#### ****Database Project Reservation Hotel (PU SUITES)****

#### ****Name : Nabil Fadhlur Rahman (001202400169)****

**Aurelio Sami Rabbani (001202400218)**

#### ****s****

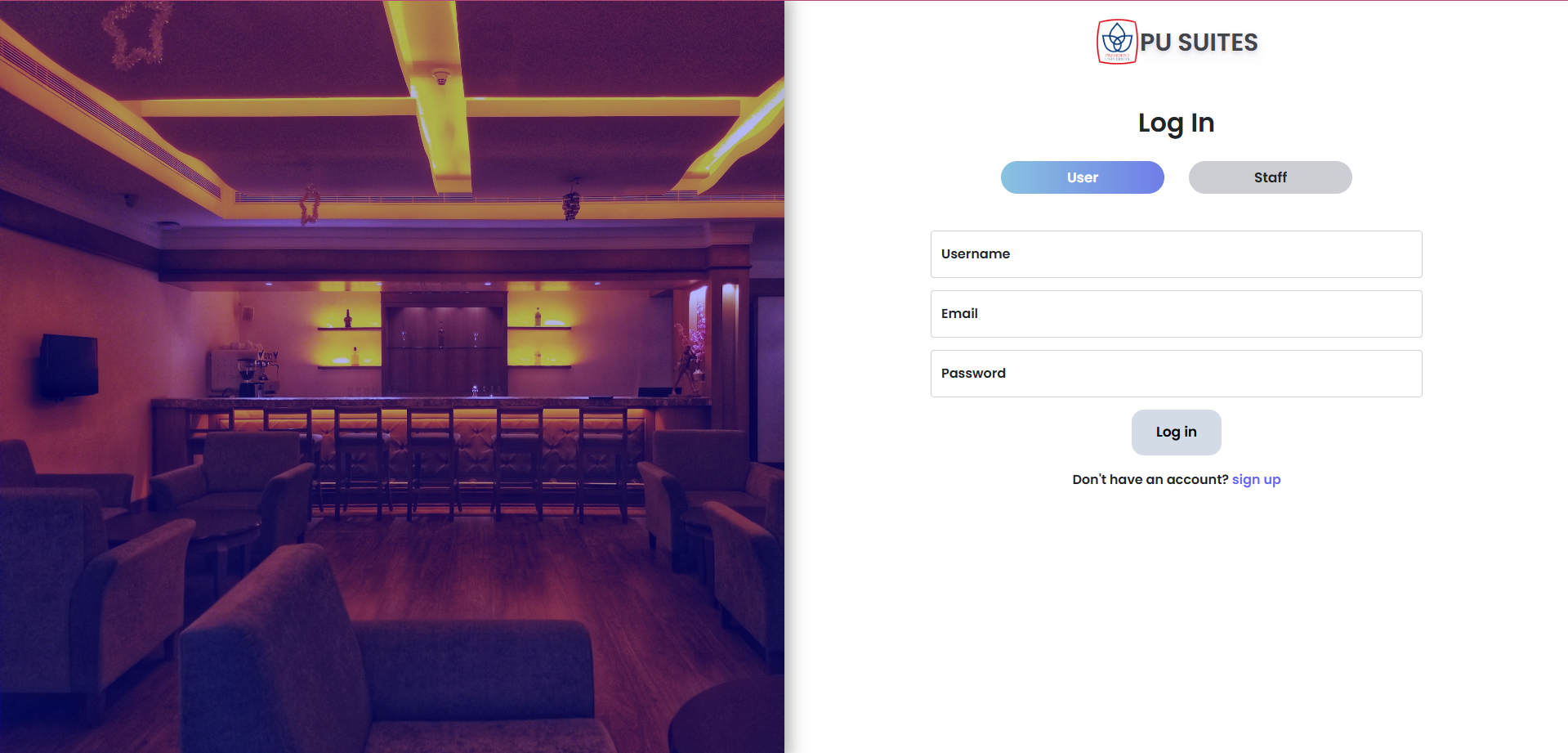
#### ****A.1 Overview****

The **Hotel Management System** aims to automate and streamline hotel operations, allowing customers to manage bookings, room availability, and payments. Employees can manage customer details, room assignments, and generate reports for better operational efficiency.

The system asks users to log in or register first. Upon registration, users fill out details like name, email, password, etc. After the registration is successful, they can log in and start using the system's features.

Upon logging in, customers are presented with options to book a room, view available rooms, and make payments. Employees can add rooms, update availability, and manage bookings.

The system allows for the creation of an admin account, which provides access to special features such as user monitoring, deletion, and database maintenance.



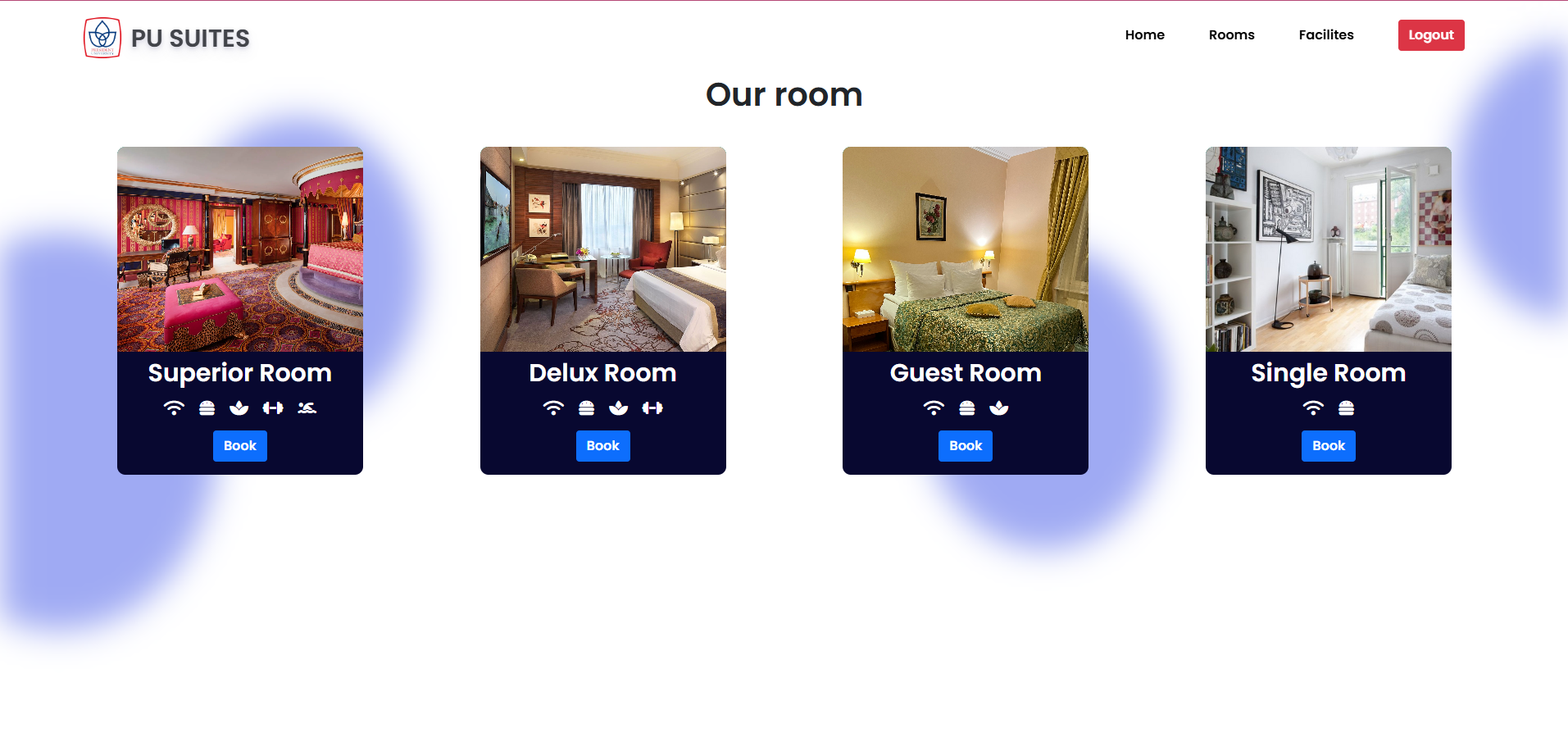
##### **Form A.1.1 (Sign-Up or Log-In Page)**

* The sign-up button opens a form asking for personal details.
* After submitting, users will be notified of successful registration or if the email/username is already taken.

