**DAMG 6210 – Sec 03 – Project 2 – Group 6**

# **Team members:**

|  |  |
| --- | --- |
| Simran Bhatia | 1097172 |
| Netra Awate | 1529534 |
| Ashwin Krishnan Meialagan | 2924384 |
| Prasanth Subburaj | 2925310 |
| Pavan Kumar Kora | 2924046 |
| Kousalyaa Senthilvel | 2968598 |

# **Project Topic:**

Database Design for Booking Rental Co-Working Spaces

# **Problem Statement:**

Budding startups and freelancers often forgo spending large amounts of money on an office setup in order to establish a designated workplace. However, working remotely comes with its own pains of blurred boundaries, lost concentration and myriad distractions. Using temporary office spaces for a few hours is the perfect solution to their dilemma.

1. It’s inefficient and time-consuming to log onto different platforms to look at what co-working spaces they have available.
2. It is frustrating to manage different login and credentials for each platform.
3. Overbooking and conflicting bookings can cause issues and waste time.
4. To avoid inconsistency and double booking, a single platform is ideal.

# **Problem Solution:**

This database design enables users who are individual workers (customers) to find a co-working office space nearby and book it for a certain period defined by the customer. Office or buildings will be posted by their owners with the name of rental, available time, address, photo, rental rate, and available spots.

To create a database for an office space rental company that has the features listed below:

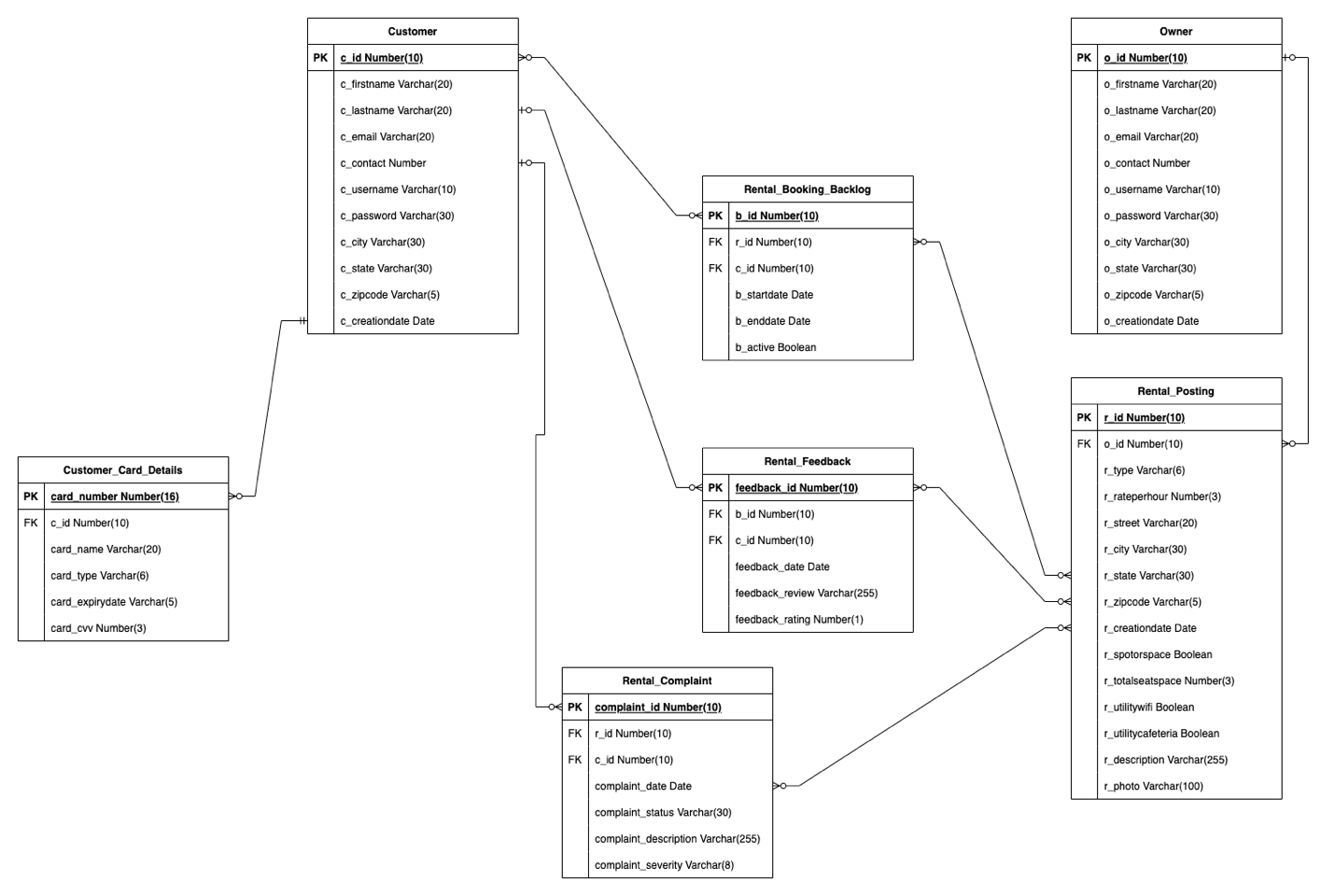
1. A Business User profile will be created with all the attributes from **OWNER**, and the rental posted by the owner with location details will be added to the **RENTAL** table.
2. A Customer profile will be created with all the attributes required for the **CUSTOMER** table, including but not limited to first name, last name, and contact details.
3. To ensure the transaction of money for booking a rental space, the customer must add credit card details as per **CUSTOMER\_CARD\_DETAILS.**
4. Customers can view rental listing information from the **RENTAL \_POSTING** table
5. Customers can book a space by filtering with specific attributes like location, date, or time.
6. Customers can edit their bookings from their rental history.
7. Customers can cancel their bookings from their rental history.
8. Customers can add a review or lodge a complaint about a rental once the booking duration is completed and the records will be stored in **RENTAL\_FEEDBACK** or **RENTAL\_COMPLAINT** tables respectively.

