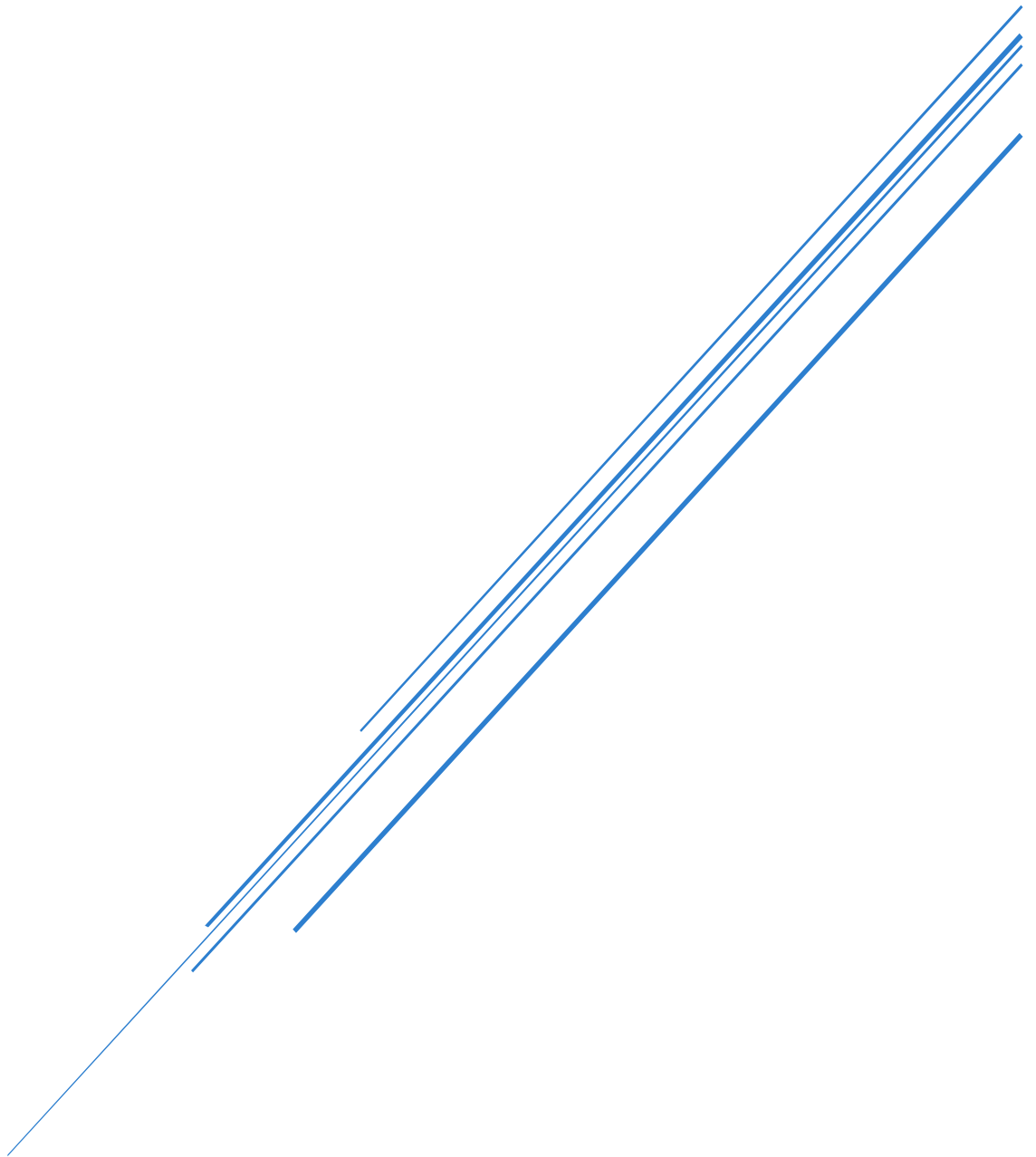


EVENT MANAGEMENT SYSTEM

Phase 5



CSAI
Software Engineering

Team Contributions:-

- **Zeyad: Was mainly responsible for the core backend and system foundation throughout the project.**
 - He implemented the authentication functionality, including login and signup, along with the required controllers and repositories.
 - He also worked on the base layout and user interface styling, including the base template and CSS files.
 - In addition, Ziad handled feedback functionality, database connection logic, security utilities, and application configuration files.
 - He implemented the main application setup, database initialization, and utility modules.
 - He was also responsible for writing multiple unit tests, preparing the Dockerfile, initializing the database, managing dependencies, and creating the final README file.
- **Sara: Worked mainly on event registration functionality and testing.**
 - She implemented the event registration logic, including its controller and repository.
 - She also worked on the event details page and ensured correct interaction between the frontend and backend.
 - In Phase 5, she focused on writing unit tests related to registration and other system components.
 - Additionally, she prepared the testing report documenting the testing process and results.
- **Retaj: Was responsible for event management features and documentation.**
 - She implemented event-related functionalities, including event creation, editing, listing, and organizer-specific views.
 - She also worked on the main dashboard and multiple event-related pages.