##### 

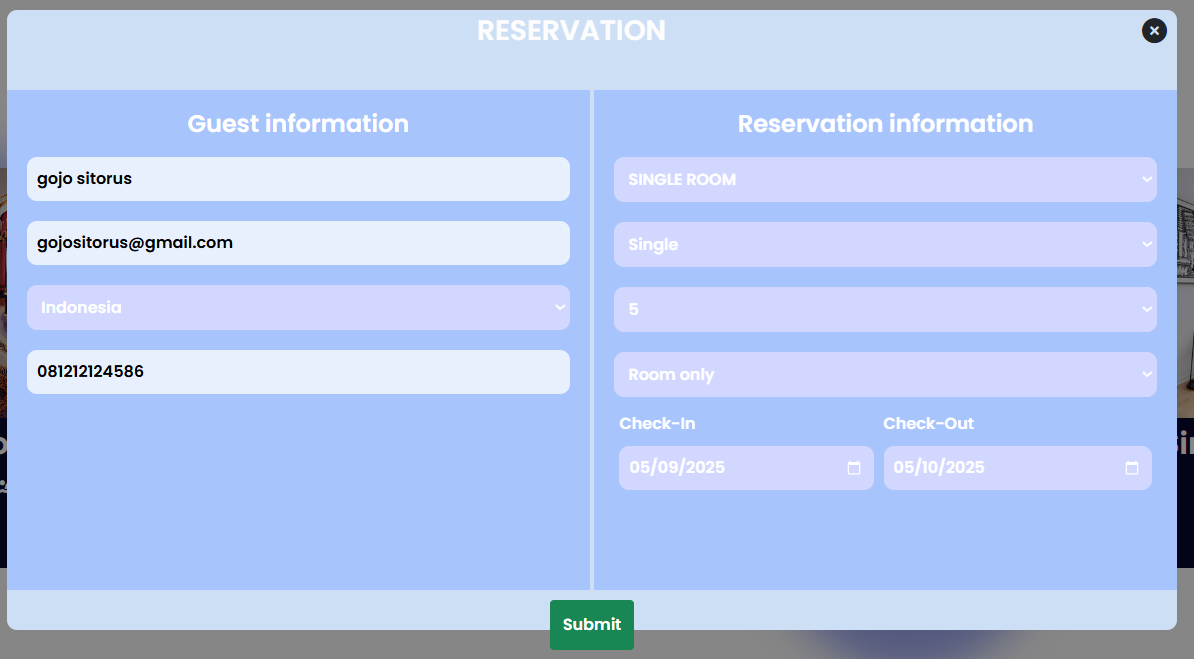
##### **Form A.1.2 (Sign-Up Dialog)**

* If email is available, the user proceeds to the next page, which confirms the success of account creation.



##### **Form A.1.3 (Booking Page and Search)**

* Customers can search available rooms based on their desired check-in and check-out dates.
* They can view a list of available rooms with their prices and details.

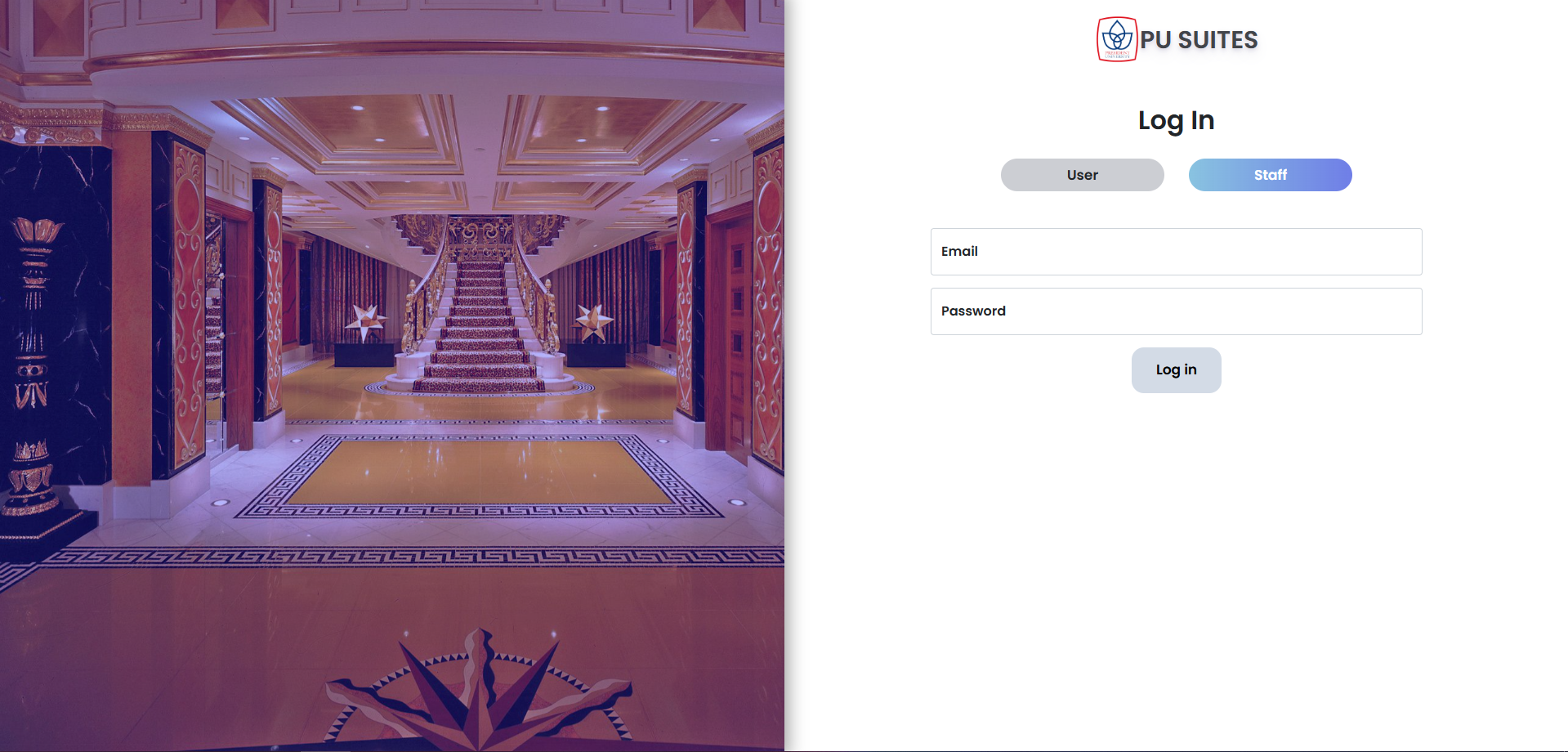


##### **Form A.1.4 (Booking Form)**

* Users select the room and complete the booking by providing necessary details.
* A payment form will appear upon booking confirmation.