## **Logical Diagram:**

Diagram

Description automatically generated

## **Data Model ER Diagram:**



## **Facts:**

* All rentals are directly posted by owners without any broker.
* A customer can book a space only once for the given period of time.
* Customers won’t be allowed to book a rental after the space availability of rental is exceeded.
* Customers cannot book a space without credit card details, booking for a group of people will also require credit card details for all.

## **Assumptions:**

* The customer should book a rental space for a minimum of 2 hours.
* Customers cannot be owners to rent the space further.

## **Database Identification:**

Following are the entities with their attributes and description to build database design:

### **Customer:**

|  |  |  |  |
| --- | --- | --- | --- |
| Attributes | Datatypes | Constraints | Description |
| c\_id | Number(10) | Primary Key | Unique identifier for each customer |
| c\_firstname | Varchar (20) | Not Null | Customer’s first name |
| c\_lasttname | Varchar (20) | Not Null | Customer’s last name |
| c\_email | Varchar (20) | Not Null, Unique | Unique customer email |
| c\_contact | Number(10) | Not Null, Unique | Unique customer phone number |
| c\_username | Varchar (10) | Not Null, Unique | Unique customer username for login |
| c\_password | Varchar (30) | Not Null | Unique customer password for login |
| c\_city | Varchar (30) | Not Null | Customer’s city of residence |
| c\_state | Varchar (30) | Not Null | Customer’s state of residence |
| c\_zipcode | Varchar (5) | Not Null | Customer’s zip code of residence |
| c\_creationdate | DATE | Default SYSDATE | Default SYSDATE to view customer profile creation date |

### **Owner:**

|  |  |  |  |
| --- | --- | --- | --- |
| Attributes | Datatypes | Constraints | Description |
| o\_id | Number(10) | Primary Key | Unique identifier for each owner |
| o \_firstname | Varchar (20) | Not Null | Owner’s first name |
| o\_lasttname | Varchar (20) | Not Null | Owner’s last name |
| o\_email | Varchar (20) | Not Null, Unique | Unique owner email |
| o\_contact | Number | Not Null, Unique | Unique owner phone number |
| o\_username | Varchar (10) | Not Null, Unique | Unique owner username for login |
| o\_password | Varchar (30) | Not Null | Unique owner password for login |
| o\_city | Varchar (30) | Not Null | Owner’s city of residence |
| o\_state | Varchar (30) | Not Null | Owner’s state of residence |
| o\_zipcode | Varchar (5) | Not Null | Owner’s zip code of residence |
| o\_creationdate | DATE | Default SYSDATE | Default SYSDATE to view owner profile creation date |