 - In addition to development tasks, she handled all documentation work, including User Documentation, Technical Documentation, and the Phase 5 Report.
 - She also prepared the final presentation slides and organized the demo flow for the project.

1. Phase 5 Requirements:-

Phase 5 focused on:

- Testing
- Documentation
- Docker deployment
- Final presentation and demo

2. Testing Summary:-

- Unit tests were implemented for repository and authentication logic.
- Integration testing was used to verify routing functionality.
- All tests passed successfully.

3. Docker Summary:-

- Dockerfile was created to containerize the application.
- The system runs inside a Docker container using Flask.
- Docker ensures environment consistency.

4. Demo Description:-

The demo demonstrates:

- User registration and login
- Event browsing and registration
- Feedback submission
- Running the application using Docker

5. Challenges & Solutions:-

Challenges:

- Database handling in Docker
- Git merge conflicts
- Flask application context during testing

Solutions:

- Proper use of Flask app context
- Git feature branches
- Clear project structure

6. Conclusion:-

The Event Management System successfully meets all requirements of Phase 5. The project demonstrates practical application of software engineering concepts taught in the course.

7. Screenshots and Visual Evidence:-

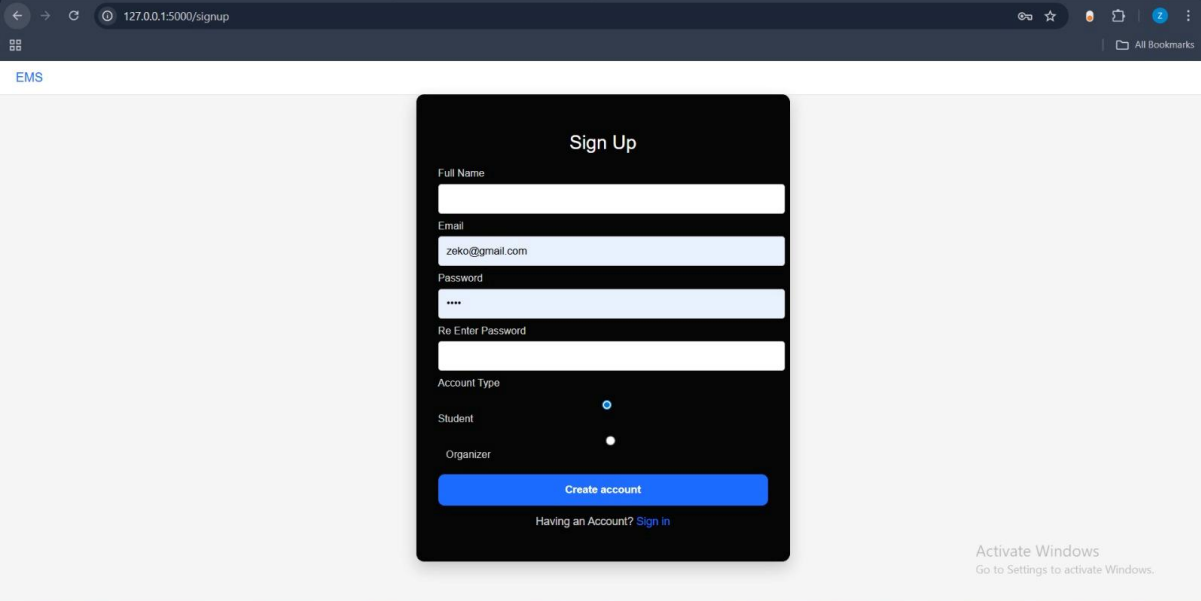
This section provides visual evidence demonstrating the successful implementation of the Event Management System and Phase 5 requirements.