##### **Form A.1.5 (Payment Form)**

* Users input payment details and finalize the transaction.
* The system tracks the payment status and updates the booking accordingly.



**Form A.1.6 (Admin Log-In Form)**

* The admin will be required to input their **email** and **password** that are already registered in the system.
* The **email** and **password** are used to verify the admin's identity.

#### ****A.2 Business Functions****

**Users should be able to:**

* **Make a Booking:** Select a room type, choose check-in and check-out dates, and complete the booking process by providing personal details and payment.
* **View Available Rooms:** Search for available rooms by specifying check-in and check-out dates and preferred room type.
* **View Profile:** View and update their profile, including name, email, password, and profile picture.
* **Manage Bookings:** View, modify, or cancel upcoming bookings based on availability and policies.
* **Rate and Review Rooms:** Rate their stay and leave a review for the rooms they have booked.
* **Receive Notifications:** Get notifications about booking confirmations, cancellations, or payment updates.
* **Edit Profile:** Update their profile picture, cover photo, personal bio, location, and website link.
* **Change Email and Username:** Change their email address and username, provided the new ones are available.
* **Change Password:** Update their password using the current one and confirm the new password.
* **Log Out:** Log out from their account to ensure their session is securely closed.

**Admins should be able to:**

* **Monitor Users:** Monitor user activity, including booking history and reviews, to ensure proper system usage.
* **Manage Users:** Create, update, or delete user accounts and change user roles (e.g., customer, admin).
* **View User Profiles:** Access and review detailed user profiles to manage their data and booking history.
* **Manage Room Listings:** Add, update, or remove room listings, including room types, availability, and pricing.
* **View Booking History:** View all users' booking history and the current status of each reservation.
* **Generate Reports:** Create reports regarding bookings, revenue, user activity, and overall system performance.
* **Handle Payments:** Oversee payment processes, confirm successful transactions, and address payment-related issues.
* **Update System Settings:** Modify system settings related to pricing, room types, policies, and general configurations.
* **Delete User Accounts:** Remove user accounts and all associated data from the system, including bookings and reviews.

### ****A.3 Data Requirements****

* **User Account Information** (email, username, name, and password)
* **User Profile Information** (profile picture, cover picture, biography, location, and website link)
* **User Bookings** (room type, check-in date, check-out date, total cost, and payment status)
* **User Feedback or Ratings** (rating score, review text, and review time)
* **User Payment Information** (payment method, amount, payment status, and payment time)
* **User Notifications** (user who triggers a notification, notified user, notification type, notification target, and time)
* **User Room Preferences** (room type, preferred amenities, price range, and location)
* **User Payment History** (payment ID, payment amount, booking ID, and payment time)
* **Admin Information** (admin’s name, admin’s email, and password)

### ****A.4 BUSINESS RULES****

1. The user should be able to create an account with a valid email address, username, full name, and password.
2. The user should be able to book a room by selecting the room type, check-in, and check-out dates, and completing the payment.
3. The user should be able to view their booking details, including the room type, dates, and total cost, on their profile page.
4. The user should be able to cancel a booking within a specified time frame before check-in.
5. The user should be able to update their profile information, including their name, email, password, and profile picture.
6. The user should be able to see available rooms based on their selected check-in and check-out dates.
7. The admin should be able to add, modify, or delete rooms from the system.
8. The admin should be able to manage user accounts, including creating, updating, or deleting user profiles.
9. The admin should be able to view all bookings and manage their status (confirmed, canceled, etc.).
10. The system should track the payment status of each booking, ensuring the payment is processed before confirming the reservation.

**B.1 OVERVIEW**

The project **PU SUITES** aims to create an online hotel management system. It enables users to search for available rooms, book rooms, manage their profiles, and handle reservations. Users can create an account by providing their name, email address, and password. The system allows users to make bookings by selecting a room type, check-in, and check-out dates, and completing the payment process. Users will also be able to modify their personal information and view past bookings. The system includes an admin panel for the management of users, rooms, and bookings. Admins can view and manage user activity, room availability, and perform administrative tasks like generating reports and deleting user accounts when necessary.

**B.2 REVISED BUSINESS RULES AND ASSUMPTIONS**

1. **User Account Creation**:  
   The user should be able to create an account by providing a valid email address, username, full name, and password. A unique identifier will be automatically generated for each user upon account creation.
2. **Room Booking**:  
   The user should be able to search for available rooms by entering check-in and check-out dates and selecting a preferred room type. After selecting a room, users can proceed with booking by providing personal details and payment information.
3. **Profile Management**:  
   The user should be able to update their profile by changing their name, email address, password, profile picture, cover photo, biography, location, and website link.
4. **View Booking Details**:  
   The user should be able to view their booking details on their profile page, including room type, check-in and check-out dates, total cost, and payment status.
5. **Modify and Cancel Booking**:  
   The user should be able to modify or cancel their booking before the check-in date, based on hotel policies and room availability.
6. **Admin Account Creation**:  
   Admins should be able to create their accounts by providing a valid admin name, email, and password. The admin name and email must be unique, and a unique identifier will be generated for each admin account.
7. **Admin Login**:  
   Admins should be able to log in to the admin panel using their registered admin name and password.
8. **Room Management**:  
   Admins should be able to add new rooms, update room details such as price and availability, and remove rooms that are no longer available.
9. **User Management**:  
   Admins should be able to create, update, and delete user accounts. They can also modify user roles (e.g., customer, employee, admin).
10. **Booking Monitoring**:  
    Admins should be able to monitor user bookings, including modifying booking statuses (e.g., confirmed, canceled) and viewing detailed booking history.
11. **Payment Handling**:  
    Admins should be able to monitor payment status for bookings and ensure successful payment processing before confirming the reservation.
12. **Generate Reports**:  
    Admins should be able to generate reports on user activity, booking statistics, room availability, and revenue generation.
13. **Notification System**:  
    Users should receive notifications regarding booking confirmations, cancellations, payment updates, and profile changes. Admins should also receive notifications for important actions such as user registrations and booking issues.
14. **Room Preferences**:  
    Users should be able to specify their preferences for room types, amenities, and price range when searching for available rooms.
15. **Room Availability**:  
    Users should only be able to book rooms that are available for the selected check-in and check-out dates.
16. **Feedback and Reviews**:  
    Users should be able to rate their stay and leave feedback for rooms they have booked. Reviews should be visible on the room’s page and help other users make informed decisions.
17. **Email and Username Changes**:  
    Users should be able to change their email address and username to another available one after logging into the system.
18. **Password Changes**:  
    Users should be able to change their password after confirming their old password.
19. **Admin Monitoring**:  
    Admins should be able to monitor user activity, including viewing booking details, user profiles, and overall system usage.
20. **Account Deletion**:  
    Admins should be able to delete user accounts and all associated data, including bookings and reviews, when necessary.
21. **System Security**:  
    The system should ensure secure password handling by hashing passwords before storage. Admins should have secure access to sensitive user data.
22. **Session Management**:  
    Users and admins should be automatically logged out after a period of inactivity to maintain security.
23. **Payment Method**:  
    Users should be able to pay for their bookings through various methods, such as credit cards, PayPal, or bank transfers.
24. **Booking Confirmation**:  
    Once payment is confirmed, users should receive a booking confirmation email with booking details.
25. **User Interaction Logging**:  
    The system should log every user interaction, such as bookings, cancellations, profile updates, and feedback, to ensure proper tracking and reporting.
26. **Admin Control of User Roles**:  
    Admins should have the ability to assign user roles, including the ability to grant or revoke admin privileges to other users.
27. **User Notifications**:  
    Users should be notified of important events, including booking updates, cancellations, and payment issues. Notifications should be displayed in reverse chronological order.
28. **Data Privacy and Compliance**:  
    The system should ensure that user data is protected and complies with relevant data privacy regulations (e.g., GDPR).
29. **Trends and Popular Rooms**:  
    Admins should be able to monitor trends, such as the most popular room types or booking periods, to optimize the availability and pricing of rooms.
30. **Room Reservation Limits**:  
    Users should be able to reserve only a certain number of rooms per booking based on system limitations and available inventory.