### **Rental\_Posting:**

|  |  |  |  |
| --- | --- | --- | --- |
| Attributes | Datatypes | Constraints | Description |
| r\_id | Number(10) | Primary Key | Unique identifier for each rental |
| o \_id | Number (10) | Foreign Key | Owner identifier o\_id from OWNER table to map with rental posting table |
| r\_type | Varchar (6) | Not Null | Rental types : Office, Café, Studio |
| r\_rateperhour | Number(3) | Not Null | Rental’s hourly rate per entity |
| r\_street | Varchar (20) | Not Null, Unique | Rental’s street address |
| r\_city | Varchar (30) | Not Null | Rental’s city address |
| r\_state | Varchar (30) | Not Null | Rental’s state address |
| r\_zipcode | Varchar (5) | Not Null | Rental’s zip code |
| r\_creationdate | DATE | Default SYSDATE | Default SYSDATE to view owner rental posting creation date |
| r\_spotorspace | Boolean | Not Null | 0 if single spots are available or 1 for the whole office |
| r\_totalseatspace | Number(3) | Not Null | Total number of seats available for customers to occupy when renting by spot, else 0 |
| r\_utilitywifi | Boolean | Not Null | Does the rental space provide wifi for the occupants? |
| r\_utilitycafeteria | Boolean | Not Null | Does the rental space have a cafeteria for the occupants? |
| r\_description | Varchar (255) | Not Null | Description posted by owner for the rental space |
| r\_photo | Varchar(100) |  | URL linking to photo of rental (optional) |

### **Rental\_Booking\_Backlog:**

|  |  |  |  |
| --- | --- | --- | --- |
| Attributes | Datatypes | Constraints | Description |
| b\_id | Number (10) | Primary Key | Booking history identifier b\_id for bookings by all customers |
| r\_id | Number (10) | Foreign Key | Rental unique identifier r\_id from RENTAL POSTING BACKLOG table to map with rental booking table |
| c\_id | Number (10) | Foreign Key | Customer unique identifier c\_id from CUSTOMER table to map with rental booking table |
| b\_startdate | Date | Not Null | Start datetime stamp of booking |
| b\_enddate | Date | Not Null | End datetime stamp of booking |
| b\_active | Boolean | Not Null | 0 if booking has been cancelled, 1 if active |

### **Customer\_Card\_Details:**

|  |  |  |  |
| --- | --- | --- | --- |
| Attributes | Datatypes | Constraints | Description |
| card\_number | Number (16) | Primary Key | 16 digit credit/debit card number of the customer |
| c\_id | Number (10) | Foreign Key | Customer unique identifier c\_id from CUSTOMER table to map with rental booking table |
| card\_name | Varchar (20) | Not Null | Name mentioned on the credit/debit card |
| card\_type | Varchar (6) | Not Null | “Credit” or “Debit” to specify whether or credit or debit card is being used |
| card\_expirydate | Varchar(5) | Not Null | Expiry month and year of provided card in “MM/YY” format |
| card\_cvv | Number (3) | Not Null | Card security CVV number |

### **Rental\_Complaint:**

|  |  |  |  |
| --- | --- | --- | --- |
| Attributes | Datatype | Constraints | Description |
| complaint\_id | Number (10) | Primary key | Each complaint by the customer will be given a unique ID complaint\_id |
| r\_id | Number (10) | Foreign key | Rental unique identifier r\_id from RENTAL POSTING BACKLOG table to map with rental booking table |
| c\_id | Number (10) | Foreign key | Customer unique identifier c\_id from CUSTOMER table to map with rental booking table |
| complaint\_date | date | Not null, Default SYSDATE | Complaint lodged date will be automatically mentioned as SYSDATE |
| complaint\_status | Varchar(30) | Not null | Complaint status will process in steps as: Created -> acknowledged by the owner -> problem resolved |
| complaint\_description | Varchar (255) | Not Null | In detail description of the issue faced by customer |
| complaint\_severity | Varchar (8) | Not Null | Customer can select the severity of the issue as: High, Moderate and Low to help owner resolve high severity issue first |

### **Rental\_Feedback:**

|  |  |  |  |
| --- | --- | --- | --- |
| Attributes | Datatype | Constraints | Description |
| feedback\_id | Number (10) | Primary Key | Each feedback by the customer will be given a unique ID |
| b\_id | Number (10) | Foreign key | Booking history identifier b\_id from RENTAL BOOKING BACKLOG table to map with rental booking |
| c\_id | Number (10) | Foreign key | Customer unique identifier c\_id from CUSTOMER table to map with rental booking |
| feedback\_date | date | Not null, Default SYSDATE | Feedback submitted date will be automatically mentioned as SYSDATE |
| feedback\_review | Varchar (255) | Not Null | In detail feedback of the experience at the rental the customer |
| feedback\_rating | Number (1) | Not Null | Rating can be given from range of 0 - 5 for each rental space |

