CIS234A: Real World Programming

Project: Food Pantry Notification System – Sprint 1

Name: Khaylub Thompson-Calvin

Date: 4/14/2025

Assignment 1: Project Introduction

Food Pantry Notification System

1. **Formulate clarifying questions you'd like answered for the 4 Sprint 1 stories.**

Part 1: Clarifying Questions for Sprint Stories

**Login / Account Creation (Subscriber Account Creation & Login)**

1. Should users be able to log in using either username or email, or just one?

2. What level of password complexity is required (symbols, length)?

3. How should session management be handled after login (Flask-Session)?

4. What happens if a duplicate username or email is submitted?

**Send Notification (Send New Notification to Subscribers)**

1. Should notifications support HTML formatting (bold text, clickable links)?

2. Can staff send messages to a subset of subscribers (by group or tag)?

3. Can notifications be scheduled for future delivery?

4. What is the expected length limit for the notification subject/body?

**Template Creation (Create and Save Notification Templates)**

1. What placeholders will be supported in templates (e.g., `{{}}`, `{{}}`)?

2. Should templates be pre-approved, or can anyone on staff create one?

3. Can templates be edited after being saved or only deleted and re-created?

4. Will there be categories or tags to organize templates?

**Review Log (Review Notification Log)**

1. Should logs include failed delivery attempts and error details?

2. How long should notification logs be retained before deletion (if at all)?

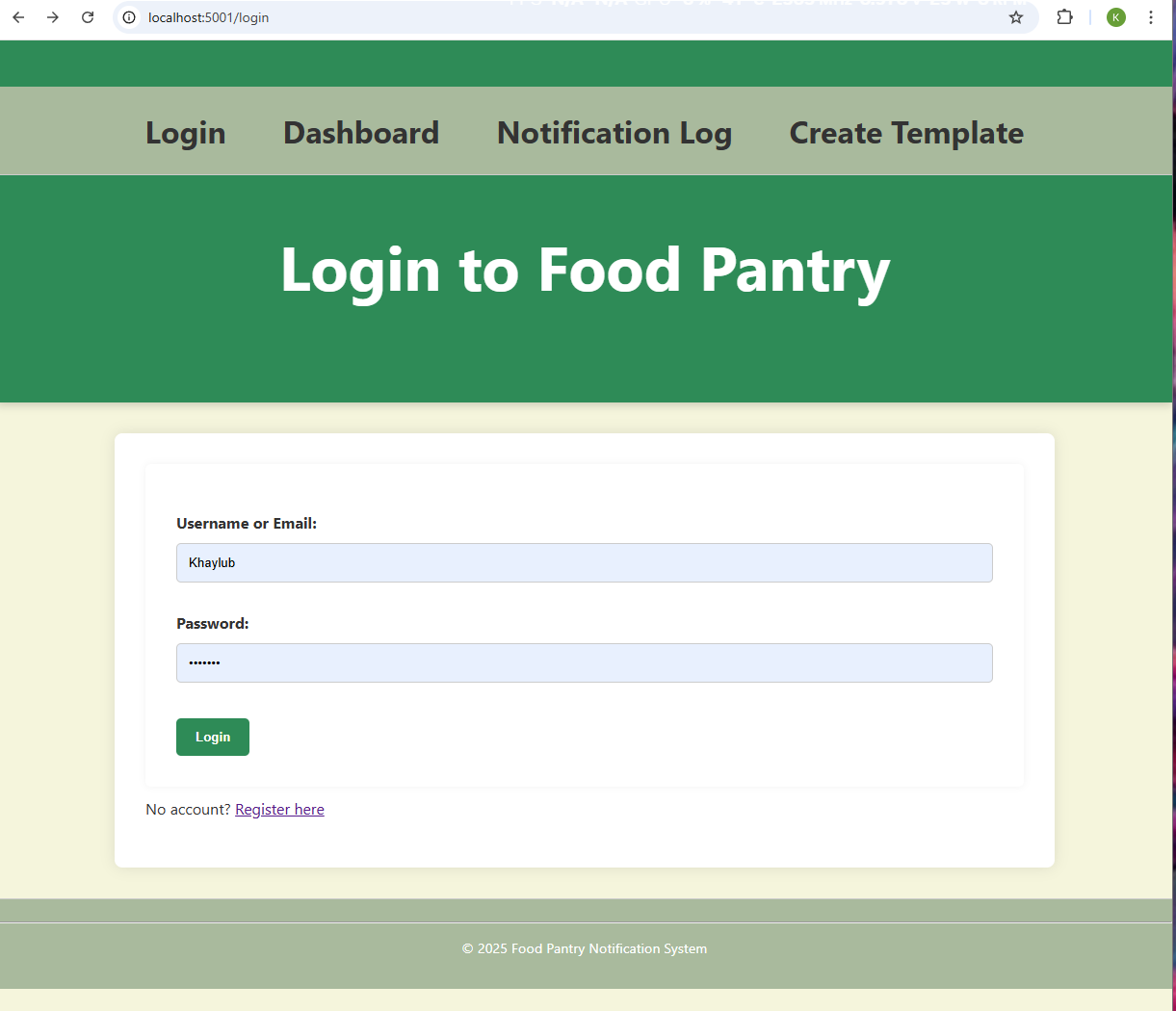
3. Can logs be filtered by date, sender, or keyword?

