Introduction to UnWrap 'n' Savor

We don't deliver, we warp to you

•••

Team - 7

https://github.com/karankumbhar47/Food-Delivery-System

Team:

- Shubham Balasaheb Daule (12141550)
- Karan Sunil Kumbhar (12140860)
- Udan Vedant Ghanshyam (12141690)
- Ahire Sandesh Naval (12140110)

Problem statement

The absence of a dedicated Food Delivery System within our campus is a notable challenge affecting the overall dining experience for our students, faculty, and staff.

Challenges:

1. Lack of Centralized Ordering Platform.
2. Limited Accessibility to On-Campus Eateries.
3. Manual and Time-Consuming Ordering Process.
4. Missed Convenience for Campus Community.

Designing and Implementing an Android App for Food Delivery System to meets the challenges facing in our campus ?

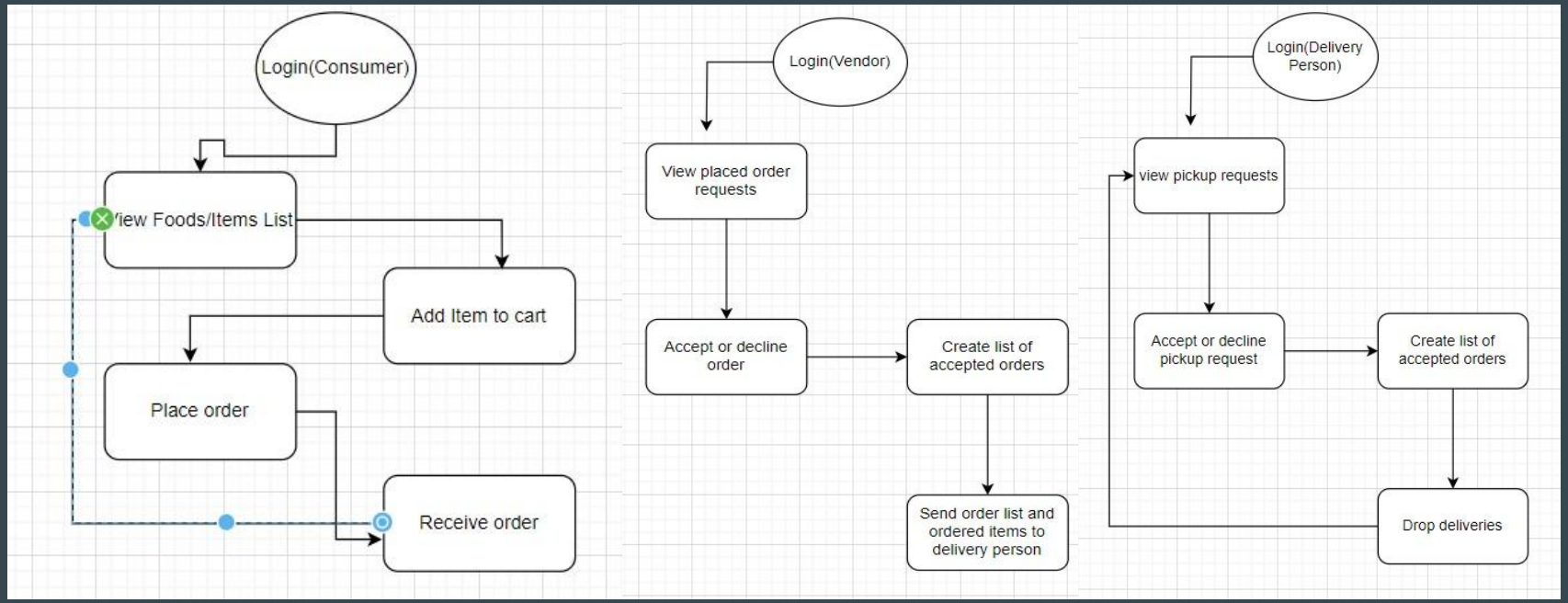
Overview

In response to the identified challenges in our campus dining experience, we are introducing a revolutionary Food Delivery System to enhance convenience and accessibility for our students, faculty, and staff.

Key Points:

1. Seamless Ordering Process.
2. Integration with On-Campus Eateries.
3. Efficient Backend Processing.
4. Optimized Delivery Logistics.