# **Business Rules**

1. Database Management consists of 3 human/user-faced profiles: **ANONYMOUS**, **OWNER** and **CUSTOMER**
2. Database Management consists of 3 developer-faced profiles: **MANAGER**, **DEVELOPER** and **ADMIN**
3. All users should be able to login using their username and password
4. **CUSTOMER**s must be able to register and have their credentials recorded, including company name, username, email, password, payment information and home address
5. **OWNER**s must be able to register and have their credentials recorded, including username, email, password, payment information and home address
6. Only one profile can be created from one set of credentials
7. The **OWNER** is responsible for posting rental spaces with the required description and availability of utilities
8. **OWNER** can post multiple rental postings with different locations and attributes
9. **CUSTOMER** cannot be an **OWNER** to rent a space further
10. **OWNER** must be allowed to withdraw coworking space listings and make changes to some details such as name of rental, available time, rental rate, photo, capacity, whether they are renting n spots or whole office
11. **CUSTOMER** can book a space for a minimum of 2 hours and provide credit or debit card details to confirm the booking
12. **CUSTOMER** cannot book a space if a rental spacing exceeds its availability for a given period
13. **CUSTOMER** can cancel a booking any time before 1 hour of the rental space booking start time which initiate the refund process
14. **CUSTOMER** can provide **FEEDBACK** once for a property rental
15. **CUSTOMER** can raise a complaint which will be addressed by the **OWNER** and resolution status will be updated
16. A **CUSTOMER** can book a space for 1 person or a group of people as per the seat availability as mentioned in the rental space posting under attribute **r\_totalseatspace**
17. The **CUSTOMER** can log in to their account and view their active rental details from **RENTAL\_BACKLOG** table
18. **CUSTOMER** should be able to see their current reservation and previous booking history
19. **CUSTOMER** must be able to filter available listings on some specific criteria, like whether or not a rental space has internet available, available spots, etc
20. **CUSTOMER** must be allowed to leave reviews about the services
21. **CUSTOMER** must be allowed to report incidents if they encounter any
22. **CUSTOMER** should be able to sort listings

# **Views**

The following views will be included in the database model:

## **Rental\_Complaint\_View**

To allow a **CUSTOMER** to view only complaints they posted.

## **Rental\_Posting\_View**

To show only **OWNER**’s rental properties posted by that **OWNER**.

## **Rentals\_View**

To show a **CUSTOMER** all available listings starting from the current date.

## **Rental\_Booking\_Backlog\_View**

To show each **CUSTOMER** only their own booking log.

## **Customer\_View**

To show **MANAGER**s the list of customers without email, contact, password columns.

## **Owner\_View**

To show **MANAGER**s the list of owners without email, contact, password columns.

## **Customer\_Card\_View**

All roles except **CUSTOMER** should only be able to see c\_id, card\_name, and card\_type columns from the customer card details table. The customer should only be allowed to view the last four digits of their credit card number.

## **Rental\_Feedback\_View**

To show **OWNER**s only feedback posted about their own properties.

# **Dataflow Diagrams:**

## **Rental\_Complaint\_View ->**

Diagram

Description automatically generated

## **Rental\_Posting\_View ->**

Diagram

Description automatically generated

## **Rentals\_View ->**

Diagram

Description automatically generated

## **Rental\_Booking\_Backlog\_View ->**

Diagram

Description automatically generated

## **Customer\_View ->**

Diagram

Description automatically generated

## **Owner\_View ->**

Diagram

Description automatically generated

## **Customer\_Card\_View ->**

Diagram

Description automatically generated

## **Rental\_Feedback\_View ->**

Diagram

Description automatically generated

## **Transaction to Book a Listing ->**

Diagram

Description automatically generated

## **Transaction to Cancel a Booking ->**

Diagram

Description automatically generated

# **Security**

## **Profiles**

User-role profiles and developer-role profiles are both defined below. User-role profiles will be used by users of the application and database, whereas develop-role profiles will be used by developers working on creating, developing, and testing the application.

### **User-role Profiles**

1. Anonymous
2. Customer
3. Owner

#### **Anonymous Profile**

When a user is not logged in or their profile is not known, they will only have very basic permissions (only to be able to sign up and sign in as a registered user) and will be considered “Anonymous” users.

