Report: Build a Django Middleware to Store Logs and Their Error Status with Graph

**1. Task Description**

The task involves creating a Django middleware component that captures all incoming HTTP requests and responses, storing logs related to each request, including error statuses. This middleware will also be responsible for analysing the logged data and presenting it visually in the form of graphs.

The middleware should:

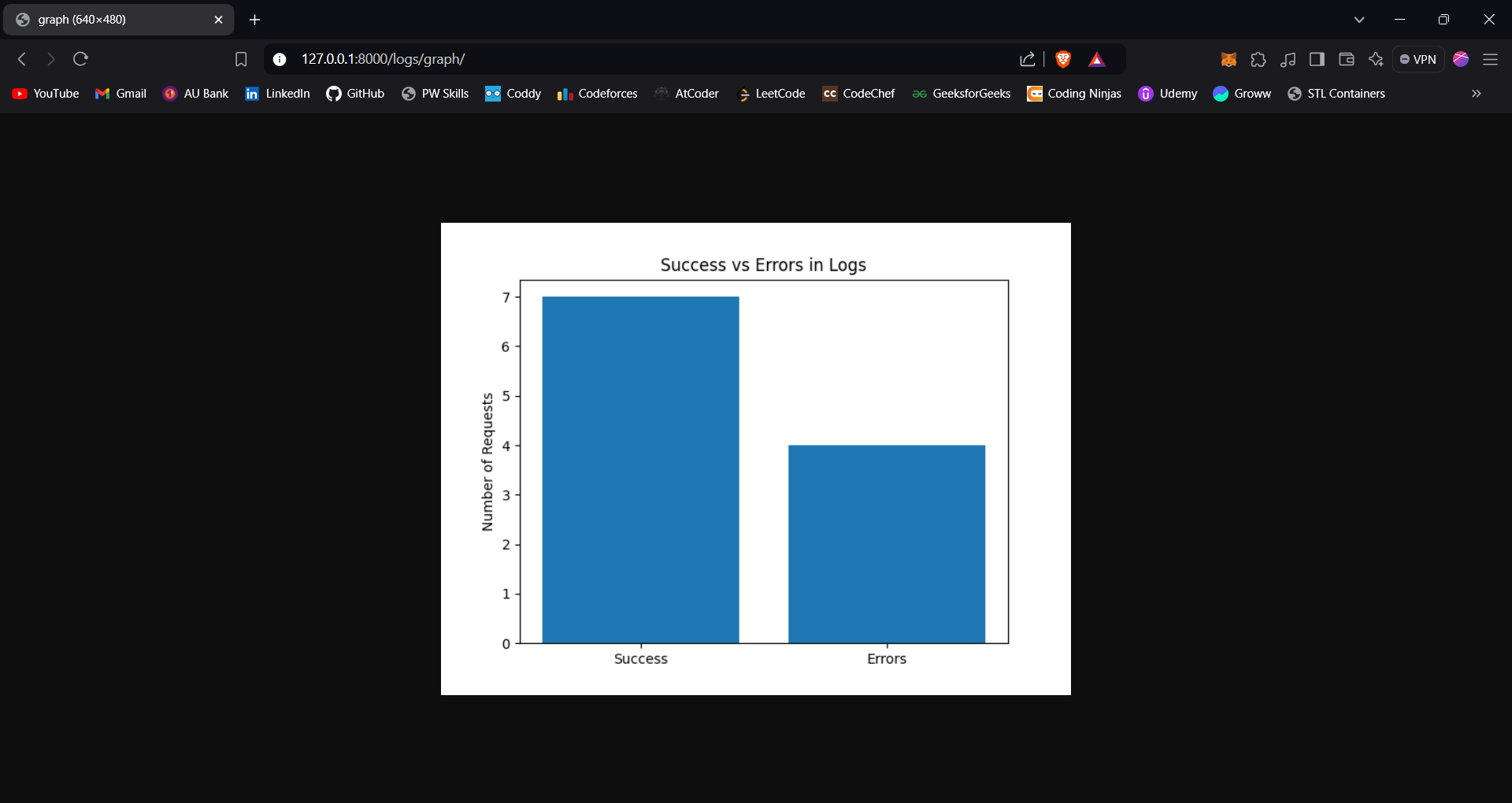
- Intercept requests and responses.

- Log the request method, URL, status code, and timestamp.

- Store this data in a database.

- Provide an interface to retrieve the logs and display them in graphical form using a visualization library.

**2. Output Screenshot**



**3. Widget/Algorithm Used in Task**

The task employs the following key components:

**Middleware Implementation**

- **Django Middleware**: The middleware is implemented by creating a new class that inherits from MiddlewareMixin . It overrides the \_\_call\_\_ method to log the request and response data.

- **Logging**: Python's built-in logging library is utilized to handle logging, which can be configured to output to both the console and a log file.

**Data Storage**

- **Database Models**: A Django model is created to store log entries, capturing details such as request method, URL, status code, and timestamp.

- **Database Migrations**: Migrations are run to create the necessary tables in the database.

**Data Visualization**

- **Graphing Library**: A Python library like Matplotlib is used to visualize the logged data. The chosen library is integrated into the Django templates to render the graphs.

**Conclusion**

The middleware effectively logs all HTTP requests and their statuses, providing valuable insights into application behaviour through visual representations. This functionality is crucial for monitoring and debugging Django applications.