Project Challenges & Solutions Documentation

Project: Secure Login System using Flask & MongoDB

1. User Authentication & Session Management

Problem:
Initially, after login, the navigation bar was still showing `Login` and `Register` links instead of `Dashboard` and `Logout`. Session data was not being detected correctly inside templates.

Cause:
`session` object was not properly passed to templates or was undefined in some files.

Solution:
- Used Flask's `session` properly (`from flask import session`).
- Updated `base.html` to check session variable with Jinja2.

Solution Screenshot:

- Ensured session is set at login using `session ['user'].

```
session["user"] = {"username": user["username"], "role": user["role"]}
```

2. Account Locking After Failed Attempts

Problem:
Users were supposed to be locked for 1 minute after multiple failed login attempts, but they could still log in immediately.

Cause:
Datetime comparison error: `TypeError: can't compare offset-naive and offset-aware datetimes`.

Solution:
- Standardized all datetime objects to timezone-aware (`datetime.now(timezone.utc)`).
- Saved `locked_until` in MongoDB as UTC and compared correctly.

Solution Screenshot:

```
from datetime import datetime, timezone
now = datetime.now(timezone.utc)
```

3. Form Validations (Client-side + Server-side)

Problem:
Client-side validations (like empty fields, weak password) sometimes didn't match with backend checks, leading to inconsistent results.

Solution:
- Added `required`, `minlength`, and proper input types (`email`, `password`) in HTML.
- Added backend checks for security (regex for email, strong password check).
- Created a reusable `is_strong_password` function with regex ensuring uppercase, lowercase, digit, and special character.

5. Template Errors

Problem:
`jinja2.exceptions.TemplateAssertionError: block 'content' defined twice` appeared.

Cause:
Multiple templates extended incorrectly with duplicate `{% block content %}`.

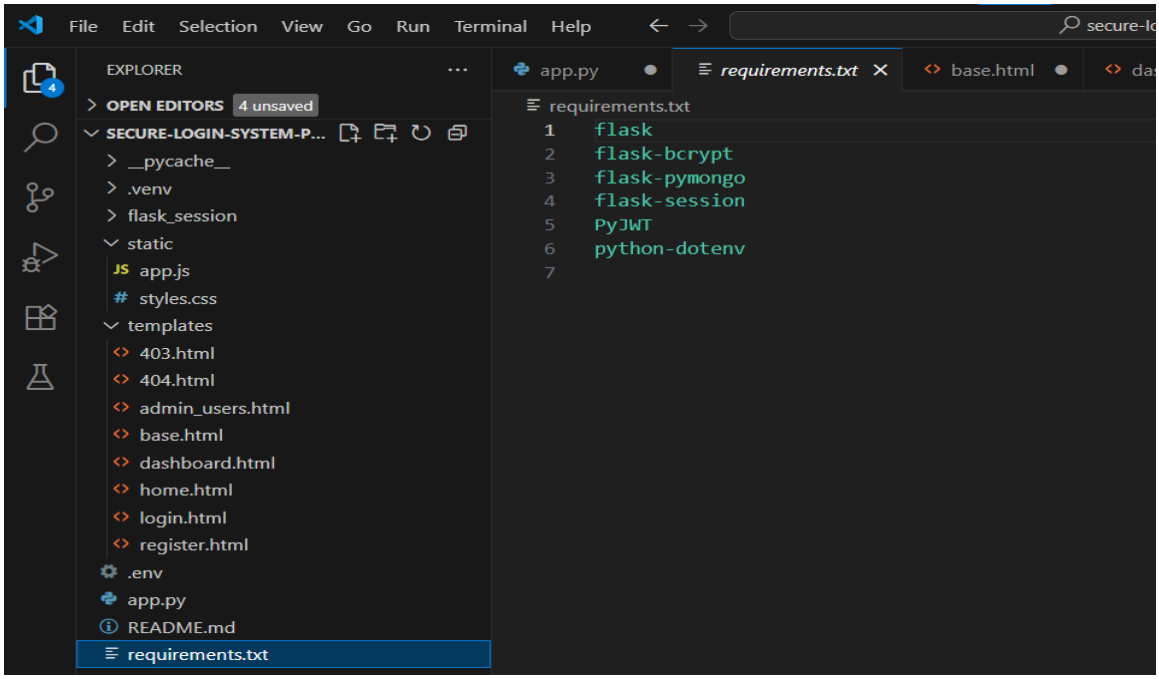
Solution:
- Cleaned template inheritance.
- Ensured only `base.html` defines the block, and other pages extend it.

6. Environment & Dependencies

Problem:
Some errors occurred due to missing or mismatched dependencies (Flask, PyJWT, Flask-Bcrypt, Flask-PyMongo).

Solution:
- Created a `requirements.txt` file.

Solution Screenshot:



- Installed all dependencies with `pip install -r requirements.txt`.

7. Flash Messages Not Showing Consistently

Problem:
Validation errors were not always visible after redirect.

Cause:
Redirection removed flashed messages.

Solution:
- Rendered `register.html` directly with errors instead of redirecting on validation failures.
- Used `with_categories=True` in Jinja template to style messages.

Final Outcome

- Secure login & registration with hashed passwords.
- Role-based access (Admin/User).
- Account locking after failed attempts.
- Full form validation both client & server-side.
- Dynamic navigation bar based on login state.
- Clean UI with proper error handling.