# Task 2: Database Models and User Management

**Objective:** To define the User table structure in your database using SQLAlchemy's ORM and to set up Flask-Migrate, the tool that will manage changes to your database schema. By the end of this task, you will have a dev.db file in your instance folder containing an empty users table.
**Estimated Time:** 2-3 hours

## Step 1: Define the User Model

**File to Modify:** app/models.py
**Action:**

1. Import the db object from app.extensions.
2. Create a class named User that inherits from db.Model. This class will represent the users table in your database.
3. Inside the User class, define the columns for your table. Each column is an instance of the db.Column class. You will need the following columns:
    - id: An integer that will be the primary key for the table.
    - username: A string, which must be unique.
    - email: A string, which also must be unique.
    - password_hash: A string to store the hashed version of the user's password. **Crucially, you must never store passwords directly.**
    - role: A string to store the user's role (e.g., 'user' or 'admin').
    - is_approved: A boolean (True/False) to track if an admin has approved the user's account. It should default to False.

**Why:** A SQLAlchemy model provides an object-oriented way (class User) to interact with a relational database table (users). Instead of writing raw SQL, you can create, read, update, and delete users by interacting with Python objects, which is safer and more maintainable.
**Documentation:**

- Flask-SQLAlchemy: Declaring Models
- Flask-SQLAlchemy: Column and Data Types

## Step 2: Integrate Flask-Migrate

**File to Modify:** app/__init__.py (Your Application Factory)
Action:
You've already initialized the Migrate extension in app/extensions.py. Now you just need to ensure it's properly linked to both the Flask app and the SQLAlchemy db instance inside your factory.

1. Inside your create_app function, after you have initialized db.init_app(app), add the line to initialize migrate.init_app(app, db).

**Why:** Flask-Migrate acts as the bridge between your SQLAlchemy models (what your database *should* look like) and Alembic (the tool that actually performs the database schema changes). This line explicitly connects the Flask application instance and the database

instance to the migration engine.
**Documentation:**

- Flask-Migrate Initializing

## Step 3: Create the Initial Database Migration

Action:
This step is performed in your terminal. Make sure you are in the root directory of your project (exoplanet_analyzer/).

1. **Initialize the migration repository (only do this once per project):**
   ```
   flask db init
   ```
   This will create a new migrations folder. This folder is essential and should be committed to Git.
2. **Generate the first migration script:**
   ```
   flask db migrate -m "Initial user model"
   ```
   This command will inspect your User model, compare it to the (non-existent) database, and automatically generate a Python script in the migrations/versions/ folder that contains the instructions to create the users table. The -m flag adds a helpful message.

**Why:** Instead of manually writing SQL CREATE TABLE statements, we let the tools do it for us. This is less error-prone and creates a repeatable history of every change made to your database structure. Anyone who clones your project can run these migrations to perfectly replicate your database schema.
**Documentation:**

- Flask-Migrate Creating a Migration Repository
- Flask-Migrate Creating a Migration Script

## Step 4: Apply the Migration to the Database

Action:
This is the final step, also performed in your terminal.

1. **Upgrade the database:**
   ```
   flask db upgrade
   ```

**Why:** The migrate command only *created* the instruction script. The upgrade command is what actually *runs* the script, connects to your database, and executes the necessary SQL to create the users table.
Verification:
After running flask db upgrade, look inside your instance folder. You should now see a dev.db file. You have successfully created your database and its first table.
You are now ready for the next task: building the user authentication system. Let me know when you've completed this one.