Project Specification: Thryft

Course: CS-UY 4513 - Software Engineering

Professor: Dr. DePasquale

Project Title: Thryft

Date: 09-28-2025

1.0 Project Overview

The system, named Thryft, will be a centralized platform for buying and selling goods through a swipe-based interface inspired by Tinder. It is intended to be used by students, local sellers, and administrators who want a streamlined, engaging way to exchange goods.

2.0 Core Requirements

The project must adhere to the mandatory characteristics outlined in the project specification template document.

2.1 User-Based System

The application must be a user-based system, requiring a secure user authentication and authorization framework. User accounts will be managed via standard email/password registration and login.

2.2 User Roles (Minimum 3)

- **Buyer:** Can browse for products by swiping through the catalog. The buyer can also filter the products by category to help them find what they are looking for. Lastly, the buyer will be able to see their previous purchases.
- **Seller:** The seller can create/manage listings, respond to buyer messages, and accept/decline offers.
- **Moderator:** The moderator can delete fraudulent users and listings. The moderator also has access to view users, listings, and reports.

2.3 Persistent Storage

Proposed Database Schema (max 10 tables):

- **users:** Stores user and profile information (user_id, name, location…)
- **role:** Define role for user (buyer, seller, moderator)
- **user_roles:** Link user to role(s)
- **listings:** Stores data for listings
- **swipes:** Records swipe actions
- **messages:** Stores messages & offers for each user
- **flags:** Stores all user reports and fraudulent listings

- **likes:** Stores product ID and user ID for a liked item

2.4 Modular Architecture (3-5 Modules)

The system must be logically divided into a minimum of 3 and a maximum of 5 modules. These modules must demonstrate clear dependencies and rely on each other for full functionality.

Proposed Modules:

1. **User & Identity Management**
   Manages user registration, login, and profile information.
   - Defines and assigns user roles.
   - **Dependency:** Core dependency for all other modules.

2. **Product Management**
   - Allows Sellers to create, update, and delete product listings.
   - **Dependency:** Depends on User & Identity Management for authentication and permissions.

3. **Swipe Engine**
   - Records swipe actions (like/dislike).
   - Allows Buyers to browse and filter products.
   - **Dependency:** Depends on Product Management for listings and User & Identity Management for role enforcement.

4. **Messaging**
   - Users can view incoming and outgoing likes, messages, and offers.
   - **Dependency:** Depends on Swipe Engine (for likes) and User module (for roles).

5. **Moderation**
   - Allows Moderators to take down fraudulent products.
   - Allows Moderators to review reports and moderate users/listings.
   - Allows Moderators to remove users
   - **Dependency**: Depends on the User & Identity Management for users and Product Management for listings

2.5 API Interfaces

Each module must expose a RESTful API interface. This is crucial for enabling the modules to communicate with each other and for potential future integrations. The API should follow standard REST conventions.

Example Endpoints:

- **User & Identity Management API:**
  - POST /api/users/register
  - POST /api/users/login
  - GET /api/users/:id

- **Product Management & Moderation API:**
    - POST /api/listing
    - GET /api/item
    - DELETE /api/listing/:id
    - PUT /api/listing/:id

- **SWIPE & Match Engine**
    - POST /api/swipes
    - GET /api/likes
    - POST /api/message

- **Messaging**
    - POST /api/send-message
    - GET /api/receive-message

- **Moderations**
    - DELETE /api/listing/:id
    - DELETE /api/user/:id
    - GET /api/users/:id
    - GET /api/listing

3.0 Technical Stack

- **Language:** Ruby
- **Framework:** Ruby on Rails
- **Database:** PostgreSQL
- **Testing:** RSpec for unit and integration testing

4.0 Deliverables

The final submission must include:

1. **Complete Source Code:** A well-structured and commented codebase, including all necessary configuration files.
2. **Project Documentation:** A README.md file that explains the system's architecture, setup instructions, and how to run the application.
3. **API Documentation:** A separate document or file detailing all API endpoints, their expected parameters, and response formats.
4. **Presentation:** A brief presentation and demonstration of the application's core features.