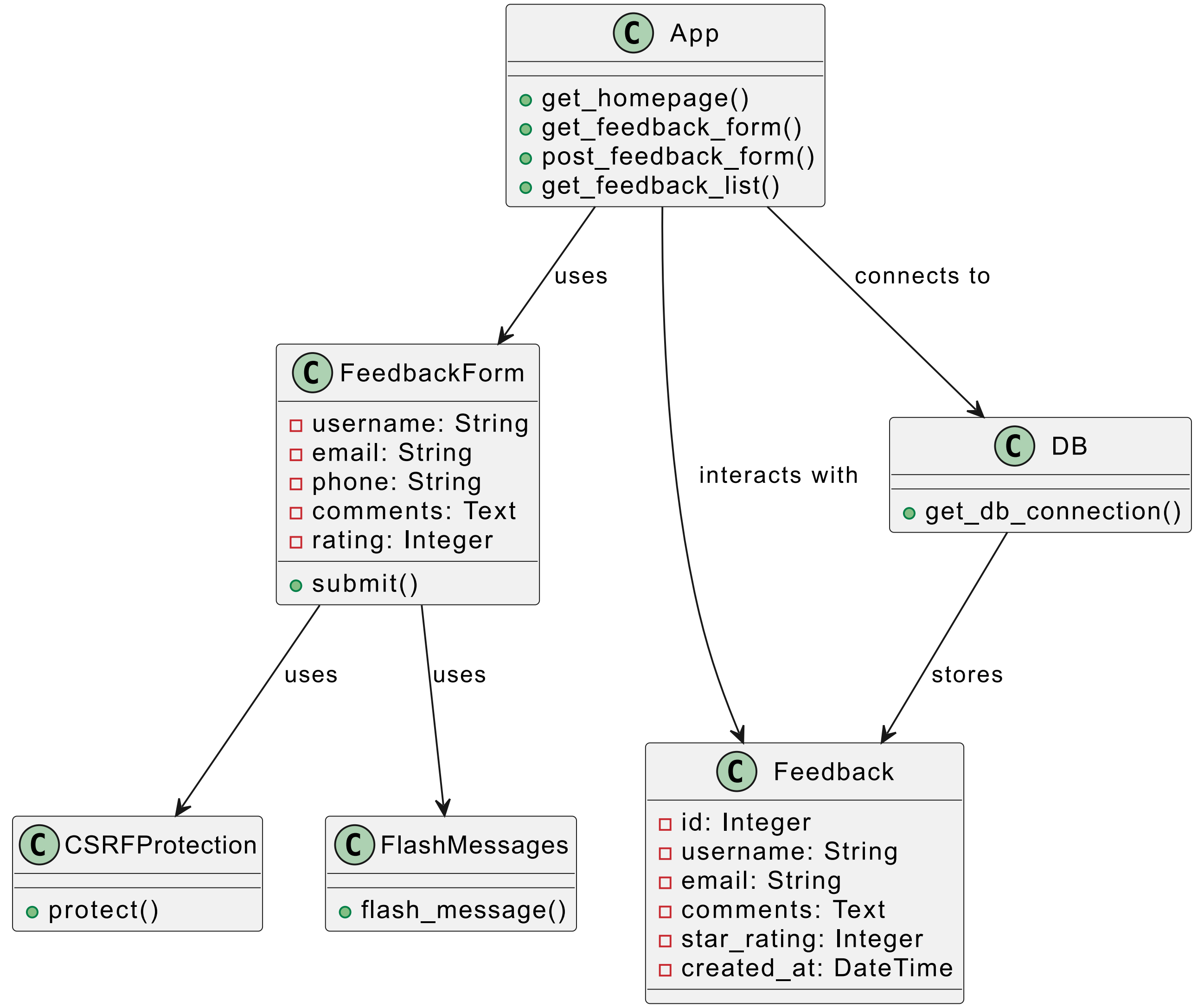
SASVA Project Scan Report

Git URL: https://github.com/MakMohanK/Feeback_App.git | Branch: main

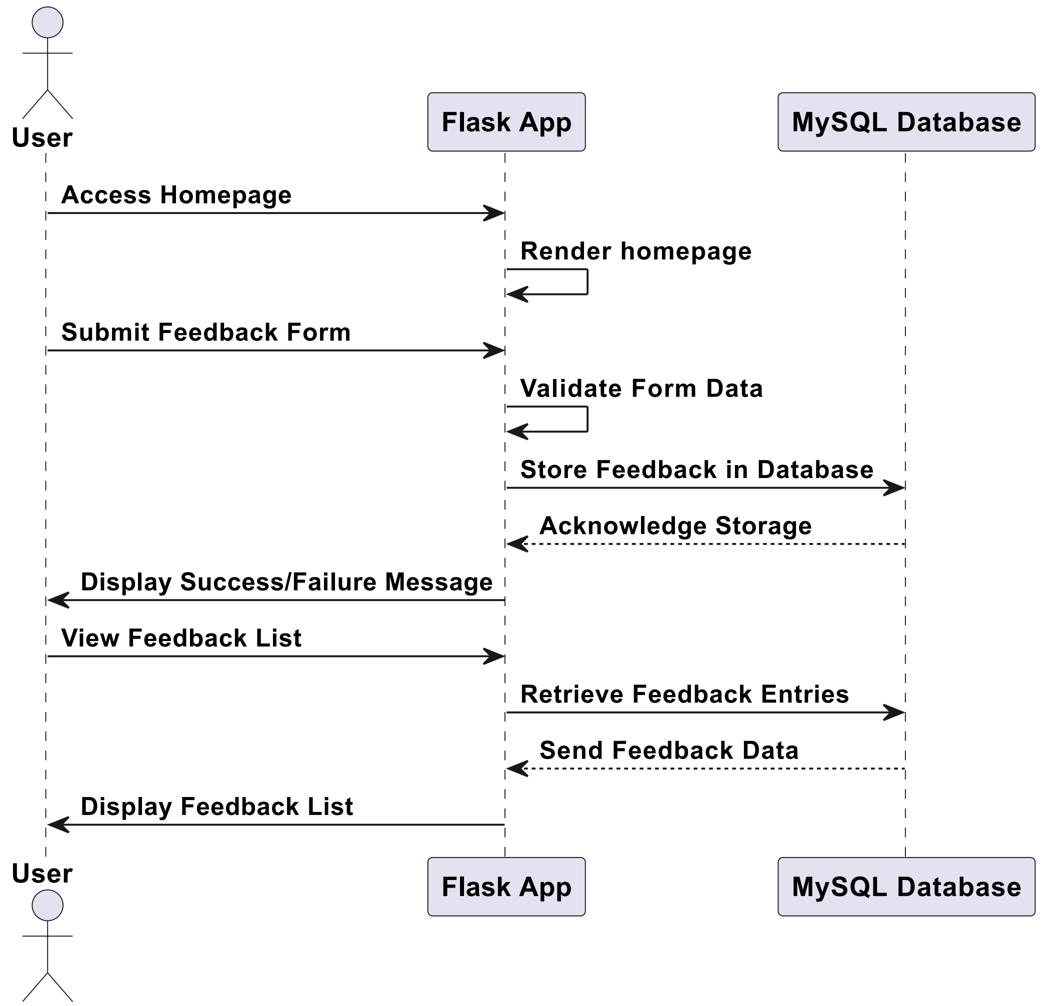
1) Class Diagram:

The class diagram represents the structure of the feedback management system. The `FeedbackForm` class defines the fields and methods for the feedback form, including user input fields and a submit method. The `Feedback` class represents the feedback data stored in the database, with fields for user details and feedback content. The `App` class contains methods for handling different routes in the application, such as displaying the homepage, handling feedback form submissions, and listing feedback entries. The `DB` class provides a method for establishing a database connection. The `CSRFProtection` and `FlashMessages` classes are used by the `FeedbackForm` to ensure security and provide user feedback. The diagram shows the interactions between these components, highlighting how the application processes and stores user feedback.



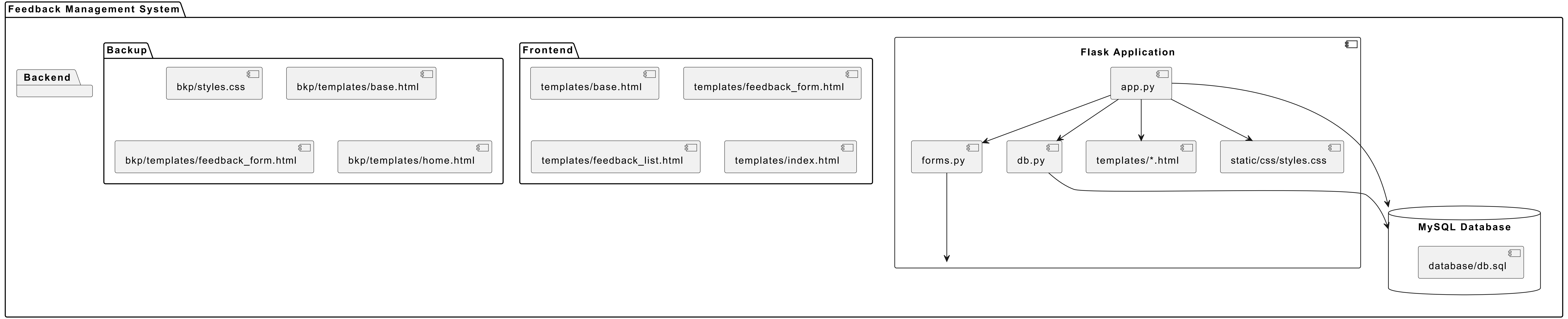
2) Sequence Diagram:

This sequence diagram illustrates the interactions within the web-based feedback management system. The user accesses the homepage, submits feedback through a form, and views the feedback list. The Flask application handles form validation, stores feedback in a MySQL database, and retrieves feedback entries for display. The system provides feedback to the user through success or failure messages.