4. Who is authorized to view or manage the logs?

1. **Using a GUI builder, construct the necessary user-interface screens for these stories. Part 2: GUI Screenshots**

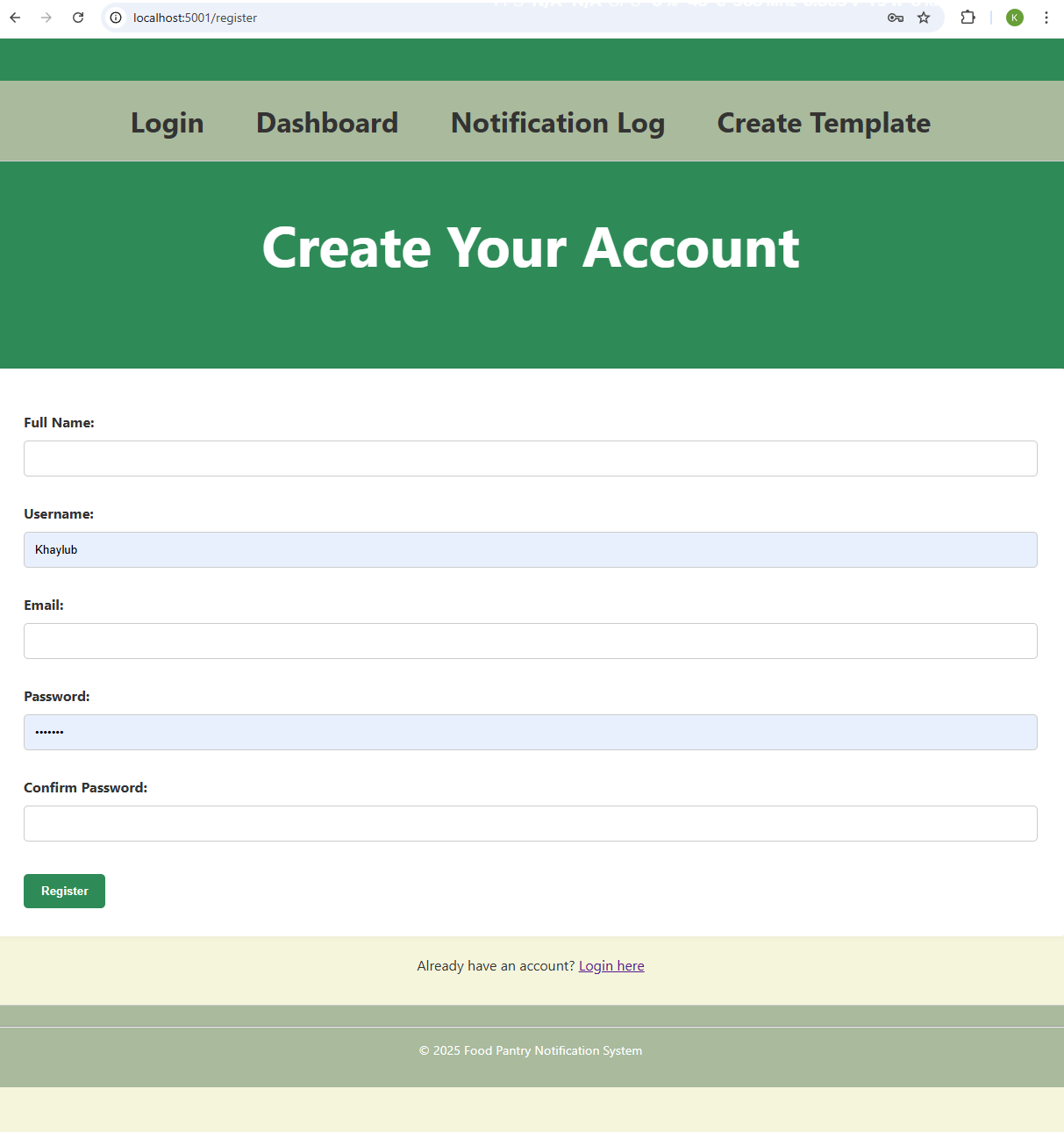
 This template provides a login form for the Food Pantry Notification System.