B.3. ERD (CONCEPTUAL DATA MODELLING)

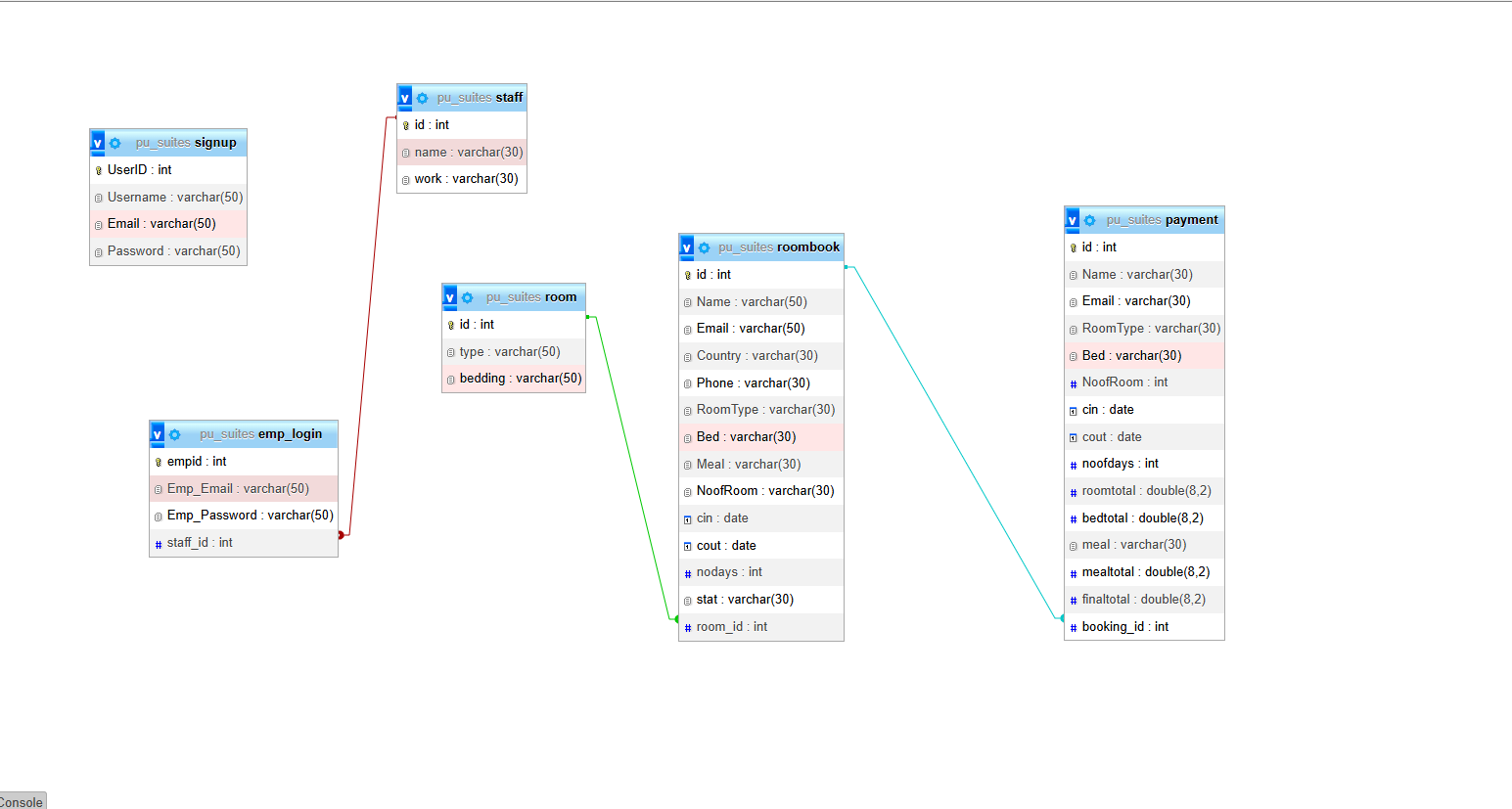


Image B.3.1 (Entity Relational Diagram PU SUITES)

### ****B.4 JUSTIFICATIONS OF ERD BASED ON BUSINESS RULES & ASSUMPTIONS (PU SUITES)****

#### **BUSINESS RULES RELATED TO ENTITY USERS**

**BR1: A user should be able to create an account with a username, name, email address, and password. On the creation of every user account, a unique identifier will be automatically generated to further ease the user in using the application.**

The part that declares ‘created with a username, name, email address, and password’ justifies the existence of **username**, **name**, **email**, and **password** attributes in the **Users** entity. These are fundamental attributes needed for user identification, authentication, and access control within the application. These attributes ensure each user has a unique account and the system can validate access to the platform.

The part that declares ‘a unique identifier will be automatically generated’ justifies the existence of **user\_id** as the primary key (PK) for the **Users** entity. The **user\_id** attribute is a unique identifier for each user, ensuring there is no duplication of user records. It facilitates the tracking of user interactions, bookings, and activities within the platform, providing a way to uniquely identify every user.

**BR29: Users should be able to edit their profile and input their personal information such as their profile image, cover image, biography (short description of themselves), location, and website.**

This rule justifies the inclusion of **profile\_picture**, **cover\_picture**, **bio**, **location**, and **website** attributes in the **Users** entity. These attributes allow users to personalize their profiles and provide relevant information about themselves. By giving users the ability to edit these attributes, the platform offers a more engaging and customizable experience.

**BR34: An admin should be able to claim the position of a PIC for users. While a user can only have one PIC.**

The rule justifies the inclusion of the **PIC** attribute in the **Users** entity. This attribute establishes the relationship between admins and users, allowing an admin to be assigned to a specific user (as a **PIC**). This ensures that every user is managed by one admin, fostering accountability and oversight. The **user\_id** serves as the primary key for the **Users** entity.

#### **BUSINESS RULES RELATED TO ENTITY ROOMS**

**BR3: Each room should have a unique identifier. The room information will include type, price, availability, description, and capacity.**

The **Rooms** entity justifies the inclusion of **room\_type**, **price**, **availability\_status**, **room\_description**, and **room\_capacity** attributes. These fields help define each room in the system, allowing users to filter and search for rooms based on these attributes. The unique **room\_id** identifier ensures each room is distinctly represented.

**BR9: Users can book a room for specific dates (check-in and check-out). The booking process will associate the user with the room they book.**

The **Bookings** entity justifies the relationship between **Users** and **Rooms** through **user\_id** and **room\_id** foreign keys. This relationship ensures that the system can track which user has booked which room for what time period. The **check\_in\_date** and **check\_out\_date** attributes allow the booking process to function correctly, ensuring the correct rooms are allocated to users for their specified stay dates.