The screenshots illustrate the main system features, user interactions, and Docker execution.

7.1 User Authentication Screenshots:

1. Signup Page

This screenshot shows the user registration page where new users can create an account by providing their basic information.



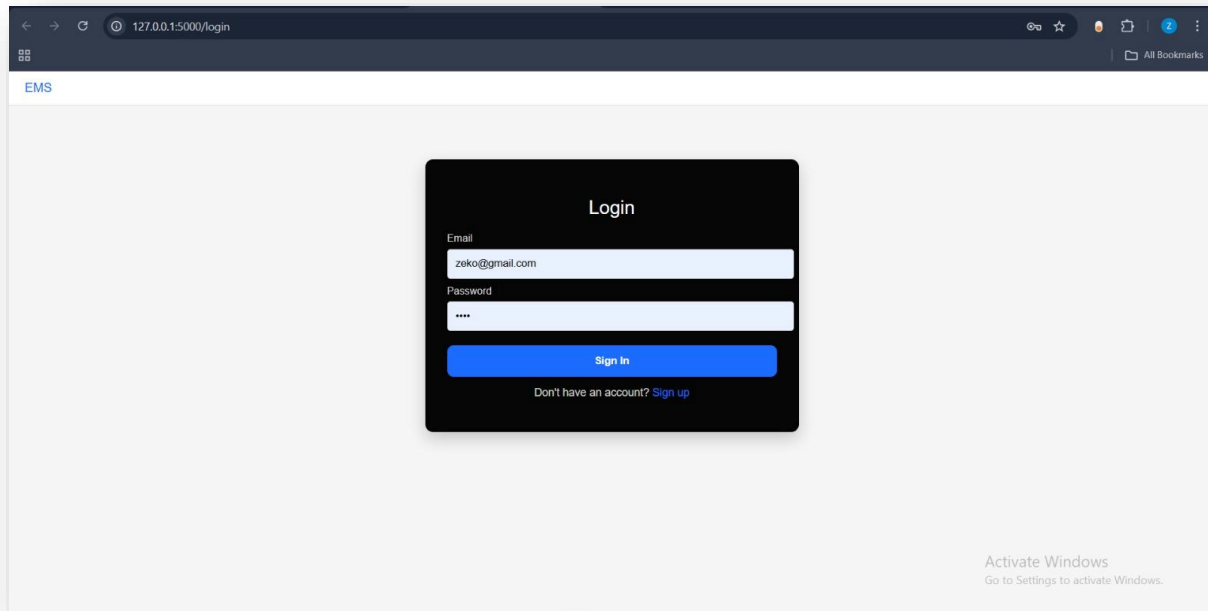
The screenshot displays a web browser window with the address bar showing "127.0.0.1:5000/signup". The page title is "EMS". The main content is a "Sign Up" form with the following fields and options:

- Full Name:
- Email:
- Password:
- Re Enter Password:
- Account Type: Radio buttons for "Student" (selected) and "Organizer".
- Buttons: "Create account" (blue) and "Having an Account? Sign in" (link).

An "Activate Windows" watermark is visible in the bottom right corner of the browser window.