 It extends the base layout and defines blocks for title, header, and content.



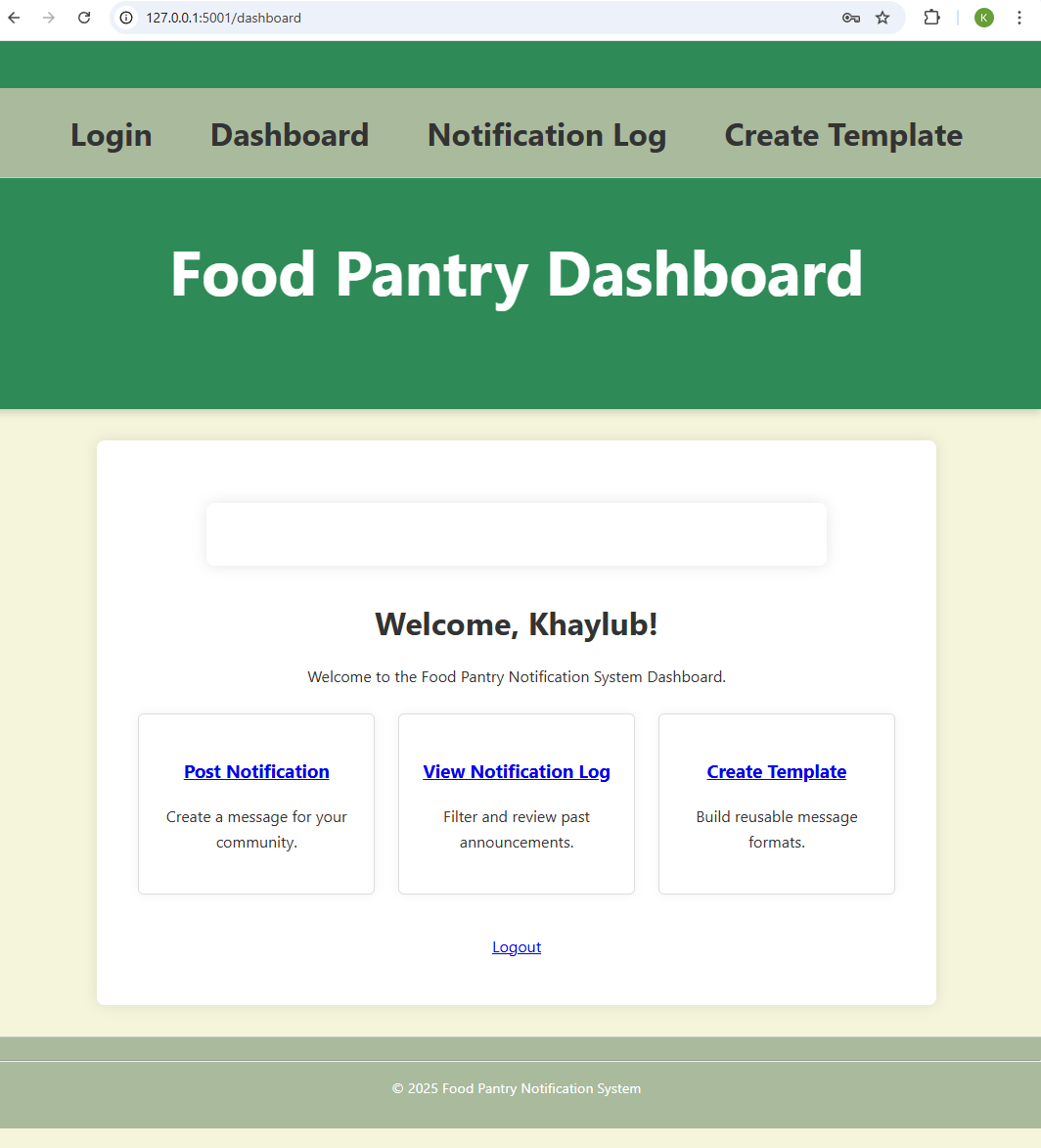
This template provides the registration page for the Food Pantry Notification System.