#### **BUSINESS RULES RELATED TO ENTITY BOOKINGS**

**BR8: A user should be able to book a room by selecting a check-in and check-out date.**

The **Bookings** entity justifies the relationship between **Users** and **Rooms** through **check\_in\_date**, **check\_out\_date**, **total\_cost**, and **payment\_status** attributes. These attributes are essential for managing bookings, tracking payment statuses, and calculating the total cost of a room for the specified dates. This ensures that each booking is properly recorded and processed.

**BR16: A user can make only one booking per room for the same dates.**

This rule ensures that no user can book the same room for overlapping dates, preventing overbooking. The **Bookings** entity uses a composite primary key of **user\_id** and **room\_id**, ensuring that each booking is unique to a specific user and room combination. This prevents duplicate bookings and maintains room availability

#### **BUSINESS RULES RELATED TO ENTITY PAYMENTS**

**BR12: Each booking made by a user will be associated with a payment record. The user must complete a payment for the booking to be finalized.**

The **Payments** entity justifies the inclusion of **payment\_amount**, **payment\_method**, and **payment\_status** attributes. These attributes help track the payments associated with each booking. The **booking\_id** foreign key links payments to the corresponding bookings, ensuring that payments are correctly processed and linked to specific reservations.

**BR18: Payments should be linked to a user’s booking, and a payment can be made only once for a specific booking.**

This rule ensures that there is only one payment per booking, preventing issues of duplicate payments. The **Payments** entity uses **payment\_id** as the primary key (PK) to uniquely identify each payment and links the payment to the booking through the **booking\_id** foreign key. This ensures each payment record is accurately tied to its respective booking.

#### **BUSINESS RULES RELATED TO ENTITY ADMIN**

**BR32: Admins should be able to manage user accounts by registering as administrators and logging into the admin panel.**

The **Admin** entity justifies the inclusion of **admin\_name**, **admin\_email**, and **admin\_password** attributes, which are required for admin registration, login, and account management. The **admin\_id** primary key uniquely identifies each admin, providing access control and functionality for managing user data and platform operations.

**BR33: Admins will be able to monitor user activities, assign PIC roles, and remove users if needed.**

The **Admin** entity allows admins to manage users through the **PIC** attribute, which identifies which admin is responsible for each user. This functionality ensures that admins can monitor, assign responsibilities, and remove users when necessary, maintaining system integrity and user accountability.

#### **BUSINESS RULES RELATED TO ENTITY NOTIFICATIONS**

**BR14: Notifications should be sent to users regarding booking updates, new followers, and other relevant events.**

The **Notifications** entity justifies the inclusion of **notification\_type**, **notification\_message**, and **user\_id** attributes. These attributes allow the system to track and send notifications for various events such as booking updates, new followers, and other important actions. **user\_id** helps identify the recipient of each notification.

**BR13: Notifications should be ordered from latest to oldest, and users can mark them as read.**

The **Notifications** entity’s **created\_at** attribute allows notifications to be displayed in chronological order, helping users easily navigate and manage their notifications. The **count** attribute tracks whether a user has opened the notification or not, allowing them to manage unread notifications efficiently.

#### **BUSINESS RULES RELATED TO ENTITY REVIEWS**

**BR17: Users should be able to review rooms they have stayed in.**

The **Reviews** entity justifies the inclusion of **rating**, **review\_text**, and **created\_at** attributes. These attributes allow users to leave feedback on rooms they have stayed in, which can be valuable to other users when making booking decisions. The **review\_id** primary key uniquely identifies each review, ensuring each review is linked to a specific booking.

### ****BUSINESS RULES RELATED TO RELATIONSHIPS BETWEEN ENTITIES****

#### **RELATIONSHIP BETWEEN USERS AND BOOKINGS**

**BR19: Any user who books a room should be associated with the booking.**

The **Users** entity has a one-to-many relationship with the **Bookings** entity. Each user can have multiple bookings, but each booking is linked to one user. This ensures accurate tracking of user bookings and provides a clear view of the user's booking history.

#### **RELATIONSHIP BETWEEN USERS AND ROOMS**

**BR2: A user can book multiple rooms, but only one room at a time.**

The **Bookings** entity relates **Users** to **Rooms**, ensuring users can book multiple rooms over time but can only book one room at a time for the same dates. This prevents double bookings for the same time period and maintains accurate room availability.

#### **RELATIONSHIP BETWEEN USERS AND PAYMENTS**

**BR10: Payments must be linked to bookings, and each user can make only one payment per booking.**

The **Payments** entity connects the **Users** entity to the **Bookings** entity. This relationship ensures that each payment is unique and tied to a specific booking. It also prevents users from making duplicate payments for the same booking.

#### **RELATIONSHIP BETWEEN ADMIN AND USERS**

**BR34: An admin can manage multiple users, but each user can only have one admin managing them.**

The **Admin** entity has a one-to-many relationship with the **Users** entity, where each user is assigned to one admin (via the **PIC** attribute). This ensures clear accountability and allows admins to manage multiple users, but each user can only be managed by one admin at any given time.

**C.1. REVISED BUSINESS RULES AND ASSUMPTIONS (PU SUITES)**

1. **Account Creation** A user should be able to create an account with a username, name, email address, and password. On the creation of every user account, a unique identifier will be automatically generated to help the user access the application.
2. **Room Booking** A user should be able to book a room for specific dates (check-in and check-out). The booking process will associate the user with the room they book.
3. **Profile Editing** A user should be able to edit their profile and input their personal information such as their profile image, cover image, biography (short description of themselves), location, and website.
4. **Admin Role and PIC Assignment** An admin should be able to claim the position of a PIC (Person in Charge) for users. While a user can only have one PIC.
5. **Previous Bookings and Payment History** Users should be able to view their previous bookings and payment history to easily track their activities and financial records.
6. **Room Booking Feedback** The user should be able to like any user’s room booking as a form of feedback or personal preference.
7. **Notifications** A user should be able to receive notifications regarding booking status, promotional offers, and system messages, ensuring they stay informed about their activities.
8. **Admin Management** Admins should be able to manage the platform by monitoring user activity, managing bookings, and handling payments to maintain smooth operations.
9. **Payment Completion** Each booking made by a user will be associated with a payment record. The user must complete a payment for the booking to be finalized.
10. **Room Search** The user should be able to search for available rooms based on room type, price, availability, and location using a search filter.
11. **Booking Confirmation and Receipts** The system should provide feedback on the user’s interaction, such as booking confirmations, payment receipts, and room availability statuses.
12. **Admin Control** The system should allow admins to manage user accounts by assigning roles, viewing user activities, and removing users if necessary to ensure platform integrity.
13. **Room Availability** The system should manage room availability in real-time, ensuring that rooms shown as available can be booked immediately, preventing overbooking.
14. **Booking Modification** Users should be allowed to modify their booking dates or switch rooms if necessary, subject to room availability. This provides flexibility for users' changing plans.
15. **User Preferences** Users should be able to set preferences for room type, price range, location, and amenities when searching for rooms, to help them find suitable options.
16. **Booking Cancellation Policy** A clear cancellation policy should be in place, allowing users to cancel bookings within a specified time frame for a refund, subject to the policy terms.
17. **Room Description and Photos** Each room listing should include detailed information such as room features, amenities, photos, and descriptions, so users can make informed decisions.
18. **Payment Methods** The system should support multiple payment options such as credit cards, bank transfers, and digital wallets to accommodate various user preferences.
19. **Room Service Requests** Users should be able to request additional services such as room service, extra amenities, or housekeeping during their stay, enhancing their experience.
20. **Admin Reporting** Admins should have access to reporting tools to analyze booking trends, user activity, payment status, and platform performance.
21. **User Activity Tracking** The system should track user activity, such as room searches, bookings, and cancellations, to offer personalized recommendations and improve user experience.
22. **Booking Reminders** Users should receive reminder notifications about their upcoming bookings, check-in times, and payment due dates to ensure they stay informed.
23. **Mobile Access** The system should allow users to book rooms, manage their profiles, and receive notifications through a mobile app for easy access on the go.
24. **Data Privacy and Security** The system should ensure the privacy and security of user data, including payment information, contact details, and personal preferences, complying with data protection regulations.