2. Login Page

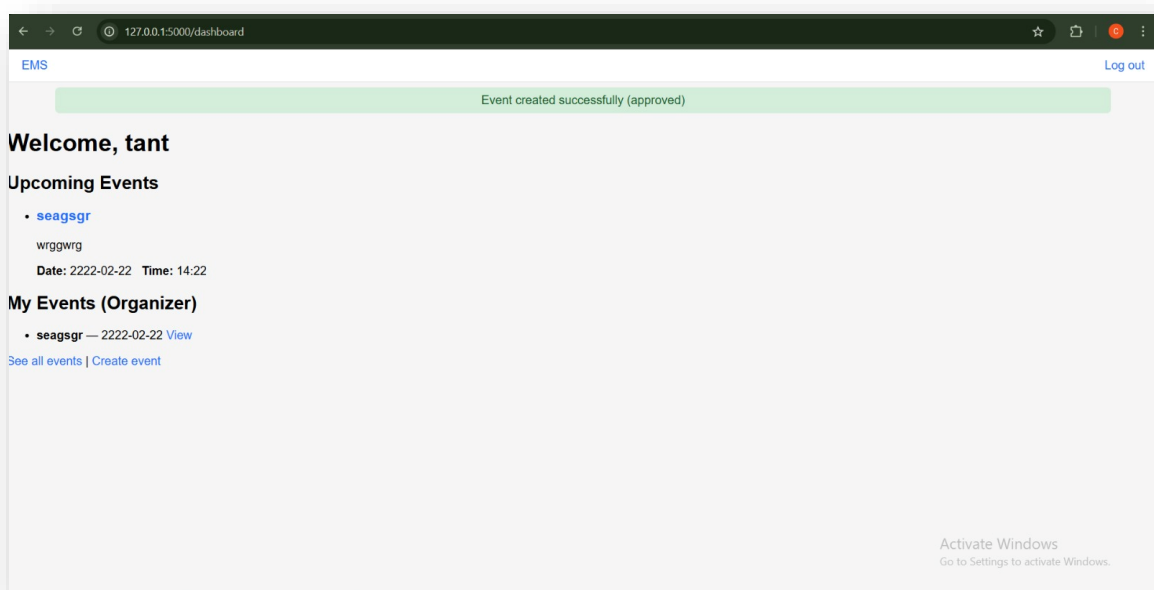
This screenshot shows the login page where registered users can enter their credentials to access the system.



7.2 Dashboard and Event Browsing:

- Event Dashboard

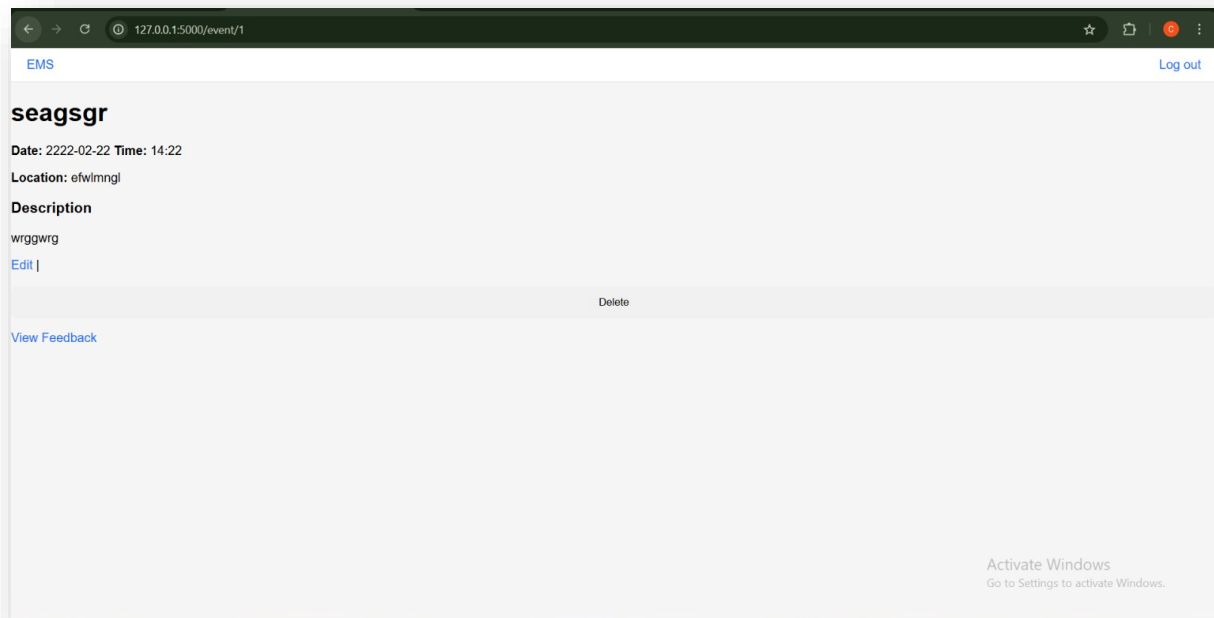
This screenshot shows the main dashboard displaying available events after the user logs in successfully.