 It extends the base layout and defines blocks for title, header, and main content



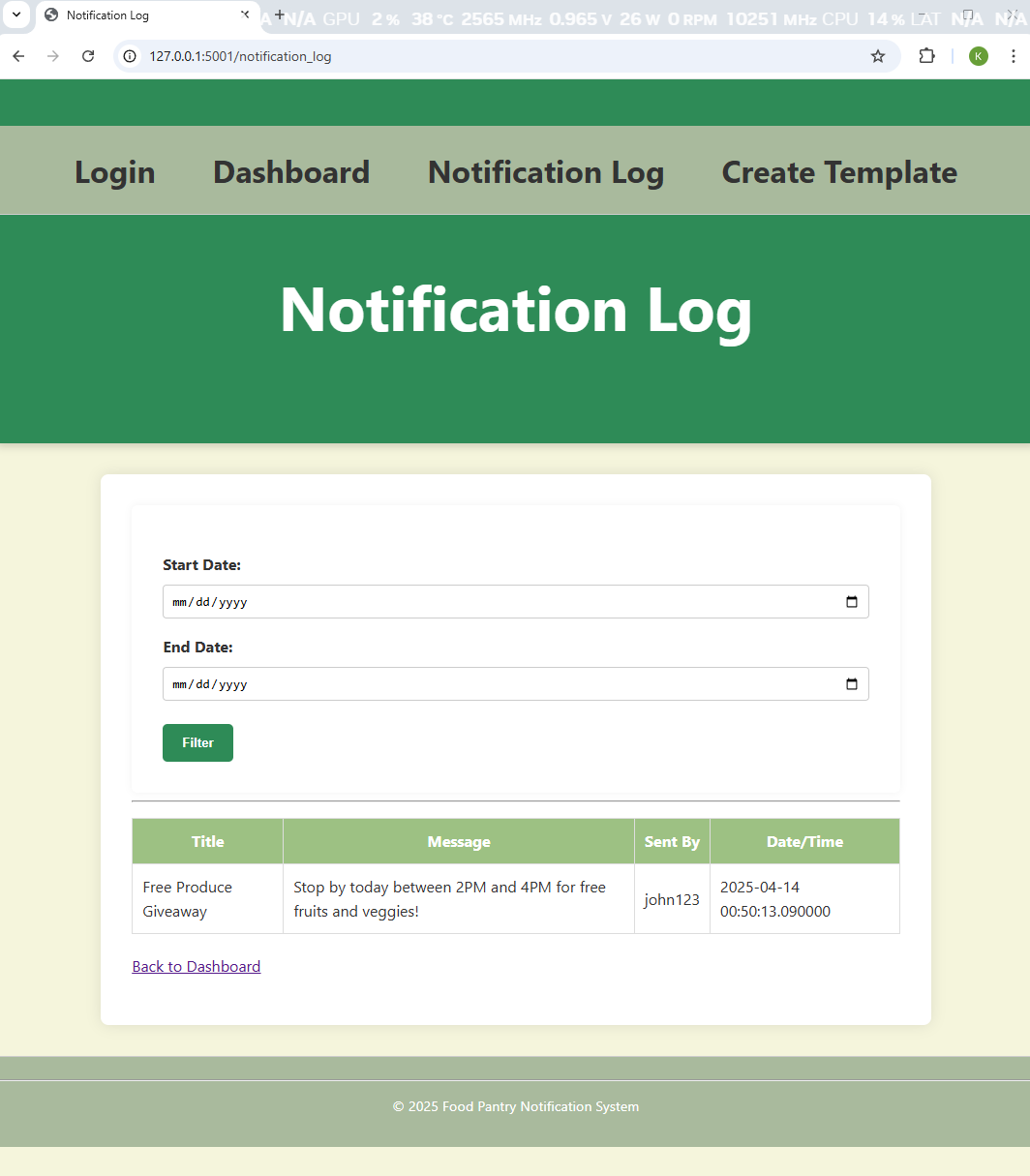
 This template renders the user dashboard for the Food Pantry Notification System.

 It shows login status, notification tools, and dashboard navigation.