C.2. REVISED ERD

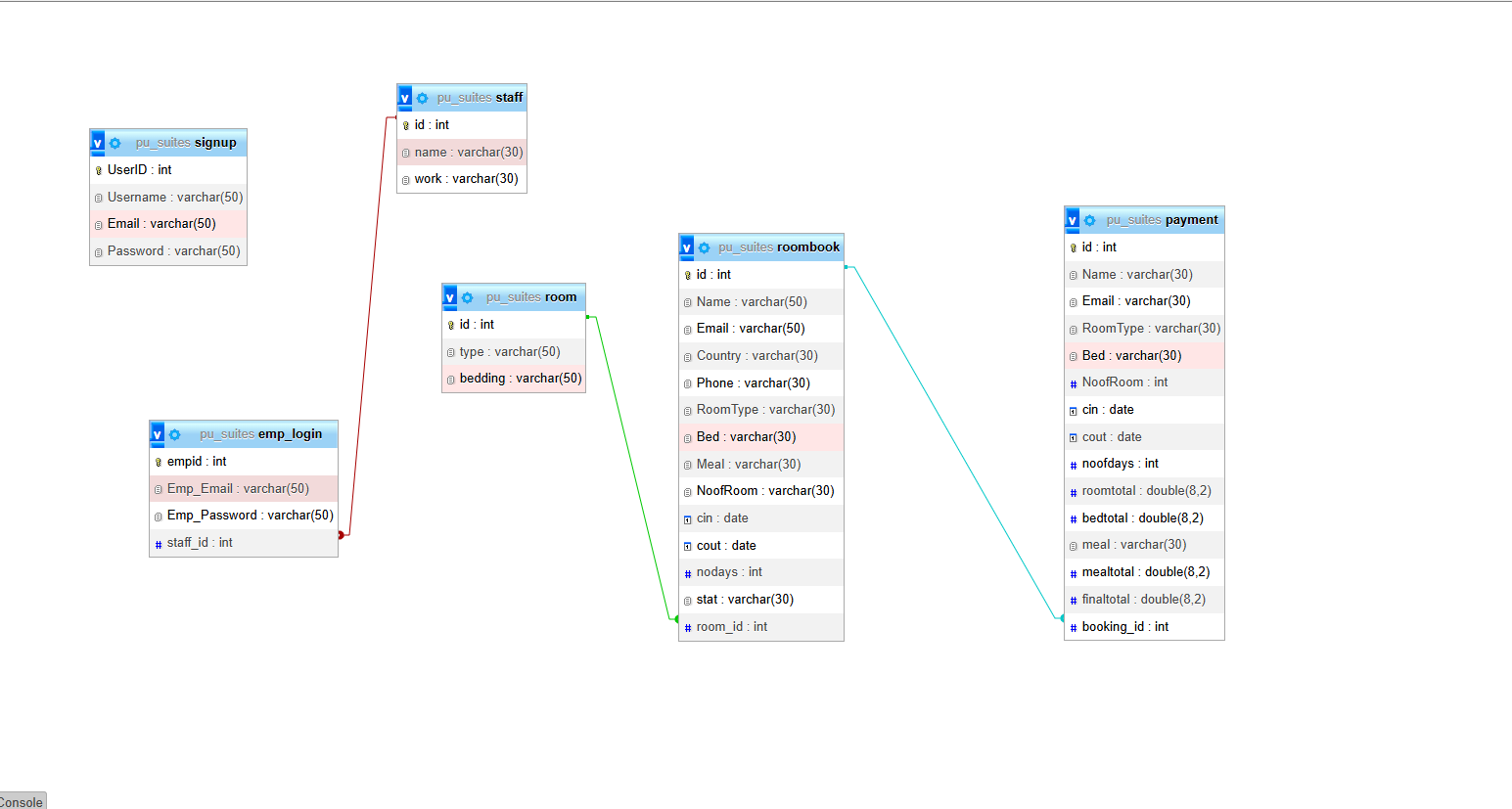


Image C.2.1 (Revised Entity Relational PU SUITES)

*user (user\_id, username, email, password, name, img\_profile, img\_cover, bio, location, website, PIC\*)*

*PIC references admins*

*follow (follower\_id\*, following\_id\*, follow\_time)*

*follower\_id references user*

*following\_id references user*

*rooms (room\_id, room\_type, price, availability\_status, room\_description, room\_capacity)*

*no references user*

*bookings (booking\_id, user\_id\*, room\_id\*, check\_in\_date, check\_out\_date, total\_cost, payment\_status)*

*user\_id references user*

*room\_id references rooms*

*payments (payment\_id, booking\_id\*, payment\_amount, payment\_method, payment\_status)*

*booking\_id references bookings*

*notifications (notif\_id, notify\_for\*, notify\_from\*, target, type, count, notif\_time)*

*notify\_for references user*

*notify\_from references user*

*comments (comment\_id, booking\_id\*, user\_id\*, comment\_msg, comment\_time)*

*booking\_id references bookings*

*user\_id references user*

*admins (admin\_id, admin\_name, email, password)*

### **C.4. LIST OF FUNCTIONAL DEPENDENCIES RELATED TO EACH BUSINESS RULES**

### ****USERS****

**BR30**: The user should be able to change their email address and username to another available and valid email address and username.

* username → {username, email, password, name, img\_profile, img\_cover, bio, location, website, PIC}
* email → {username, email, password, name, img\_profile, img\_cover, bio, location, website, PIC}

**BR1**: The user should be able to create an account with the correct email address, username, name, and password. On the creation of every user account, a unique identifier will be automatically generated to further ease the user in using the application.

* user\_id → {username, email, password, name, img\_profile, img\_cover, bio, location, website, PIC}

### ****ROOM BOOKINGS****

**BR9**: Users can book a room for specific dates (check-in and check-out). The booking process will associate the user with the room they book.

* room\_id → {name, country, phone, room\_type, bed, meal, no\_of\_room}

**BR8**: A user should be able to book a room by selecting a check-in and check-out date.

* booking\_id → {user\_id, room\_id, check\_in\_date, check\_out\_date}

**BR16**: A user can make only one booking per room for the same dates.

* {user\_id, room\_id, check\_in\_date, check\_out\_date} → booking\_id

### ****ROOMS****

**BR3**: Each room should have a unique identifier. The room information will include type, price, availability, description, and capacity.

* room\_id → {room\_type, price, availability\_status, room\_description, room\_capacity}

**BR19**: Any user who books a room should be associated with the booking.

* room\_id → user\_id

### ****PAYMENTS****

**BR12**: Each booking made by a user will be associated with a payment record. The user must complete a payment for the booking to be finalized.