#### **Customer Profile**

Users who sign on to the application as **CUSTOMER** will have all privileges associated with a customer, such as viewing but not modifying apartment listings, making bookings, writing reviews etc.

#### **Owner Profile**

Business users who sign on to the application as rentees will be classified as an **OWNER**. They will not be able to make bookings or edit reviews/feedback except for adding comments, and they will be able to add and modify apartment listings.

### **Developer-role Profiles**

1. Analyst
2. Developer
3. Admin

#### **Analyst User**

An analyst will be allowed to view all the data except for confidential information such as credit card information of a customer. Analysts will not have edit privileges.

#### **Developer User**

Developers will be able to view and query all tables, except for confidential information such as credit card information of a customer. Developers will not have edit rights unless granted by the admin on case-by-case basis. Developers will not have “create” access to create tables.

#### **Admin User**

Admin will have all rights, as well as privileges to define rights for all other users and to create tables.

## **Table-wise Permissions**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Anonymous | Customer | Owner | Analyst | Developer | Admin |
| CUSTOMER TABLE | | | | | | |
| Read |  | ✔️ |  | ✔️ | ✔️ | ✔️ |
| Write | ✔️ |  |  |  | ✔️ | ✔️ |
| Update |  | ✔️ |  |  | ✔️ | ✔️ |
| Delete |  |  |  |  | ✔️ | ✔️ |
| Alter |  |  |  |  |  | ✔️ |
| Drop |  |  |  |  |  | ✔️ |
| OWNER TABLE | | | | | | |
| Read |  |  | ✔️ | ✔️ | ✔️ | ✔️ |
| Write | ✔️ |  |  |  | ✔️ | ✔️ |
| Update |  |  | ✔️ |  | ✔️ | ✔️ |
| Delete |  |  |  |  | ✔️ | ✔️ |
| Alter |  |  |  |  |  | ✔️ |
| Drop |  |  |  |  |  | ✔️ |
| RENTAL\_POSTING TABLE | | | | | | |
| Read |  | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ |
| Write |  |  | ✔️ |  | ✔️ | ✔️ |
| Update |  |  | ✔️ |  | ✔️ | ✔️ |
| Delete |  |  | ✔️ |  | ✔️ | ✔️ |
| Alter |  |  |  |  |  | ✔️ |
| Drop |  |  |  |  |  | ✔️ |
| RENTAL\_BOOKING\_BACKLOG TABLE | | | | | | |
| Read |  | ✔️\* | ✔️\*\* | ✔️ | ✔️ | ✔️ |
| Write |  | ✔️ |  |  | ✔️ | ✔️ |
| Update |  | ✔️ |  |  | ✔️ | ✔️ |
| Delete |  | ✔️ |  |  | ✔️ | ✔️ |
| Alter |  |  |  |  |  | ✔️ |
| Drop |  |  |  |  |  | ✔️ |
| CUSTOMER\_CARD\_DETAILS TABLE | | | | | | |
| Read |  |  |  |  |  | ✔️ |
| Write |  | ✔️ |  |  |  | ✔️ |
| Update |  | ✔️ |  |  |  | ✔️ |
| Delete |  | ✔️ |  |  |  | ✔️ |
| Alter |  |  |  |  |  | ✔️ |
| Drop |  |  |  |  |  | ✔️ |
| RENTAL\_COMPLAINT TABLE | | | | | | |
| Read |  | ✔️\* | ✔️\*\* | ✔️ | ✔️ | ✔️ |
| Write |  | ✔️ |  |  | ✔️ | ✔️ |
| Update |  |  | ✔️ |  | ✔️ | ✔️ |
| Delete |  | ✔️ |  |  | ✔️ | ✔️ |
| Alter |  |  |  |  |  | ✔️ |
| Drop |  |  |  |  |  | ✔️ |
| RENTAL\_FEEDBACK TABLE | | | | | | |
| Read |  | ✔️\* | ✔️\*\* | ✔️ | ✔️ | ✔️ |
| Write |  | ✔️ |  |  | ✔️ | ✔️ |
| Update |  | ✔️ |  |  | ✔️ | ✔️ |
| Delete |  | ✔️ |  |  | ✔️ | ✔️ |
| Alter |  |  |  |  |  | ✔️ |
| Drop |  |  |  |  |  | ✔️ |

\*Only values referencing that specific **CUSTOMER**

\*\*Only values referencing that specific **OWNER**