# Database Design Deliverable 2

**Entities & Attributes:**

- **User** (default Django User)
    - username, password, email, etc.
- **Applicant**
    - user (FK → User)
    - phone
    - image
    - gender
    - type
- **Company**
    - user (FK → User)
    - phone
    - image
    - gender
    - type
    - status
    - company_name

- **Job**
    - company (FK → Company)
    - start_date
    - end_date
    - title
    - salary
    - image
    - description
    - experience
    - location
    - skills
    - creation_date
- **Application**
    - company (Text)
    - job (FK → Job)
    - applicant (FK → Applicant)
    - resume
    - apply_date
- **Notification**
    - recipient (FK → User)
    - message
    - created_at
    - is_read

- **AdminActivityLog**
    - admin_user (FK → User, staff only)
    - action
    - timestamp
    - description

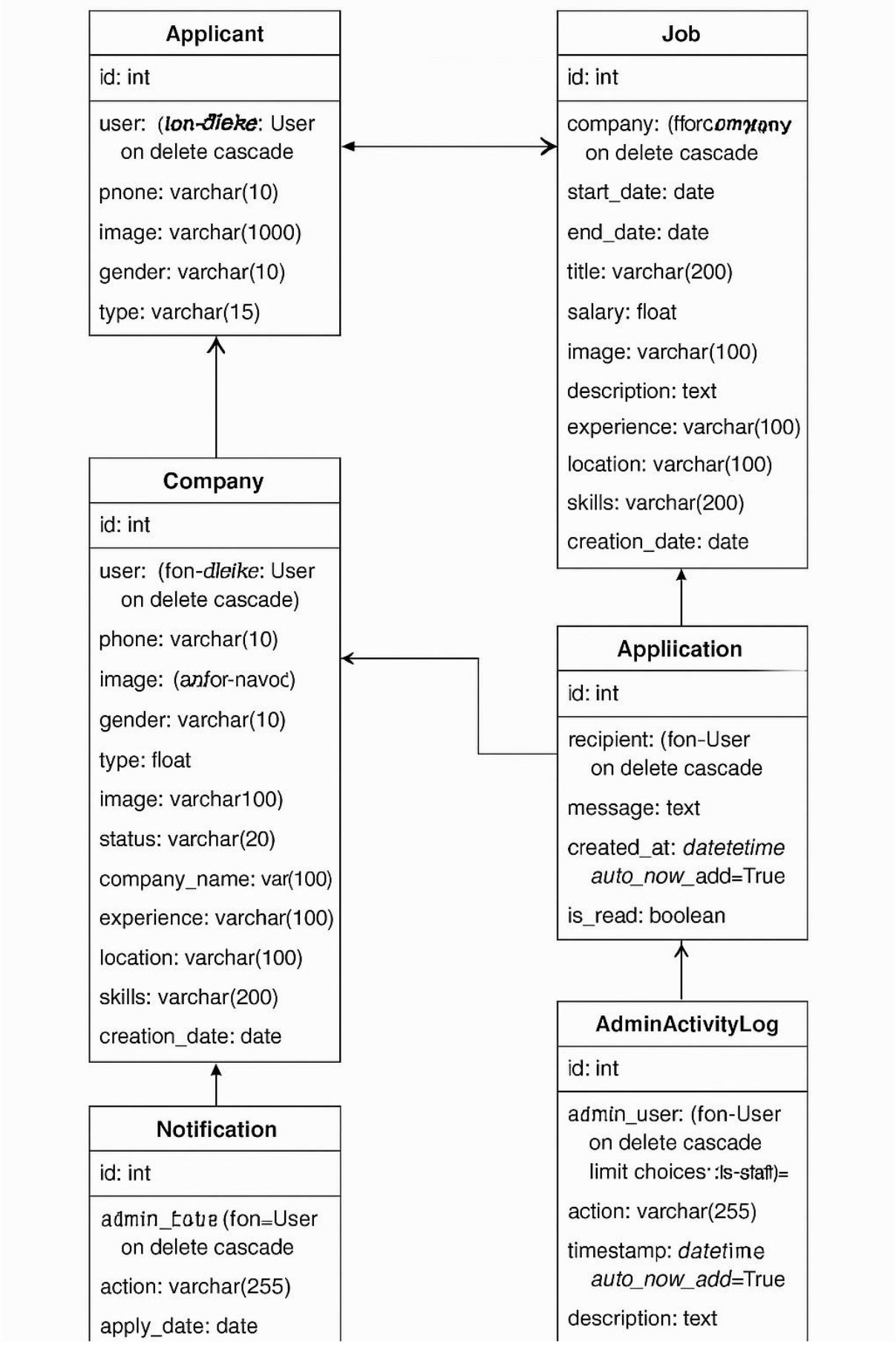
**Relationships:**

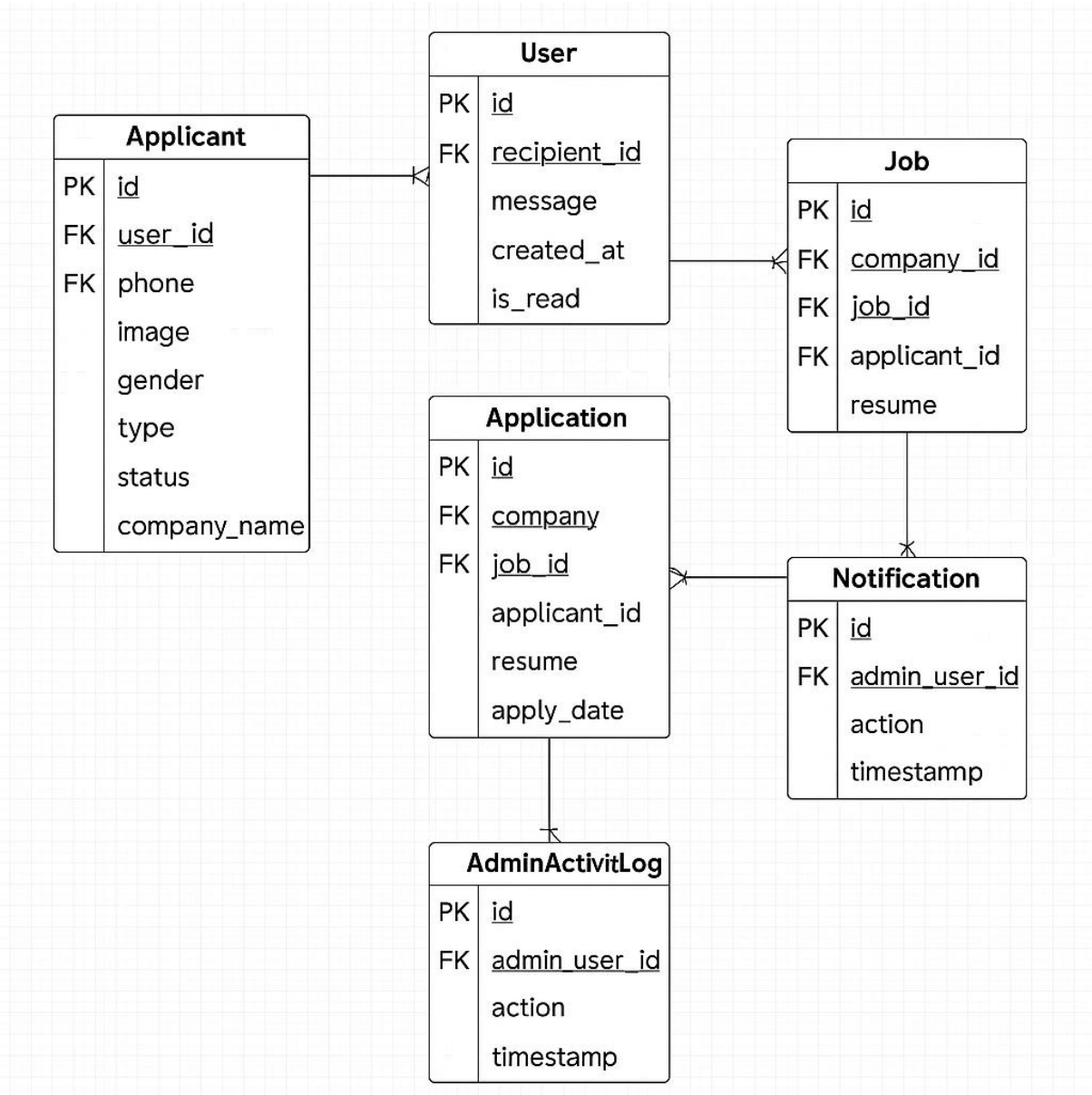
- User → Applicant (1:1 or 1:M)
- User → Company (1:1 or 1:M)
- Company → Job (1:M)
- Job → Application (1:M)
- Applicant → Application (1:M)