* payment\_id → {booking\_id, user\_id, payment\_amount, payment\_method, payment\_status}

**BR18**: Payments should be linked to a user’s booking, and a payment can be made only once for a specific booking.

* payment\_id → {booking\_id}

### ****ADMIN****

**BR32**: Admins should be able to manage user accounts by registering as administrators and logging into the admin panel.

* admin\_name → admin\_id
* email → admin\_id
* admin\_id → {admin\_name, email, password}

**BR33**: Admins will be able to monitor user activities, assign PIC roles, and remove users if needed.

* admin\_id → {user\_id}

### ****NOTIFICATIONS****

**BR14**: Notifications should be sent to users regarding booking updates, promotional offers, and system messages.

* notif\_id → {notify\_for, notify\_from, target, type, count, notif\_time}

**BR13**: The notifications should be provided in the latest to oldest order and when the user opens the notification, the user will cease to be notified.

* notif\_id → notif\_time

### ****REVIEWS****

**BR17**: Users should be able to review rooms they have stayed in.

* review\_id → {user\_id, room\_id, rating, review\_text, created\_at}

### ****RELATIONSHIPS BETWEEN ENTITIES****

#### **USERS AND ROOM BOOKINGS**

**BR19**: Any user who books a room should be associated with the booking.

* user\_id → {room\_booking\_id}

#### **USERS AND ROOMS**

**BR2**: A user can book multiple rooms, but only one room at a time.

* {user\_id, room\_id} → booking\_id

#### **USERS AND PAYMENTS**

**BR10**: Payments must be linked to bookings, and each user can make only one payment per booking.

* payment\_id → {user\_id, booking\_id}

#### **ADMINS AND USERS**

**BR34**: An admin can manage multiple users, but each user can only have one admin managing them.

* admin\_id → {user\_id}

C.5. NORMALIZATION (LOGIC DESIGN)

### ****USER ENTITY****

**Functional Dependencies**:

* username → {username, email, password, name, img\_profile, img\_cover, bio, location, website, PIC}
* email → {username, email, password, name, img\_profile, img\_cover, bio, location, website, PIC}
* user\_id → {username, email, password, name, img\_profile, img\_cover, bio, location, website, PIC}

1NF: Yes, it has a unique identifier for each row (user\_id) and no repeating attributes.  
2NF: Yes, every non-key attribute is functionally dependent on the PK (user\_id).  
3NF: Yes, there are no transitive functional dependencies.  
BCNF: Yes, there are two reasons for this:

1. Even though there are functional dependencies not from the primary key on the users entity, the determinator (email, username) is still a superkey for schema users.
2. When we consider the functional dependencies with the determinant (email and username), we can see that it is a trivial dependency.

Final Relation:

plaintext

SalinEdit

users (user\_id, username, email, password, name, img\_profile, img\_cover, bio, location, website, PIC\*)

* PIC references admins

### ****USER ENTITY****

**Functional Dependencies**:

* username → {username, email, password, name, img\_profile, img\_cover, bio, location, website, PIC}
* email → {username, email, password, name, img\_profile, img\_cover, bio, location, website, PIC}
* user\_id → {username, email, password, name, img\_profile, img\_cover, bio, location, website, PIC}

1NF: Yes, it has a unique identifier for each row (user\_id) and no repeating attributes.  
2NF: Yes, every non-key attribute is functionally dependent on the PK (user\_id).  
3NF: Yes, there are no transitive functional dependencies.  
BCNF: Yes, there are two reasons for this:

1. Even though there are functional dependencies not from the primary key on the users entity, the determinator (email, username) is still a superkey for schema users.
2. When we consider the functional dependencies with the determinant (email and username), we can see that it is a trivial dependency.

Final Relation:

plaintext

SalinEdit

users (user\_id, username, email, password, name, img\_profile, img\_cover, bio, location, website, PIC\*)

PIC references admins

### ****BOOKINGS ENTITY****

**Functional Dependencies**:

* booking\_id → {user\_id, room\_id, check\_in\_date, check\_out\_date, total\_cost, payment\_status}

1NF: Yes, it has a unique identifier for each row (booking\_id) and no repeating attributes.  
2NF: Yes, every non-key attribute is functionally dependent on the PK (booking\_id).  
3NF: Yes, there are no transitive functional dependencies.  
BCNF: Yes, there are no other functional dependencies not from the primary key on the bookings entity.

Final Relation:

plaintext

SalinEdit

bookings (booking\_id, user\_id\*, room\_id\*, check\_in\_date, check\_out\_date, total\_cost, payment\_status)

* user\_id references users
* room\_id references rooms

### ****PAYMENTS ENTITY****

**Functional Dependencies**:

* payment\_id → {booking\_id, user\_id, payment\_amount, payment\_method, payment\_status}

1NF: Yes, it has a unique identifier for each row (payment\_id) and no repeating attributes.  
2NF: Yes, every non-key attribute is functionally dependent on the PK (payment\_id).  
3NF: Yes, there are no transitive functional dependencies.  
BCNF: Yes, there are no other functional dependencies not from the primary key on the payments entity.

Final Relation:

plaintext

SalinEdit

payments (payment\_id, booking\_id\*, user\_id\*, payment\_amount, payment\_method, payment\_status)

* booking\_id references bookings
* user\_id references users

### ****ADMINS ENTITY****

**Functional Dependencies**:

* admin\_name → {admin\_name, email, password}
* email → {admin\_name, email, password}
* admin\_id → {admin\_name, email, password}

1NF: Yes, it has a unique identifier for each row (admin\_id) and no repeating attributes.  
2NF: Yes, every non-key attribute is functionally dependent on the PK (admin\_id).  
3NF: Yes, there are no transitive functional dependencies.  
BCNF: Yes, there are two reasons for this:

1. Even though there are functional dependencies not from the primary key on the admins entity, the determinator (email, admin\_name) is still a superkey for schema admins.
2. When we consider the functional dependencies with the determinant (email and admin\_name), we can see that it is a trivial dependency.

Final Relation:

plaintext

SalinEdit

admins (admin\_id, admin\_name, email, password)

### ****NOTIFICATIONS ENTITY****

**Functional Dependencies**:

* notif\_id → {notify\_for, notify\_from, target, type, count, notif\_time}

1NF: Yes, it has a unique identifier for each row (notif\_id) and no repeating attributes.  
2NF: Yes, every non-key attribute is functionally dependent on the PK (notif\_id).  
3NF: Yes, there are no transitive functional dependencies.  
BCNF: Yes, there are no other functional dependencies not from the primary key on the notifications entity.

Final Relation:

plaintext

SalinEdit

notifications (notif\_id, notify\_for\*, notify\_from\*, target, type, count, notif\_time)

* notify\_for references users
* notify\_from references users

### ****REVIEWS ENTITY****

**Functional Dependencies**:

* review\_id → {user\_id, room\_id, rating, review\_text, created\_at}

1NF: Yes, it has a unique identifier for each row (review\_id) and no repeating attributes.  
2NF: Yes, every non-key attribute is functionally dependent on the PK (review\_id).  
3NF: Yes, there are no transitive functional dependencies.  
BCNF: Yes, there are no other functional dependencies not from the primary key on the reviews entity.