7.3 Event Details and Registration:

- Event Details Page

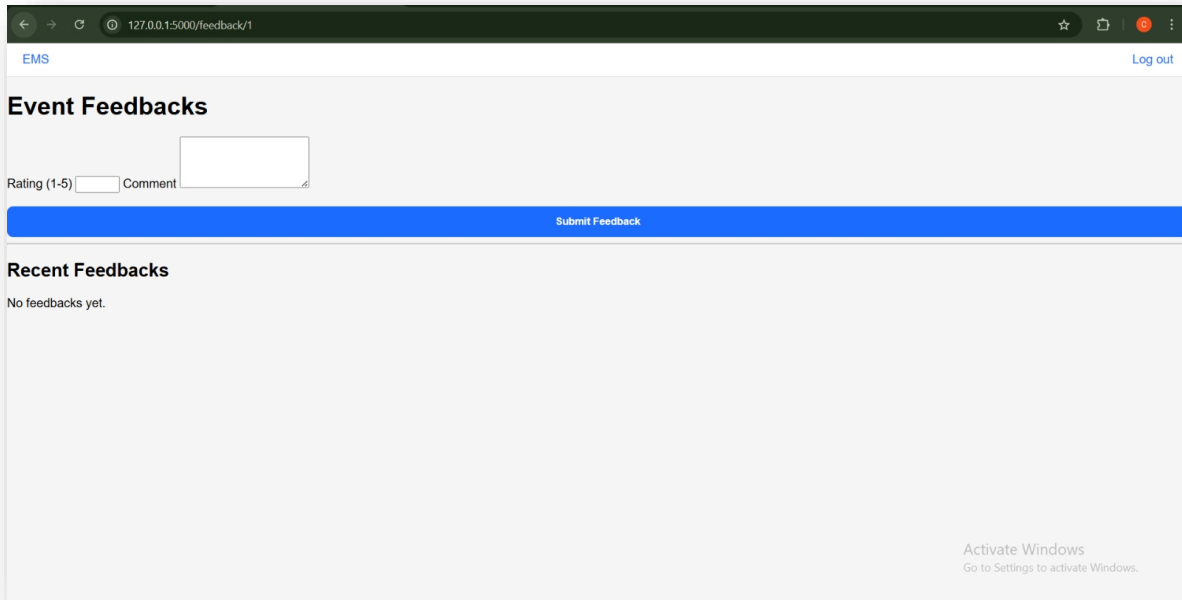
This screenshot shows the event details page, including event information and the option to register for the event.



7.4 Feedback Submission:

- Feedback Page

This screenshot demonstrates the feedback submission feature, where users can rate events and add comments.

