

1. The use case starts when the management needs to track the availability of the room
2. The management will log in to the system
3. The administrator will have to check the system for the status of all the different rooms and put it in an excel sheet

**Use case 7: Booking of Room**

**Brief description:** This use case allows the administrator to add booking

**Actor:** Administrator, Receptionist

**Flow of Events:**

**Main Flow:**

1. This use case starts when the guests wants to make a booking for the hotel
2. The receptionist will first log in to the hotel management system
3. The receptionist will click on the “Book hotel” button
4. It will direct the receptionist to the booking form
5. The guest will have to provide details required to make the booking like the date of the stay, number of adult guests staying in and credit card detail etc
6. After entering the personal details into the form, the receptionist will click “submit”
7. The system will then show the availability of the room with the requirements provided
8. The system will change the status of the room to booked

**Alternative Flow 1: Fully Booked**

**Room not available**

1. This alternative flow will start from step 7 of the main flow
2. The system will display a “Fully Booked” message
3. The guest can either choose to submit another date or cancel the booking process

**Use case 8: Checking out**

**Brief description:** This use case allows the receptionist and administrator to check out the hotel guests.

**Actor:** Receptionist, Administrator

**Flow of Events:**

**Main Flow:**

1. This use case starts when the hotel guests wants to check out
2. The receptionist will log in to the hotel management system
3. The receptionist will click the “check out” button
4. The receptionist will ask the guest for their details and check their records
5. The receptionist will generate a payment invoice which will calculate the number of days they stayed and any outstanding expenses
6. After the check out is complete, the system will change the room status to vacant

**Use case 9: Checking in**

**Brief description:** This use case allows the receptionist and administrator to check in the hotel guests

**Actor:** Receptionist, Administrator

**Flow of Events:**

**Main Flow:**

1. This use case starts when the hotel guests wants to check in
2. The receptionist will first log in to the hotel management system
3. The receptionist will click on the “check in” button
4. The receptionist will ask the guest for their details and verify their booking
5. After verify the booking, the system will update the room status from vacant to occupied after check in

**Alternative Flow 1: No booking**

1. This use case starts from step 4 of the check in
2. This system will display a “no booking made” message

# **4.3 Domain Model**

# 

# 

# 

# 

# 

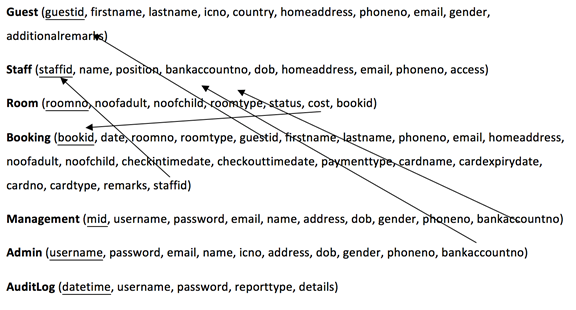
# 

# 

# 

# **5.DATABASE DESIGN**

**5.1 Database Schema**



**5.2 Assumptions**

The assumptions made for this database design are

* Customer are able to book multiple rooms at one time
* Database design is use to store all the users account details as well as booking details
* Each housekeeping staff are assigned to multiple rooms
* AuditLog are used for security purposes and backup storage are not part of the main procedure
* Mr Wang is able to add and remove accounts anytime

**5.3 Database Tables**

**Guest**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table Column Name** | **Table column Datatype** | **Table Column Null Option** | **Table Column Comment** | **Table Column is PK** | **Table column is FK** |
| **GuestId** | **Varchar(6)** | **No** |  | **Yes** | **Yes** |
| **FirstName** | **Char(25)** | **No** |  | **No** | **No** |
| **LastName** | **Char(25)** | **No** |  | **No** | **No** |
| **IcNo** | **Varchar(9)** | **No** |  | **No** | **No** |
| **Country** | **Char(50)** | **No** |  | **No** | **No** |
| **HomeAddress** | **Varchar(70)** | **No** |  | **No** | **No** |
| **PhoneNo** | **Int(8)** | **No** |  | **No** | **No** |
| **Email** | **Varchar(50)** | **No** |  | **No** | **No** |
| **Gender** | **Char(1)** | **No** |  | **No** | **No** |

**Staff**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table Column Name** | **Table column Datatype** | **Table Column Null Option** | **Table Column Comment** | **Table Column is PK** | **Table column is FK** |
| **StaffId** | **Varchar(6)** | **No** |  | **Yes** | **Yes** |
| **name** | **Char(50)** | **No** |  | **No** | **No** |
| **position** | **Char(30)** | **No** |  | **No** | **No** |
| **bankcAccNo** | **Varchar(12)** | **No** |  | **No** | **No** |
| **Dob** | **Varchar(10)** | **No** |  | **No** | **No** |
| **homeAddress** | **Varchar(70)** | **No** |  | **No** | **No** |
| **Email** | **Varchar(50)** | **No** |  | **No** | **No** |
| **phoneNo** | **Int(8)** | **No** |  | **No** | **No** |