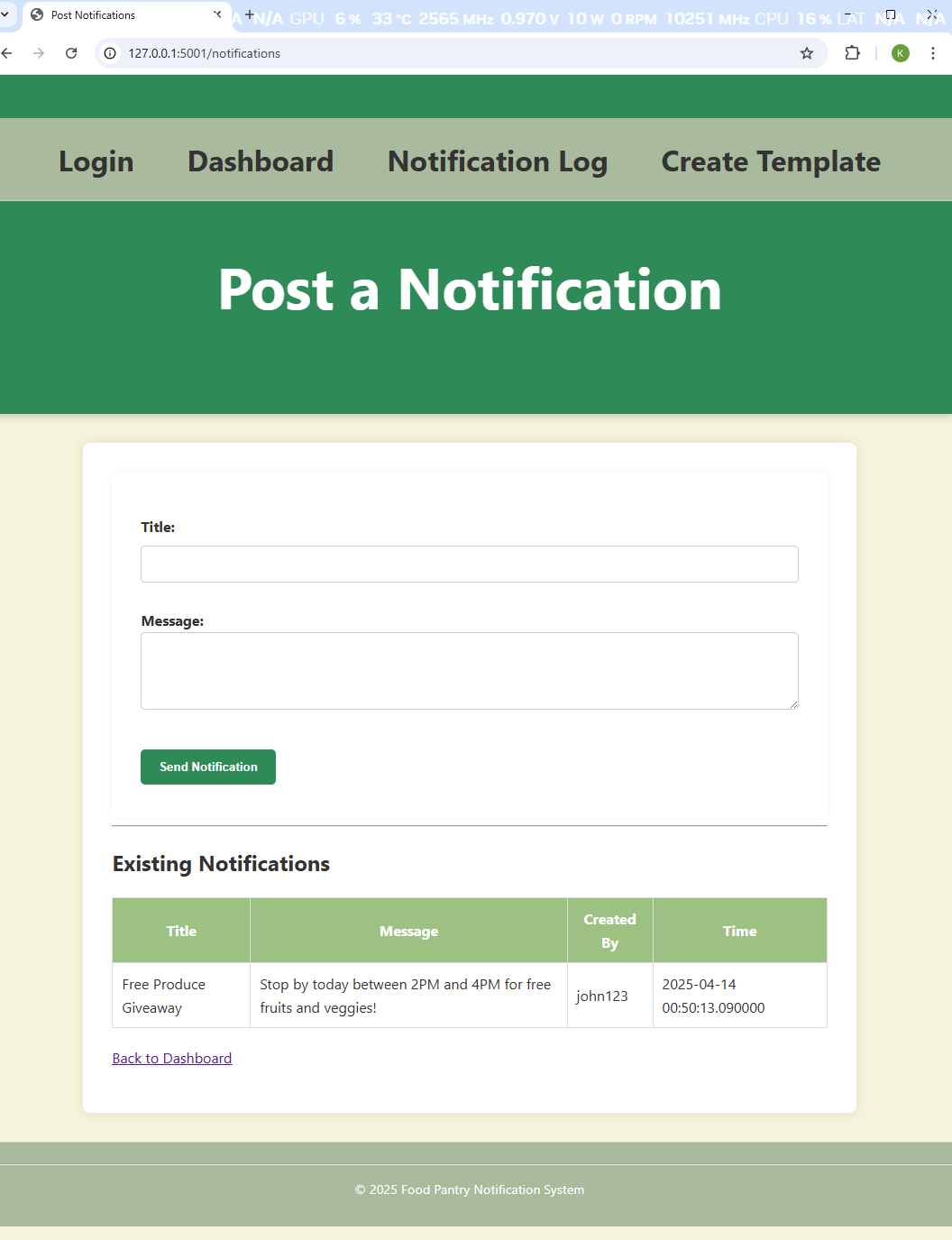
This template allows users to filter and view a log of sent notifications.

 It extends the base layout and includes form controls for date filtering.