- User → Notification (1:M)

- User (Admin) → AdminActivityLog (1:M)

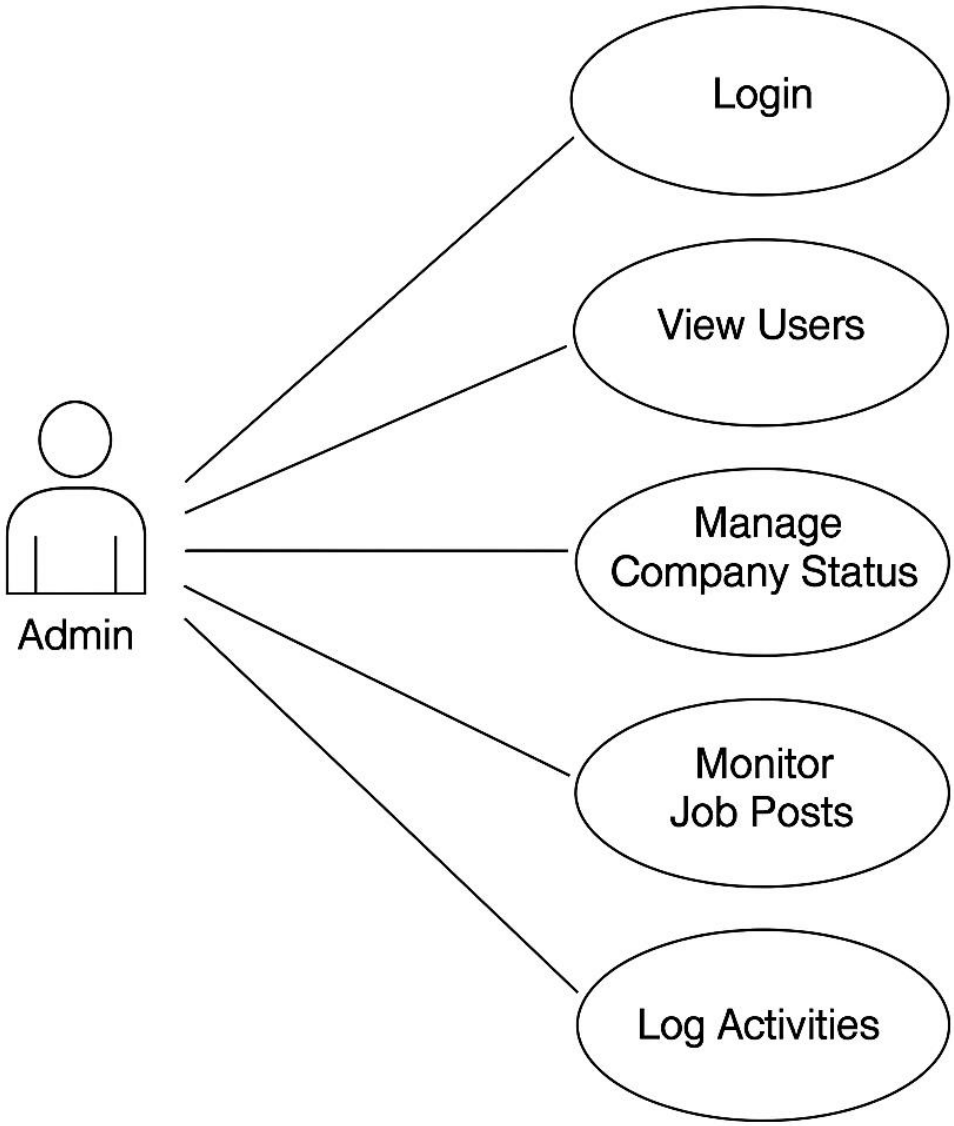
**Entity Relationship Diagram (ERD)**

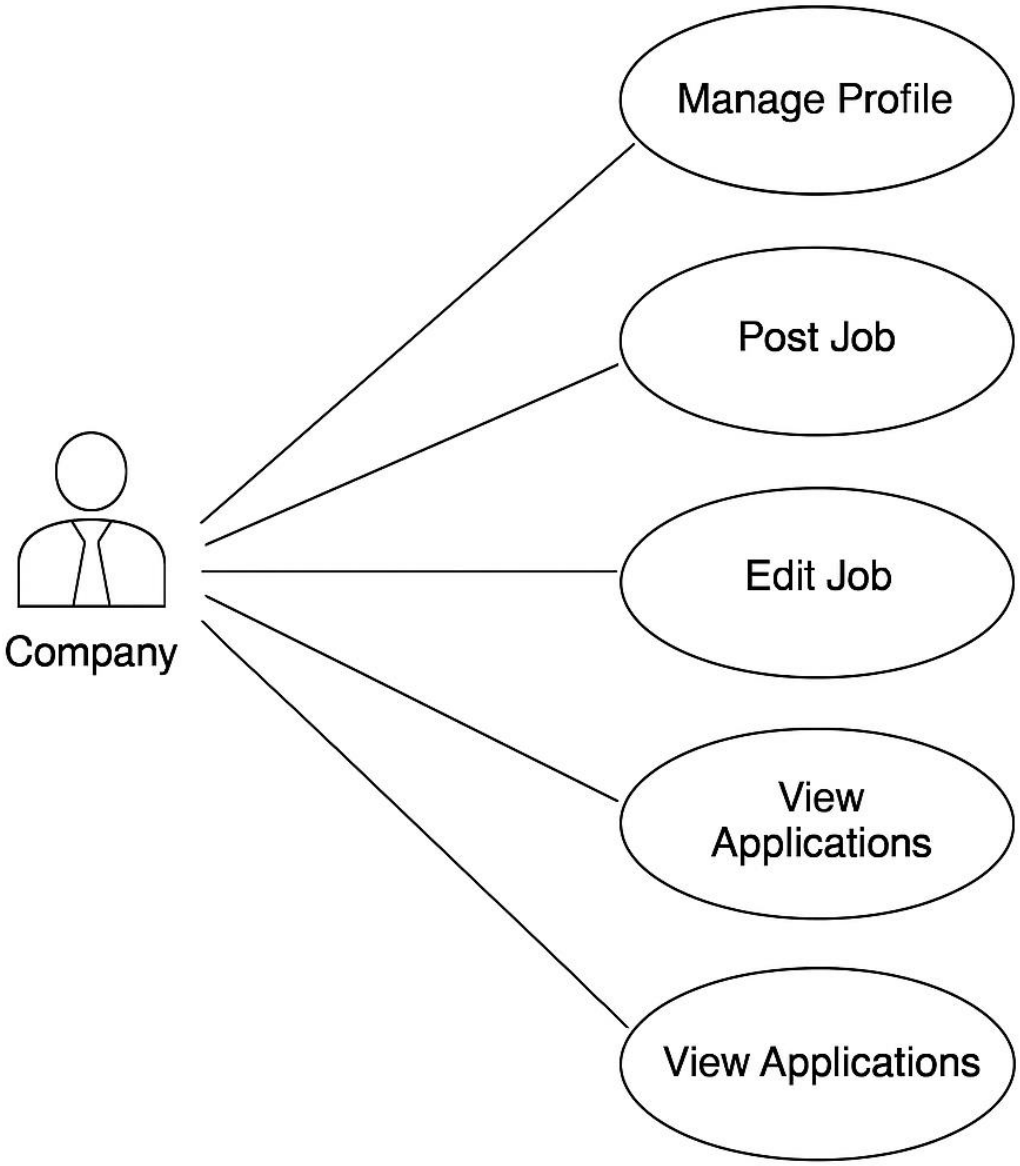