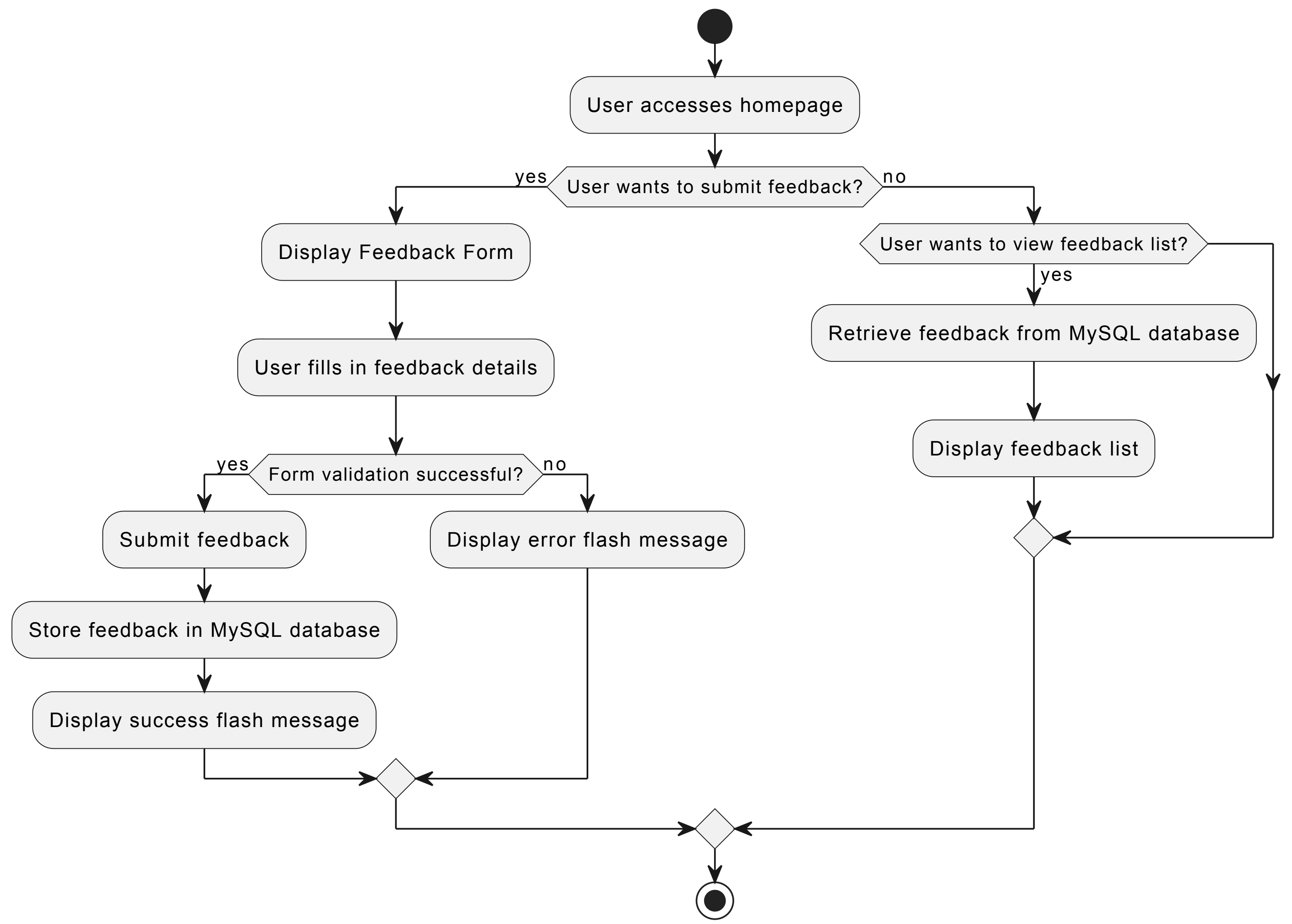
3) Logical Diagram:

The logical diagram represents the structure of the Feedback Management System. It includes the main components: the Flask application, MySQL database, frontend, backend, and backup files. The Flask application consists of 'app.py', which interacts with 'db.py' for database connections and 'forms.py' for form handling. The frontend is composed of HTML templates and CSS for styling, while the backend includes server-side logic and database interaction. The backup section contains older versions of styles and templates. The diagram illustrates the relationships and dependencies between these components, highlighting the flow of data from the frontend to the backend and into the database.



4) Activity Diagram:

The activity diagram illustrates the flow of the web-based feedback management system. Users can either submit feedback or view the feedback list. When submitting feedback, the user fills out a form, which is validated before storing the data in a MySQL database. Success or error messages are displayed based on the validation outcome. Users can also view a list of feedback entries, which are retrieved from the database and displayed in a structured format.


User accesses homepage
User wants to submit feedback?
yes
no
Display Feedback Form
User wants to view feedback list?
yes
Retrieve feedback from MySQL database
User fills in feedback details
Display feedback list
Form validation successful?
yes
no
Submit feedback
Display error flash message
Store feedback in MySQL database
Display success flash message

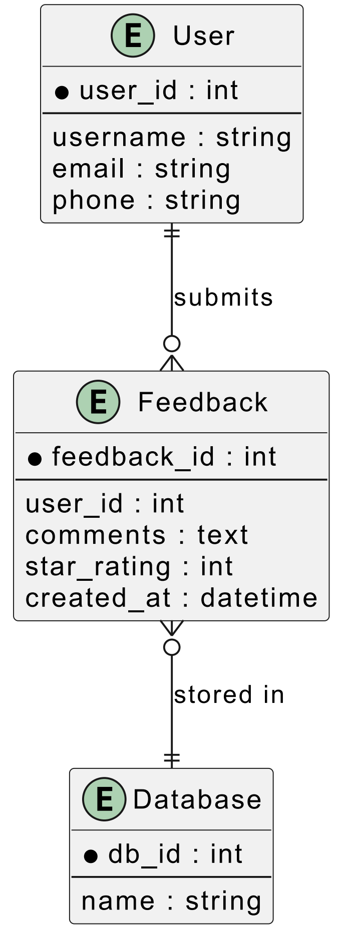
5) Component Diagram:

The component diagram illustrates the architecture of the Feedback Management System. The system is built using Flask, which interacts with a MySQL database for data persistence. The application uses Jinja2 for templating and CSS for styling the frontend. Flask-WTF is used for form handling and validation, while pymysql manages database connections. The backend consists of 'app.py', 'db.py', and 'forms.py', handling server-side logic and database interactions. The frontend includes HTML templates and CSS files for rendering the user interface. The database component is represented by 'database/db.sql', which sets up the database schema. Backup files are stored separately for styles and templates.

Feedback Management System
Flask Application
MySQL Database
Jinja2 Templating
CSS Styling
Flask-WTF
pymysql
Backup
bkp/styles.css
bkp/templates/base.html
bkp/templates/feedback_form.html
bkp/templates/home.html
Database
database/db.sql
Frontend
templates/base.html
templates/feedback_form.html
templates/home.html
templates/feedback_list.html
templates/index.html
static/css/styles.css
Backend
app.py
db.py
forms.py