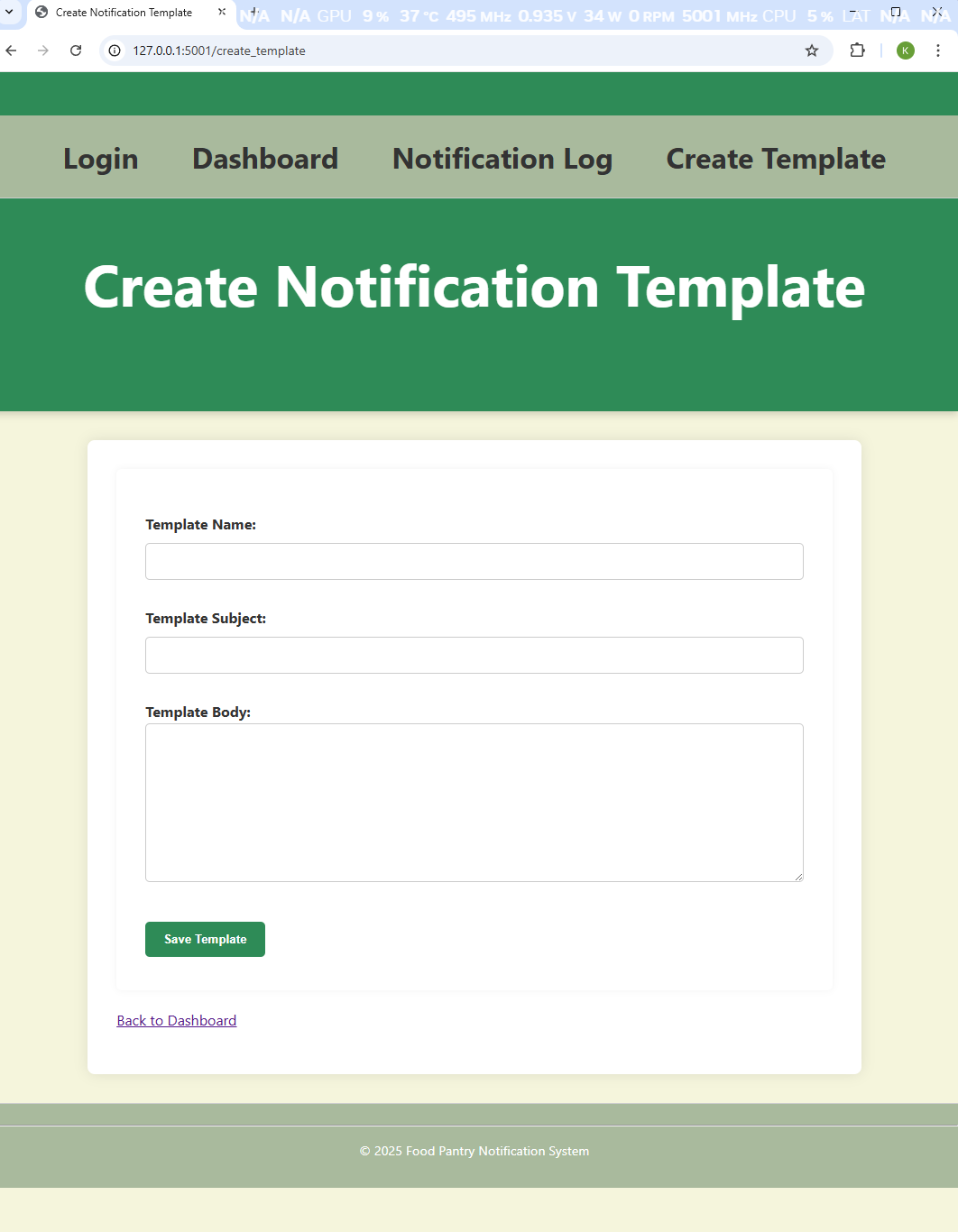
This template allows users to post new notifications and view existing ones

in the Food Pantry Notification System.



This template allows users to create and save message templates

 for the Food Pantry Notification System.



**Part 3. “Write up some implementation notes for these stories.”**

Description: Implementation details of the backend structure and MVC design of the system. This section outlines the major technical components, architecture, and implementation strategy used to develop the Sprint 1 features.

* Flask handles routing and web server
* MongoDB Atlas stores users and notifications
* Jinja Templates render dynamic views
* Flask-Session manages login sessions
* Werkzeug hashes passwords securely
* Flask Blueprints: Used to modularize controllers such as auth and notifications.

**Folder Structure**

**The system follows an MVC-style structure:**

* controllers/ – Python route logic
* models/ – MongoDB schemas (optional)
* static/ – CSS and JavaScript
* src/views/templates/ – HTML templates
* project-backlog/ – Agile planning docs
* tests/ – Functional testing scripts
* documentation/ – Notes, screenshots, flow diagrams

**Planned Routes and Logic**

* • /register – GET/POST for account creation
* • /login – GET/POST for user login
* • /logout – Clear session and redirect
* • /dashboard – Landing page after login
* • /notifications – Form to post and list messages
* • /notification\_log – Filtered log view
* /create\_template – Save reusable message formats

**Security Measures**

* Passwords are hashed using Werkzeug’s `generate\_password\_hash()` for security.
* Session cookies store user\_id and username post-login. Routes use decorators to block unauthorized access.)
* Login and register routes sanitize inputs and validate uniqueness