Ticketing System

Overview

The Ticketing System Project is a part of the final project for the 'FUTURE COLLARS BOOTCAMP PYTHON DEVELOPER' course. It is a simplified ticketing system designed to be deployed in a production environment and can be customized to meet the needs of the end-users.
The project utilizes Python Flask as the web framework and SQLAlchemy as the ORM (Object-Relational Mapping) tool for database interactions. It provides basic ticket management functionalities such as ticket creation, status updates, and comments.

Key Features

- User authentication and authorization
- Ticket creation with title, description, and status
- Ticket status updates (e.g., New, In Progress, Resolved, Closed)
- Adding comments to tickets
- Different dashboard views for users, HR personnel, and administrators
- Email notifications for ticket creation and updates

Technologies used

- Python Flask: Web framework for building the application.
- SQLAlchemy: ORM tool for interacting with the database.
- Flask-WTF: Integration of WTForms for form handling.
- Flask-Mail: Extension for sending email notifications.

Installation

1. Clone the Repository:
    - ➢ git clone https://github.com/LuisMhaske/Ticketing_System.git
2. Navigate to project directory:
    - ➢ cd <project_directory>
3. Install dependencies:
    - ➢ pip install -r requirements.txt

Usage

1. Run the application:
    - ➢ python run.py
2. Access the application in your web browser:
    - ➢ http://localhost:8080/

Project Structure

The project directory is organized as follows:

```
TicketingSystem/
 |
 ├── app/
 |    ├── templates/
 |    |    ├── admin/
 |    |    |   └── hr_approvals.html
 |    |    ├── associate/
 |    |    |   └── create_ticket.html
 |    |    ├── base.html
 |    |    ├── dashboard.html
 |    |    ├── hr_signup.html
 |    |    ├── login.html
 |    |    └── registration.html
 |    ├── __init__.py
 |    ├── auth.py
 |    ├── extension.py
 |    ├── forms.py
 |    └── models.py
 |
 ├── requirements.txt
 └── run.py
```

- **app/**: Contains the main application files, including models, forms, routes/__init__, and templates.
- **run.py**: Script to run the Flask application.
- **requirements.txt**: List of dependencies required to run the project.

Workflow

The workflow of the Ticketing System can be summarized as follows:

1. **Home Page**:
    - The Home Page serves as the entry point for users.
    - Users can navigate to various sections of the system, such as Sign Up, Login, and Ticket Creation.
2. **Sign Up**:
    - Users can register for an account by providing necessary information.
    - Differentiate between Associates, HR, and Admin during the sign-up process.
3. **Login**:
    - Registered users can securely log in to their accounts using their credentials.
4. **Ticket Creation**:
    - Once logged in, users can create new tickets to report their concerns or issues.
    - Users provide details such as title, description, and possibly category or priority level for the ticket.
    - Tickets are associated with the user who created them and are stored securely in the database.
5. **User Dashboard**:
    - Associates have access to their dashboard, where they can:
    - View their previously opened tickets.
    - Monitor the progress of their tickets.
6. **HR Dashboard**:
    - HR team members have access to the HR dashboard, where they can:
    - View tickets from all users.
    - Start working on tickets assigned to them.
    - Change the status of tickets based on their progress (e.g., New, In Progress, Resolved, Closed).
7. **Admin Portal**:
    - Admins have special privileges to manage user roles and permissions.
    - Admins can:
    - Approve or disapprove HR sign-up requests.
    - Manage user roles and permissions, including granting access to the HR portal.
    - Overall, the Ticketing System ensures privacy and security by segregating user roles and providing appropriate access controls. It streamlines the process of issue reporting and resolution within the organization.

Future Improvements

- Improve user interface and user experience.
- Implement advanced features such as file attachments, ticket assignments, and priority levels.
- Enhance security measures such as input validation and user permissions.
- Conduct thorough testing and debugging to ensure reliability and stability.