**Room**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table Column Name** | **Table column Datatype** | **Table Column Null Option** | **Table Column Comment** | **Table Column is PK** | **Table column is FK** |
| **roomNo** | **Varchar(50)** | **No** |  | **Yes** | **No** |
| **noofAdult** | **int** | **No** |  | **No** | **No** |
| **noofChild** | **int** | **No** |  | **No** | **No** |
| **roomType** | **Varchar(50)** | **No** |  | **No** | **No** |
| **status** | **Varchar(50)** | **No** |  | **No** | **No** |
| **cost** | **Varchar(50)** | **No** |  | **No** | **No** |
| **bookId** | **Varchar(50)** | **No** |  | **No** | **Yes** |

**Booking**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table Column Name** | **Table column Datatype** | **Table Column Null Option** | **Table Column Comment** | **Table Column is PK** | **Table column is FK** |
| **bookId** | **Varchar(50)** | **No** |  | **Yes** | **Yes** |
| **date** | **date** | **No** |  | **No** | **No** |
| **roomNo** | **int** | **No** |  | **No** | **No** |
| **roomType** | **Varchar(50)** | **No** |  | **No** | **No** |
| **guestId** | **Varchar(50)** | **No** |  | **No** | **No** |
| **firstName** | **Varchar(50)** | **No** |  | **No** | **No** |
| **lastName** | **Varchar(50)** | **No** |  | **No** | **No** |
| **phoneNo** | **Varchar(50)** | **No** |  | **No** | **No** |
| **email** | **Varchar(50)** | **No** |  | **No** | **No** |
| **homeAddress** | **Varchar(50)** | **No** |  | **No** | **No** |
| **noofAdult** | **int** | **No** |  | **No** | **No** |
| **noofChild** | **int** | **No** |  | **No** | **No** |
| **checkinTimedate** | **datetime** | **No** |  | **No** | **No** |
| **checkoutTimedate** | **datetime** | **No** |  | **No** | **No** |
| **paymentType** | **Varchar(50)** | **No** |  | **No** | **No** |
| **cardName** | **Varchar(50)** | **No** |  | **No** | **No** |
| **cardExpirydate** | **date** | **No** |  | **No** | **No** |
| **cardNo** | **int** | **No** |  | **No** | **No** |
| **cardType** | **Varchar(50)** | **No** |  | **No** | **No** |
| **remarks** | **Varchar(50)** | **No** |  | **No** | **No** |
| **staffId** | **Varchar(50)** | **No** |  | **no** | **Yes** |

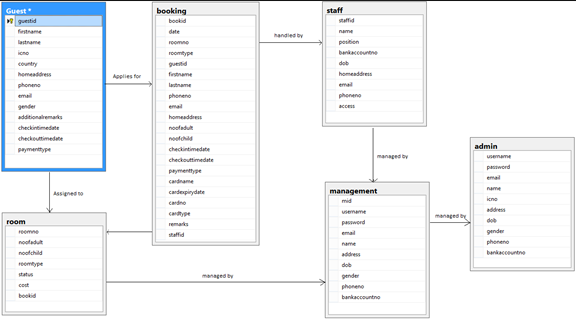
**Management**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table Column Name** | **Table column Datatype** | **Table Column Null Option** | **Table Column Comment** | **Table Column is PK** | **Table column is FK** |
| **mId** | **Varchar(50)** | **No** |  | **Yes** | **Yes** |
| **username** | **Varchar(50)** | **No** |  | **No** | **No** |
| **password** | **Varchar(50)** | **No** |  | **No** | **No** |
| **email** | **Varchar(50)** | **No** |  | **No** | **No** |
| **name** | **Varchar(50)** | **No** |  | **No** | **No** |
| **address** | **Varchar(50)** | **No** |  | **No** | **No** |
| **dob** | **Varchar(50)** | **No** |  | **No** | **No** |
| **gender** | **Varchar(50)** | **No** |  | **No** | **No** |
| **phoneNo** | **int** | **No** |  | **No** | **No** |
| **bankAccNo** | **Varchar(50)** | **No** |  | **No** | **No** |

**Admin**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table Column Name** | **Table column Datatype** | **Table Column Null Option** | **Table Column Comment** | **Table Column is PK** | **Table column is FK** |
| **username** | **Varchar(50)** | **No** |  | **Yes** | **Yes** |
| **password** | **Varchar(50)** | **No** |  | **No** | **No** |
| **email** | **Varchar(50)** | **No** |  | **No** | **No** |
| **name** | **Varchar(50)** | **No** |  | **No** | **No** |
| **icNo** | **Varchar(50)** | **No** |  | **No** | **No** |
| **address** | **Varchar(50)** | **No** |  | **No** | **No** |
| **dob** | **Varchar(50)** | **No** |  | **No** | **No** |
| **gender** | **Varchar(50)** | **No** |  | **No** | **No** |
| **phoneNo** | **int** | **No** |  | **No** | **No** |
| **bankAccNo** | **Varchar(50)** | **No** |  | **No** | **No** |

**5.4 Entity Relationship Diagram (ERD)**



**References**

**Vivian Neo - 1401811C**

<https://www.techopedia.com/definition/467/two-tier-architecture>

<https://www.techopedia.com/definition/24649/three-tier-architecture>