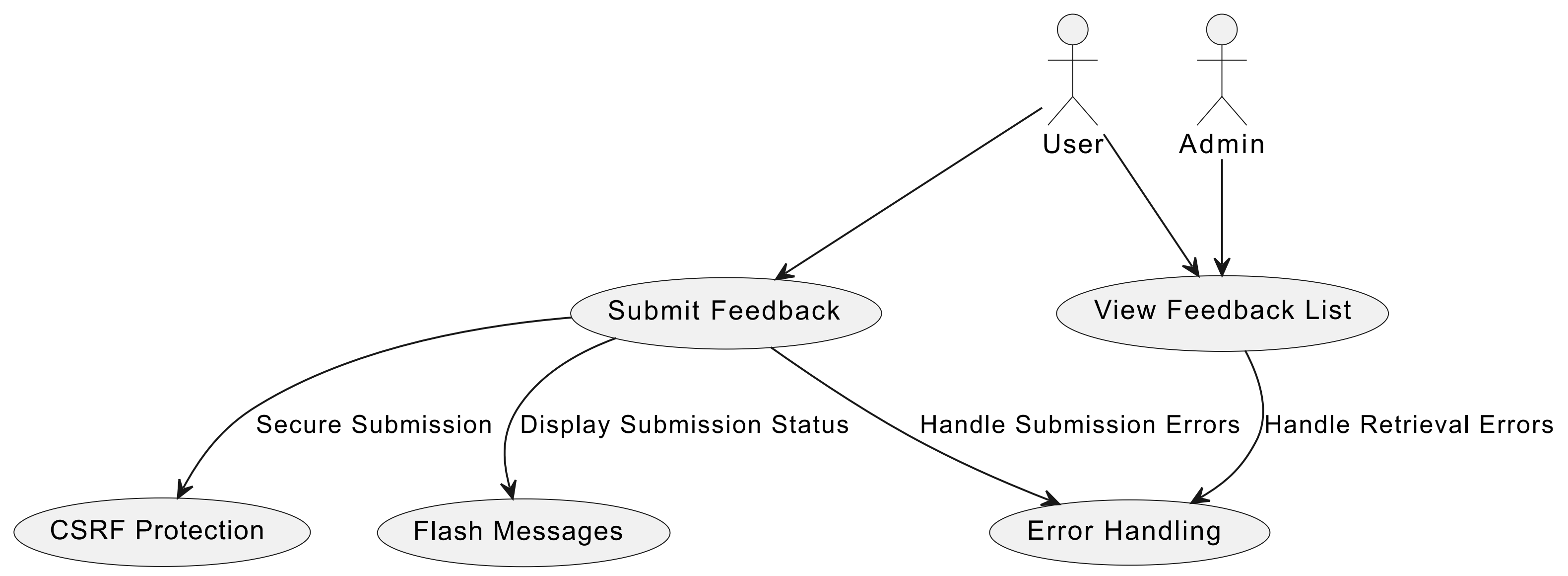
6) ER (Entity-Relationship) Diagram:

The ER diagram represents the relationships between the entities in the feedback management system. The 'User' entity contains user details such as username, email, and phone. The 'Feedback' entity stores user feedback, including comments and a star rating, and is linked to the 'User' entity through a user_id. The 'Feedback' entity is also associated with the 'Database' entity, indicating that feedback is stored in a database. The diagram illustrates how users submit feedback, which is then stored in the database.



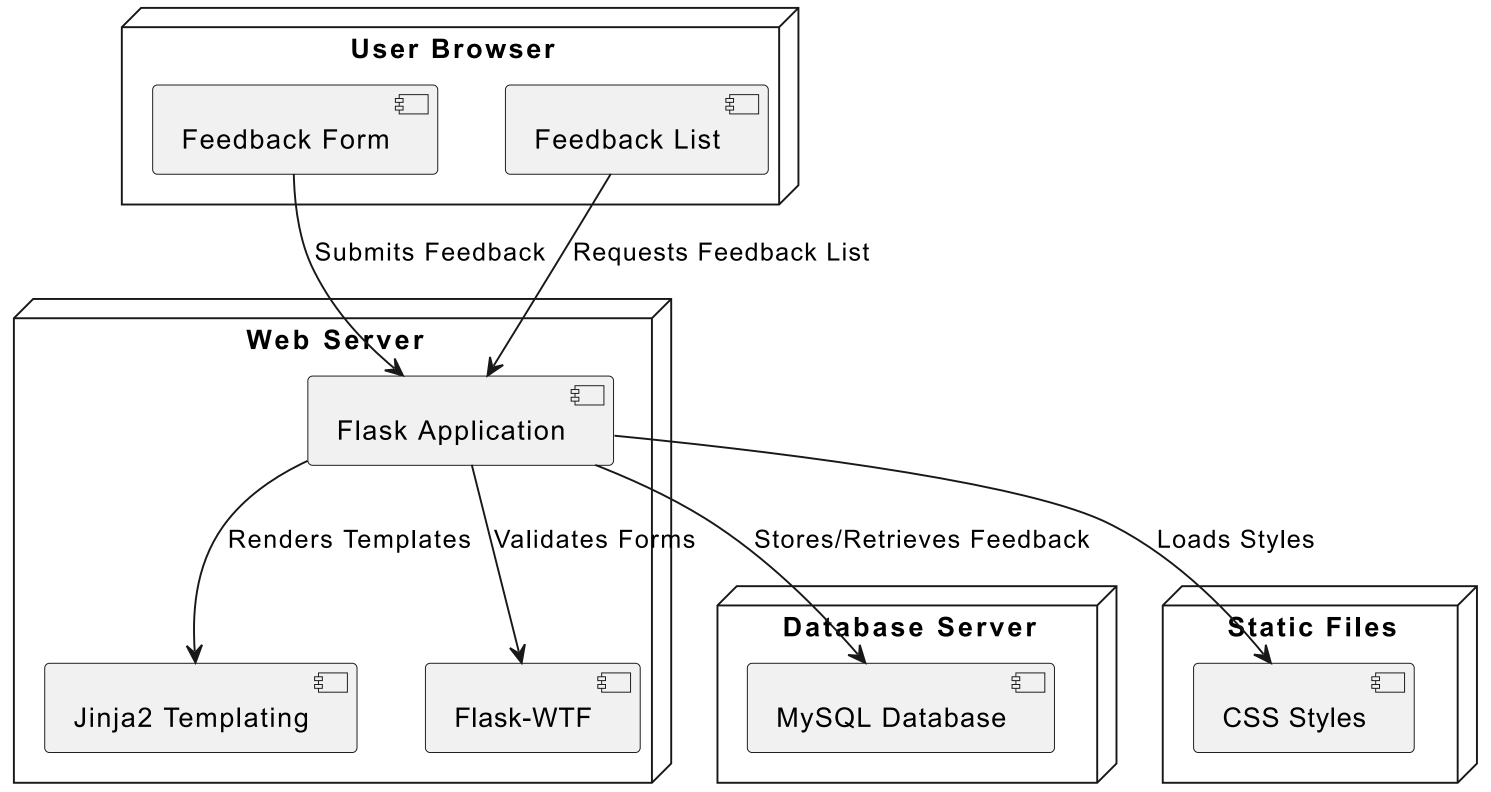
7) Use Case Diagram:

The use case diagram illustrates the interactions between users and the feedback management system. Users can submit feedback and view the feedback list. The system ensures secure submission through CSRF protection and provides feedback via flash messages. Error handling is implemented for both submission and retrieval processes. Admins can also view the feedback list.



8) Deployment Diagram:

The deployment diagram illustrates the architecture of the feedback management system. Users interact with the system through a browser, accessing the feedback form and list. The web server hosts the Flask application, which processes requests, validates forms using Flask-WTF, and renders HTML templates with Jinja2. Feedback data is stored and retrieved from a MySQL database on a separate database server. Static CSS files are used for styling the web pages.



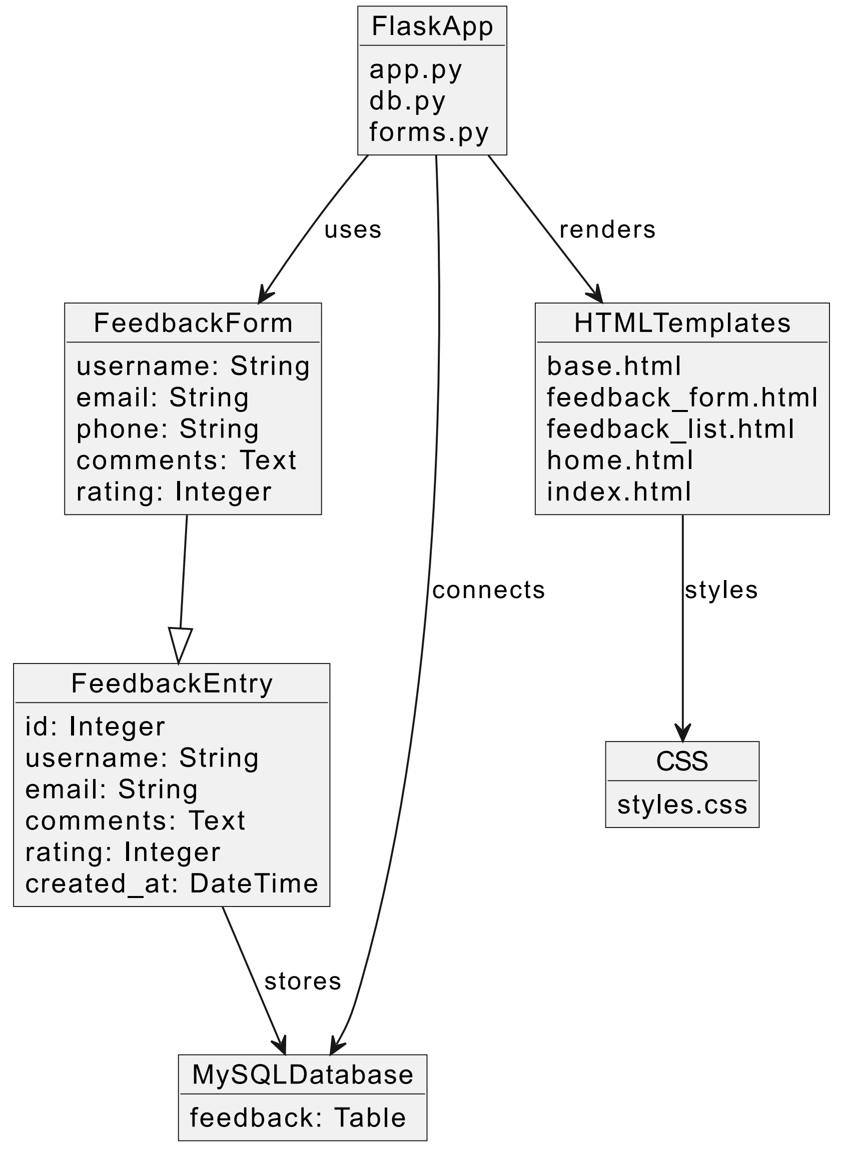
9) Composite Structure Diagram:

The composite structure diagram represents the architecture of the Feedback Management System. It includes the main components such as the Flask App, which interacts with the frontend templates and static CSS for rendering the user interface. The backend consists of Python files for application logic, form handling, and database connectivity. The database package contains SQL scripts for setting up the MySQL database. The backup package holds older versions of styles and templates. The diagram highlights the integration of Flask with Jinja2 for templating, pymysql for database connections, and Flask-WTF for form handling and CSRF protection.