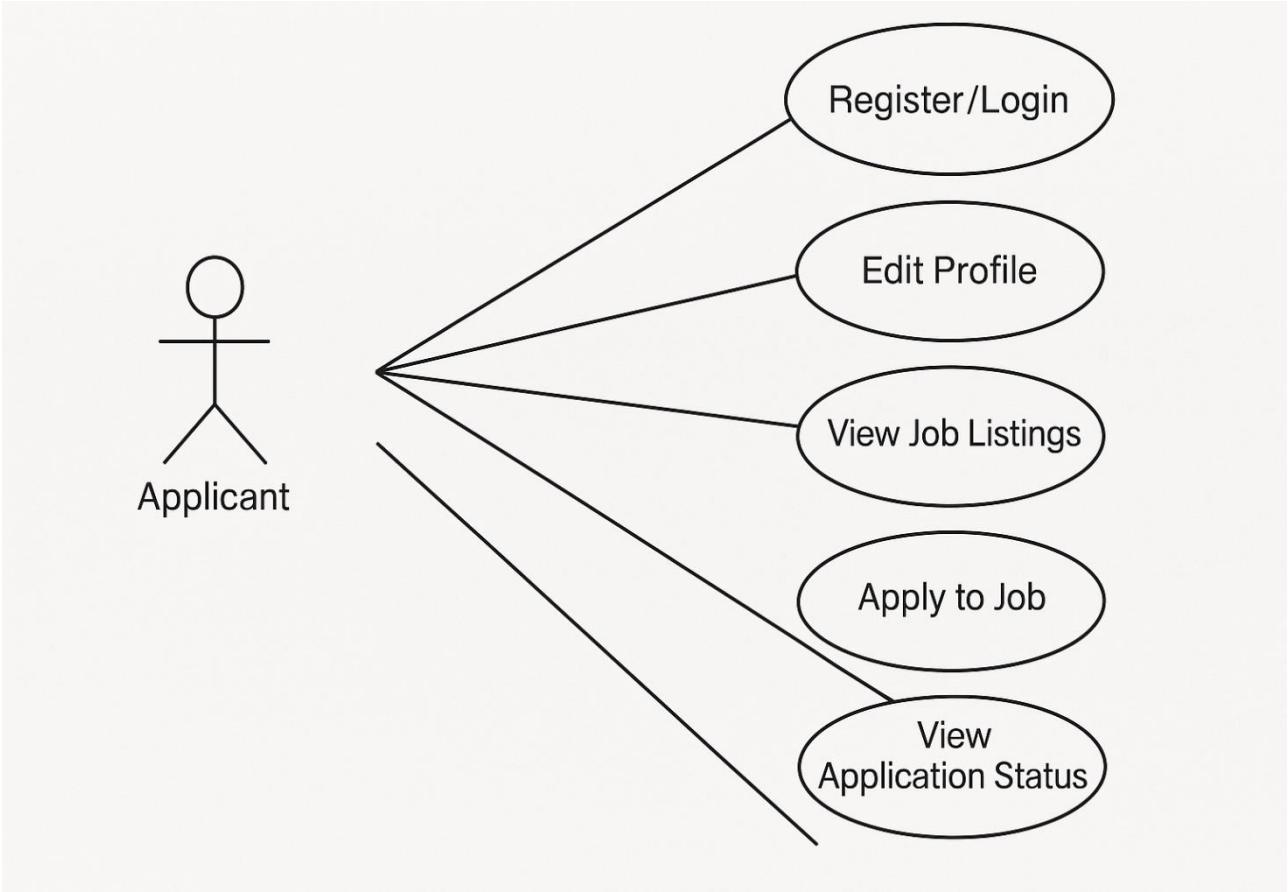

User
PK id
FK recipient_id
message
created_at
is_read
Applicant
PK id
FK user_id
FK phone
image
gender
type
status
company_name
Job
PK id
FK company_id
FK job_id
FK applicant_id
resume
Application
PK id
FK company
FK job_id
applicant_id
resume
apply_date
Notification
PK id
FK admin_user_id
action
timestamp
AdminActivitLog
PK id
FK admin_user_id
action
timestamp