Final Relation:

plaintext

SalinEdit

reviews (review\_id, user\_id\*, room\_id\*, rating, review\_text, created\_at)

* user\_id references users
* room\_id references rooms

### ****RELATIONSHIPS BETWEEN ENTITIES****

#### **USERS AND BOOKINGS**

**BR19**: Any user who books a room should be associated with the booking.

* user\_id → {booking\_id}

#### **USERS AND ROOMS**

**BR2**: A user can book multiple rooms, but only one room at a time.

* {user\_id, room\_id} → booking\_id

#### **USERS AND PAYMENTS**

**BR10**: Payments must be linked to bookings, and each user can make only one payment per booking.

* payment\_id → {user\_id, booking\_id}

#### **ADMINS AND USERS**

**BR34**: An admin can manage multiple users, but each user can only have one admin managing them.

* admin\_id → {user\_id}

**D.1. THE DATABASE**

**Queries:**

sql

SalinEdit

CREATE DATABASE pu\_suites;

CREATE TABLE `admins` (

`admin\_id` int(11) NOT NULL,

`admin\_name` varchar(50) DEFAULT NULL,

`email` varchar(50) DEFAULT NULL,

`password` varchar(50) DEFAULT NULL

);

CREATE TABLE `rooms` (

`room\_id` int(11) NOT NULL,

`room\_type` varchar(50) NOT NULL,

`price` decimal(10,2) NOT NULL,

`availability\_status` enum('available', 'booked') NOT NULL,

`room\_description` varchar(255) NOT NULL,

`room\_capacity` int(11) NOT NULL

);

CREATE TABLE `bookings` (

`booking\_id` int(11) NOT NULL,

`user\_id` int(11) NOT NULL,

`room\_id` int(11) NOT NULL,

`check\_in\_date` date NOT NULL,

`check\_out\_date` date NOT NULL,

`total\_cost` decimal(10,2) NOT NULL,

`payment\_status` enum('pending', 'completed') NOT NULL

);

CREATE TABLE `payments` (

`payment\_id` int(11) NOT NULL,

`booking\_id` int(11) NOT NULL,

`payment\_amount` decimal(10,2) NOT NULL,

`payment\_method` varchar(50) NOT NULL,

`payment\_status` enum('pending', 'completed') NOT NULL

);

CREATE TABLE `users` (

`user\_id` int(11) NOT NULL,

`username` varchar(40) NOT NULL,

`email` varchar(255) NOT NULL,

`password` varchar(32) NOT NULL,

`name` varchar(40) NOT NULL,

`img\_profile` varchar(255) NOT NULL DEFAULT 'default.jpg',

`bio` varchar(140) NOT NULL DEFAULT '',

`location` varchar(255) NOT NULL DEFAULT '',

`website` varchar(255) NOT NULL DEFAULT '',

`PIC` int(11) DEFAULT NULL

);

ALTER TABLE `admins`

ADD PRIMARY KEY (`admin\_id`);

ALTER TABLE `rooms`

ADD PRIMARY KEY (`room\_id`);

ALTER TABLE `bookings`

ADD PRIMARY KEY (`booking\_id`),

ADD KEY `user\_id` (`user\_id`),

ADD KEY `room\_id` (`room\_id`);

ALTER TABLE `payments`

ADD PRIMARY KEY (`payment\_id`),

ADD KEY `booking\_id` (`booking\_id`);

ALTER TABLE `users`

ADD PRIMARY KEY (`user\_id`),

ADD KEY `users\_ibfk\_1` (`PIC`);

ALTER TABLE `rooms`

MODIFY `room\_id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=1;

ALTER TABLE `users`

MODIFY `user\_id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=1;

ALTER TABLE `bookings`

MODIFY `booking\_id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=1;

ALTER TABLE `payments`

MODIFY `payment\_id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=1;

ALTER TABLE `admins`

MODIFY `admin\_id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=1;

ALTER TABLE `bookings`

ADD CONSTRAINT `bookings\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `users` (`user\_id`) ON DELETE CASCADE,

ADD CONSTRAINT `bookings\_ibfk\_2` FOREIGN KEY (`room\_id`) REFERENCES `rooms` (`room\_id`) ON DELETE CASCADE;

ALTER TABLE `payments`

ADD CONSTRAINT `payments\_ibfk\_1` FOREIGN KEY (`booking\_id`) REFERENCES `bookings` (`booking\_id`) ON DELETE CASCADE;

**D.2. THE DATA**

Insert statements for adding some data into your PU SUITES system.

sql

SalinEdit

INSERT INTO `admins` (`admin\_name`, `email`, `password`) VALUES

('admin1', 'admin1@pusuites.com', 'password1'),

('admin2', 'admin2@pusuites.com', 'password2');

INSERT INTO `users` (`username`, `email`, `password`, `name`, `img\_profile`, `bio`, `location`, `website`, `PIC`) VALUES

('john\_doe', 'john@example.com', 'password123', 'John Doe', 'default.jpg', 'Traveler', 'New York', 'www.johndoe.com', 1),

('jane\_doe', 'jane@example.com', 'password123', 'Jane Doe', 'default.jpg', 'Explorer', 'Los Angeles', 1);

INSERT INTO `rooms` (`room\_type`, `price`, `availability\_status`, `room\_description`, `room\_capacity`) VALUES

('Single', 100.00, 'available', 'A single room with a queen-sized bed.', 1),

('Double', 150.00, 'available', 'A double room with two queen-sized beds.', 2);

INSERT INTO `bookings` (`user\_id`, `room\_id`, `check\_in\_date`, `check\_out\_date`, `total\_cost`, `payment\_status`) VALUES

(1, 1, '2023-05-01', '2023-05-05', 400.00, 'completed'),

(2, 2, '2023-05-03', '2023-05-06', 450.00, 'pending');

INSERT INTO `payments` (`booking\_id`, `payment\_amount`, `payment\_method`, `payment\_status`) VALUES

(1, 400.00, 'Credit Card', 'completed'),

(2, 450.00, 'Credit Card', 'pending');

**D.3. QUERIES**

**SELECT all users' information:**

sql

SalinEdit

SELECT \* FROM `users`;

**SELECT all rooms available for booking:**

sql

SalinEdit

SELECT \* FROM `rooms` WHERE `availability\_status` = 'available';

**SELECT all bookings of a specific user:**

sql

SalinEdit

SELECT \* FROM `bookings` WHERE `user\_id` = 1;

**SELECT all payments made by users:**

sql

SalinEdit

SELECT \* FROM `payments` WHERE `payment\_status` = 'completed';

**COUNT number of bookings per room:**

sql

SalinEdit

SELECT `room\_id`, COUNT(\*) as `number\_of\_bookings` FROM `bookings` GROUP BY `room\_id`;

**D.4. DATABASE DESIGN AND RELATIONSHIPS**

In this section, we summarize the relationships between the entities in the PU SUITES database:

1. **Users and Bookings**:
   * Each user can make multiple bookings, but each booking is tied to a specific user.
2. **Bookings and Rooms**:
   * Each booking corresponds to a specific room, and a room can have multiple bookings, as long as the dates don’t overlap.
3. **Bookings and Payments**:
   * Each booking will have one payment associated with it. The payment tracks the booking's payment status and method.
4. **Admins and Users**:
   * Admins manage users. Each user can have a specific admin assigned to them.

**D.5. DATABASE INTEGRITY AND CONSTRAINTS**

**Primary Keys**:

* user\_id for the users table
* room\_id for the rooms table
* booking\_id for the bookings table
* payment\_id for the payments table

**Foreign Keys**:

* The bookings table references users and rooms.
* The payments table references bookings.
* The admins table is referenced by users via the PIC column.