Feedback Management System
Backend
Backup
bkp/styles.css
bkp/templates/base.html
bkp/templates/feedback_form.html
bkp/templates/home.html
Database
database/db.sql
Frontend
templates/base.html
templates/feedback_form.html
templates/home.html
templates/feedback_list.html
templates/index.html
Flask App
app.py
forms.py
db.py
templates/*.html
static/css/styles.css
pymysql
MySQL Database
Jinja2 Templating
Flask-WTF
CSRF Protection
Error Handling

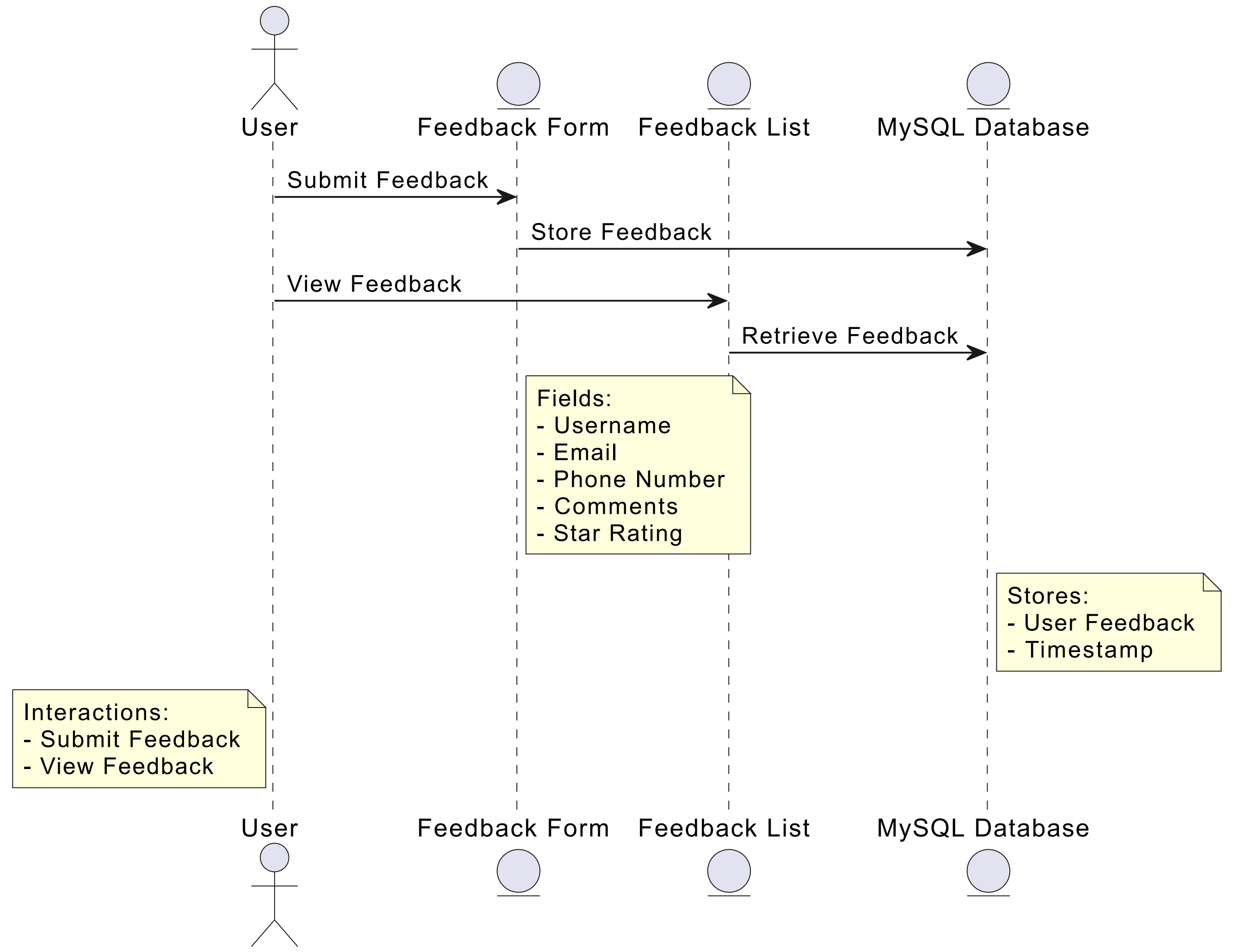
10) Object Diagram:

The object diagram represents the key components of the feedback management system. The `FeedbackForm` object captures user input, which is then validated and stored as `FeedbackEntry` in the `MySQLDatabase`. The `FlaskApp` object, consisting of `app.py`, `db.py`, and `forms.py`, manages the application logic, database connections, and form handling. It uses `HTMLTemplates` for rendering web pages and `CSS` for styling. The `FeedbackEntry` is stored in the `feedback` table within the `MySQLDatabase`. The diagram illustrates the relationships and interactions between these components.