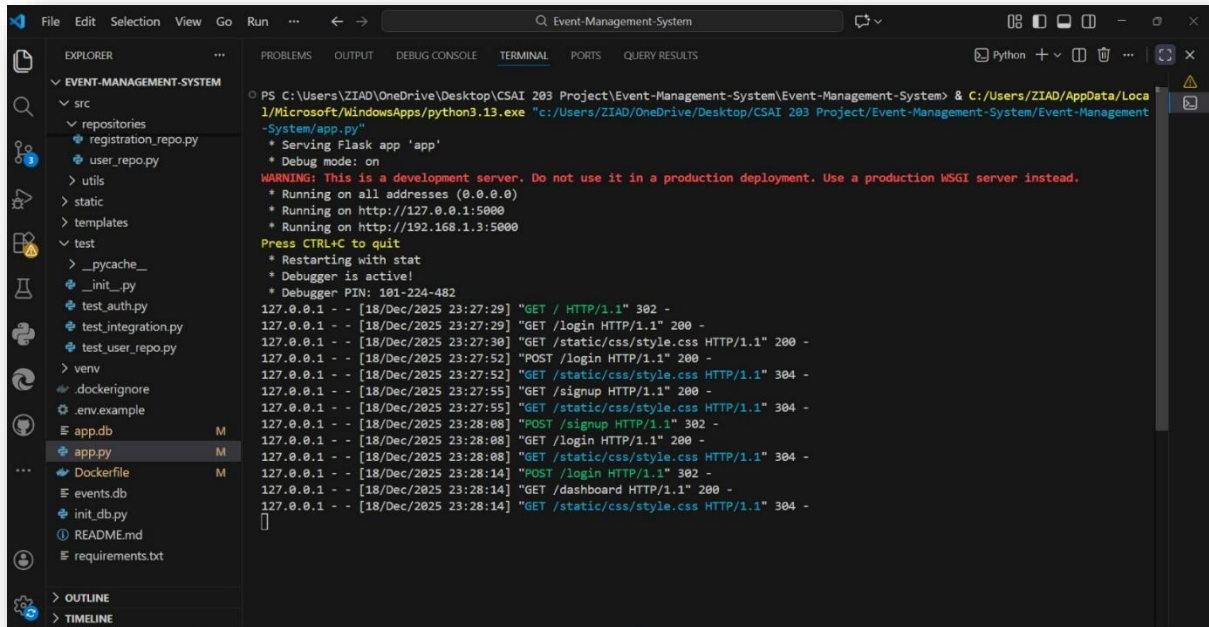


The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5000/feedback/1". The page title is "EMS" and there is a "Log out" link in the top right corner. The main heading is "Event Feedbacks". Below this, there is a form with a "Rating (1-5)" label and a text input field, followed by a "Comment" label and a text area. A blue button labeled "Submit Feedback" is positioned below the form. Under the heading "Recent Feedbacks", it says "No feedbacks yet." At the bottom right, there is a watermark that says "Activate Windows Go to Settings to activate Windows."

7.5 Docker Execution:

- Application Running with Docker

This screenshot shows the application running successfully inside a Docker container, confirming Docker integration in Phase 5.



The screenshot displays a Visual Studio Code (VS Code) interface with a terminal window open. The terminal shows the command prompt of a PowerShell session running inside a Docker container. The command executed is `python3.13.exe "c:/Users/ZIAD/OneDrive/Desktop/CSAI 203 Project/Event-Management-System/Event-Management-System/app.py"`. The output indicates that the Flask application is running successfully on all addresses (0.0.0.0) and on port 5000. The terminal also shows a warning message: "WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead." and a message stating "Press CTRL+C to quit". The terminal output shows several HTTP requests and responses, including GET requests for /, /login, /static/css/style.css, and POST requests for /login and /signup. The terminal also shows the status of the application, including "Restarting with stat", "Debugger is active!", and "Debugger PIN: 101-224-482". The VS Code Explorer on the left shows the file structure of the project, including `src`, `repositories`, `utils`, `static`, `templates`, `test`, `__pycache__`, `__init__.py`, `test_auth.py`, `test_integration.py`, `test_user_repo.py`, `venv`, `.dockerignore`, `.env.example`, `app.db`, `app.py`, `Dockerfile`, `events.db`, `init_db.py`, `README.md`, and `requirements.txt`.

```
PS C:\Users\ZIAD\OneDrive\Desktop\CSAI 203 Project\Event-Management-System> & C:/Users/ZIAD/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/ZIAD/OneDrive/Desktop/CSAI 203 Project/Event-Management-System/Event-Management-System/app.py"
-System/app.py"
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://192.168.1.3:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 101-224-482
127.0.0.1 - - [18/Dec/2025 23:27:29] "GET / HTTP/1.1" 302 -
127.0.0.1 - - [18/Dec/2025 23:27:29] "GET /login HTTP/1.1" 200 -
127.0.0.1 - - [18/Dec/2025 23:27:30] "GET /static/css/style.css HTTP/1.1" 200 -
127.0.0.1 - - [18/Dec/2025 23:27:52] "POST /login HTTP/1.1" 200 -
127.0.0.1 - - [18/Dec/2025 23:27:52] "GET /static/css/style.css HTTP/1.1" 304 -
127.0.0.1 - - [18/Dec/2025 23:27:55] "GET /signup HTTP/1.1" 200 -
127.0.0.1 - - [18/Dec/2025 23:27:55] "GET /static/css/style.css HTTP/1.1" 304 -
127.0.0.1 - - [18/Dec/2025 23:28:08] "POST /signup HTTP/1.1" 302 -
127.0.0.1 - - [18/Dec/2025 23:28:08] "GET /login HTTP/1.1" 200 -
127.0.0.1 - - [18/Dec/2025 23:28:08] "GET /static/css/style.css HTTP/1.1" 304 -
127.0.0.1 - - [18/Dec/2025 23:28:14] "POST /login HTTP/1.1" 302 -
127.0.0.1 - - [18/Dec/2025 23:28:14] "GET /dashboard HTTP/1.1" 200 -
127.0.0.1 - - [18/Dec/2025 23:28:14] "GET /static/css/style.css HTTP/1.1" 304 -
```