**Use Cases**

1. **Register as Applicant**
    - Actor: User
    - Description: User signs up and creates an applicant profile.
2. **Register as Company**
    - Actor: User
    - Description: User signs up and creates a company profile.
3. **Post Job**
    - Actor: Company
    - Description: Company posts job details and requirements.
4. **Apply for Job**
    - Actor: Applicant
    - Description: Applicant browses jobs and submits applications.
5. **View Applications**
    - Actor: Company
    - Description: Company views applications submitted for jobs.
6. **Send Notification**
    - Actor: System / Admin / Company
    - Description: Notification is sent to users (e.g., application updates).
7. **View Notification**
    - Actor: Applicant / Company
    - Description: User views received notifications.
8. **Admin Activity Log**
    - Actor: Admin
    - Description: Admin actions are recorded in the activity log.
9. **Login / Logout**
    - Actor: All users
    - Description: Users log in to access their functionalities.


Login
View Users
Manage
Company Status
Monitor
Job Posts
Log Activities
Admin


Manage Profile
Post Job
Edit Job
View Applications
View Applications
Company


Register/Login
Edit Profile
View Job Listings
Apply to Job
View
Application Status
Applicant

Feature: Data Update Verification

As an application user

I want to ensure that my data can be correctly updated in the database.

So that my profile information remains accurate.

Scenario 1: Successfully updating a user's email address.

Given a user with username "update_user" and email "old@example.com" exists in the "users" table

And I have the following updated user details:

| email | |

| updated@example.com |

When I attempt to update the email address of the user with username "update_user" to "updated@example.com"

Then the "users" table should contain a record with username "update_user" and email "updated@example.com"

And the "users" table should not contain a record with username "update_user" and email "old@example.com"

Scenario 2: Attempting to update a non-existent user's email.

Given no user exists in the "users" table with username "non_existent_user"

When I attempt to update the email address of the user with username "non_existent_user" to "new@example.com"

Then the update should fail

And no changes should be made to the "users" table.

And an error message indicating the user was not found should be logged or raised (if applicable)

Feature: Data Retrieval Verification

As a system administrator

I want to verify that data can be accurately retrieved from the database.

So that I can monitor and manage the application's data effectively.

Scenario 1: Retrieving an existing user by username.

Given a user with username "test_user" and email "test@example.com" exists in the "users" table

When I query the "users" table for the user with username "test_user"

Then the retrieved record should contain:

| username | email | |

| test_user | test@example.com |

Scenario 2: Attempting to retrieve a non-existent user by ID.

Given no user exists in the "users" table with an ID of "999"

When I attempt to retrieve the user with ID "999" from the "users" table

Then no record should be returned

And a "user not found" message should be displayed or logged (if applicable).

Feature: Data Update Verification

As an application user

I want to ensure that my data can be correctly updated in the database.

So that my profile information remains accurate.

Scenario 1: Successfully updating a user's email address.

Given a user with username "update_user" and email "old@example.com" exists in the "users" table

And I have the following updated user details:

| email |

| updated@example.com |

When I attempt to update the email address of the user with username "update_user" to "updated@example.com"

Then the "users" table should contain a record with username "update_user" and email "updated@example.com"

And the "users" table should not contain a record with username "update_user" and email "old@example.com"

Scenario 2: Attempting to update a non-existent user's email.

Given no user exists in the "users" table with username "non_existent_user"

When I attempt to update the email address of the user with username "non_existent_user" to "new@example.com"

Then the update should fail

And no changes should be made to the "users" table.

And an error message indicating the user was not found should be logged or raised (if applicable).

Explanation of Components:

Feature: A high-level description of the functionality being evaluated.

As a [role]: Defines the stakeholder benefiting from the feature.

I want [goal]: Describes the action the stakeholder wants to perform.

So that [benefit]: Explains the reason or value behind the goal.

Scenario: A specific test case illustrating the feature.

Given: Sets up the initial context or preconditions.

When: Describes the action taken by the user or system.

Then: Specifies the expected outcome or result.

Tables in Given/Then: Used to represent data in a structured format for clarity.