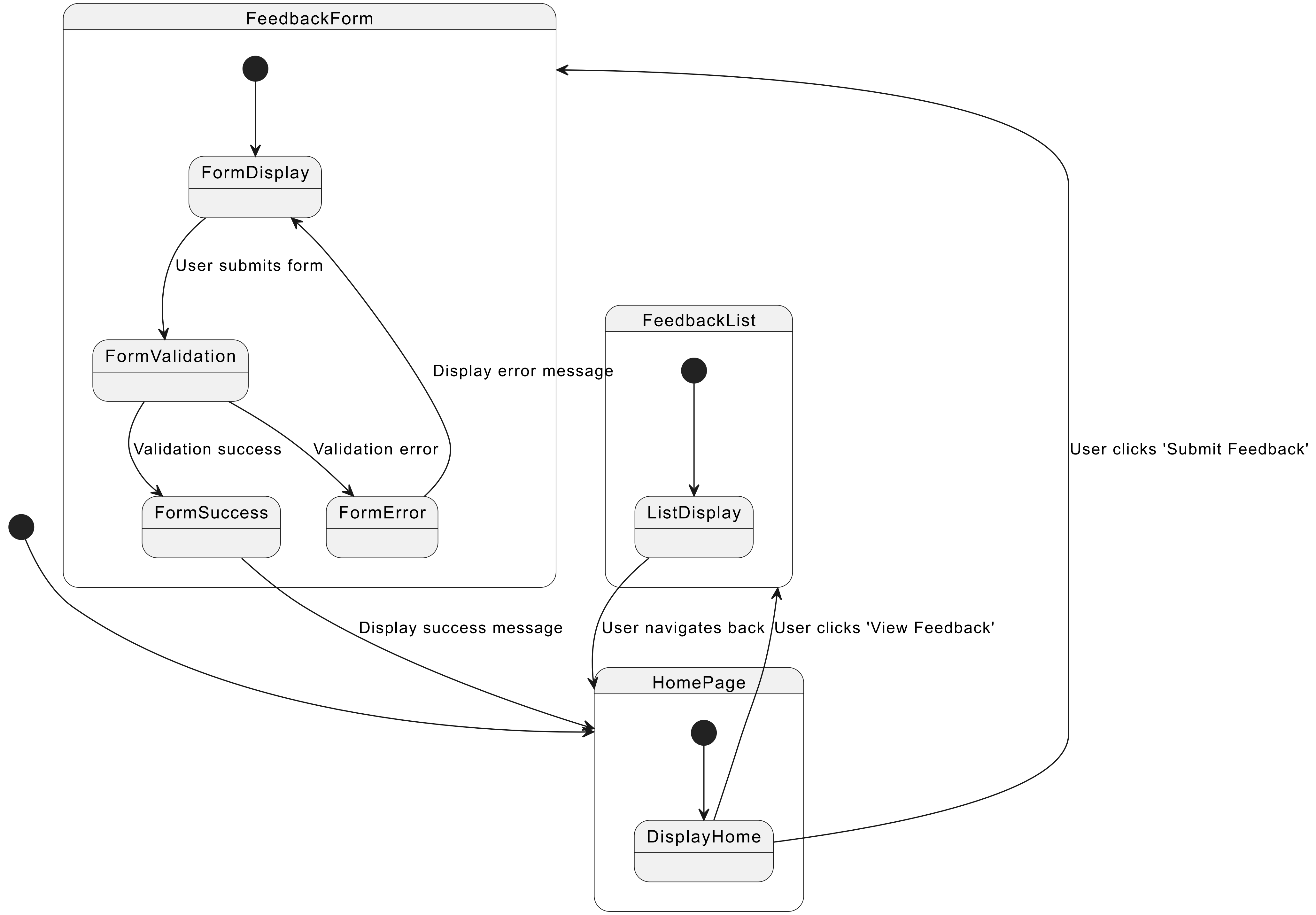
11) User Journey Map:

The user journey map illustrates the interactions between the user and the feedback management system. Users can submit feedback through a form, which is then stored in a MySQL database. They can also view submitted feedback entries, which are retrieved from the database. The diagram highlights the key components: the feedback form, feedback list, and database, along with the flow of data between them.



12) State Diagram:

The state diagram represents the user interaction flow within the feedback management system. It starts at the HomePage, where users can either navigate to the FeedbackForm to submit feedback or to the FeedbackList to view existing feedback. In the FeedbackForm state, users fill out and submit the form, which undergoes validation. Successful validation leads to a success message and a return to the HomePage, while errors prompt users to correct their input. The FeedbackList state allows users to view feedback entries and return to the HomePage.



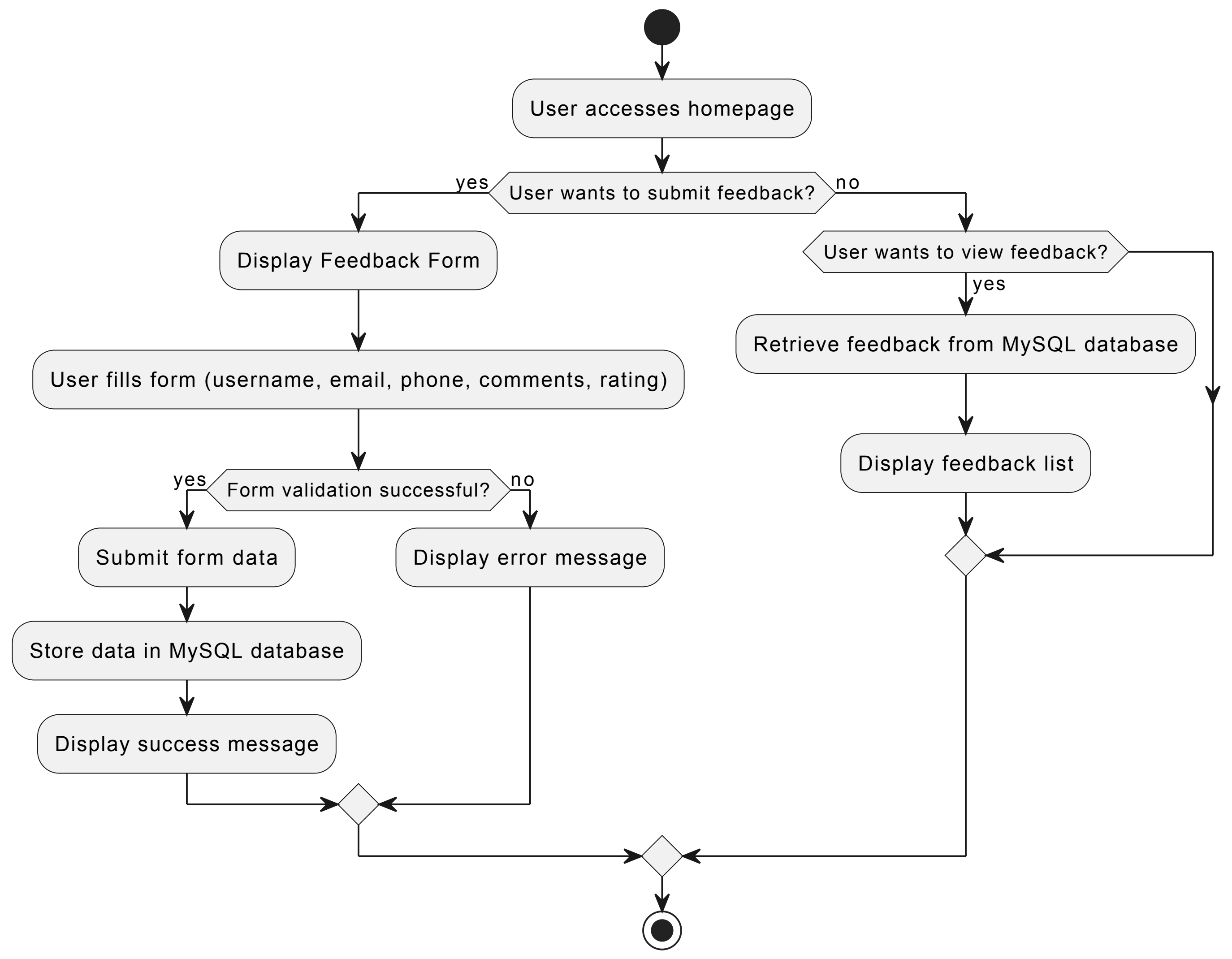
13) Architecture Diagram:

The architecture diagram represents a web-based feedback management system built using Flask. The system is divided into frontend and backend components. The backend, implemented in Python, includes 'app.py' for application logic, 'db.py' for database connections, and 'forms.py' for form handling. The frontend consists of HTML templates and CSS for styling. The application interacts with a MySQL database to store and retrieve feedback data. Key features include a feedback form, feedback list, and homepage, with routes managed by Flask. The system ensures security with CSRF protection and error handling.

Feedback Management System
Backend
Backup
bkp/styles.css
bkp/templates/base.html
bkp/templates/feedback_form.html
bkp/templates/home.html
Frontend
templates/base.html
templates/feedback_form.html
templates/home.html
templates/feedback_list.html
templates/index.html
Flask App
app.py
GET /
GET/POST /feedback_form
forms.py
db.py
templates/*.html
static/css/styles.css
Homepage
Feedback Form
pymysql
GET /feedback_list
Store Feedback
MySQL Database
database/db.sql
Retrieve Feedback
Feedback List

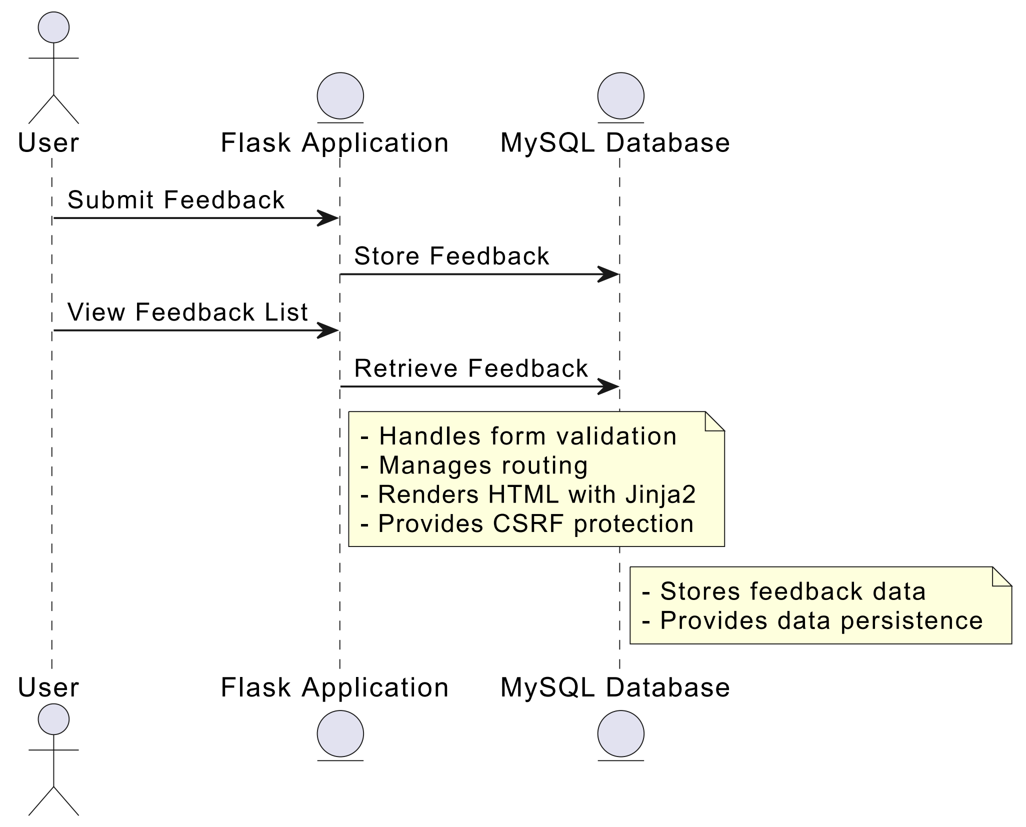
14) Workflow Diagram:

The workflow diagram illustrates the process flow of the feedback management system. Users can either submit feedback or view existing feedback. When submitting feedback, users fill out a form, which is validated before storing the data in a MySQL database. If validation fails, an error message is shown. Users can also view a list of feedback entries retrieved from the database.


User accesses homepage
User wants to submit feedback?
yes
no
Display Feedback Form
User fills form (username, email, phone, comments, rating)
Form validation successful?
yes
no
Submit form data
Display error message
Store data in MySQL database
Display success message
User wants to view feedback?
yes
Retrieve feedback from MySQL database
Display feedback list

15) Communication Diagram:

The communication diagram illustrates the interactions between the user, the Flask application, and the MySQL database in the feedback management system. Users submit feedback through the Flask application, which validates and stores the data in the MySQL database. Users can also view a list of feedback, which the application retrieves from the database. The diagram highlights the application's role in managing form validation, routing, and security features like CSRF protection, while the database ensures data persistence.


User
Flask Application
MySQL Database
Submit Feedback
Store Feedback
View Feedback List
Retrieve Feedback
- Handles form validation
- Manages routing
- Renders HTML with Jinja2
- Provides CSRF protection
- Stores feedback data
- Provides data persistence
User
Flask Application
MySQL Database