Basic Solution



Innovation

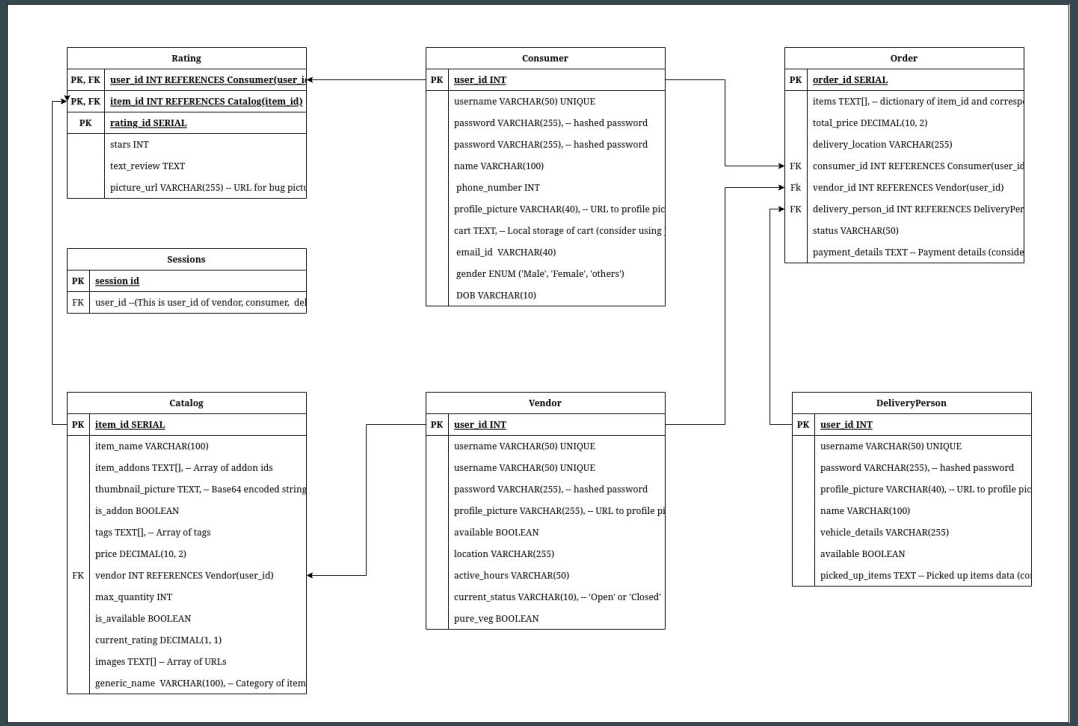
1. Personalized Recommendations:
    a. Use machine learning algorithms to analyze user preferences and order history to provide personalized food recommendations.
    b. Incorporate an intelligent system that suggests complementary items or discounts based on the user's past orders.
2. Dynamic Pricing Algorithms:
    a. Implement dynamic pricing based on factors like demand, time of day, and weather conditions.
    b. Offer personalized discounts during off-peak hours or bad weather to attract more orders.
3. Customizable Meal Plans:
    a. Allow users to create and customize weekly or monthly meal plans.
    b. Offer discounts for users who subscribe to regular meal plans, promoting customer retention.

Solution Approach, Roadmap

1. Defining Functional Requirements
2. Designing System Architecture
3. Defining a basic APIs (OpenAPI) and database schemas (SQL)
4. Choosing frameworks for different modules
5. Implementing basic test functionality of the modules for testing
6. Creating Minimum Viable Product (MVP) with continuous unit and integration testing
7. Adding additional features

Frameworks, Libraries and SDKs

- ★ Database
  - SQLite
- ★ Application Server
  - Flask
- ★ API Definition
  - OpenAPI
- ★ Client Applications
  - Android
- ★ Testing
  - Unit Tests:
    - PyTest
    - Android
- ★ Version Control
  - Git/GitHub
- ★ API Documentation
  - OpenAPI documentation generator



Technical Soundness/Feasibility

1. **Flask:** is lightweight and removes unnecessary complexity
2. **SQLite:** The self contained SQLite database simplest deployment
3. **Git:** Popular VCS and easily allows for collaboration
4. **OpenAPI:** Ensures standardization and well-documented API
5. **OpenAPI:** Extensive repository of generators accelerates the development process. The ability to generate code snippets, client libraries, and server stubs provides a rapid starting point, saving valuable development time.
6. **Android:** Being most commonly used and preferred platform for most users targeting Android as the preferred platform aligns with its widespread use. This decision ensures maximum reach of the application.
7. **Android:** Build in unit tests offer convenient solution for testing components
8. **PyTest:** Robust and feature-rich testing framework for Python. and simplifies the test writing process.
9. **PyTest:** Extensive plugin ecosystem allows for customization based on project needs.

Execution Timeline

FROM	TO	EXECUTION
05 Jan	06 Jan	Defining System Architecture and Functional Requirements.
07 Jan	10 Jan	Frontend Designing And Implementing it.
11 Jan	13 Jan	Defining API and Implementation.
14 Jan	14 Jan	Defining Database.
15 Jan	24 Jan	Backend Implementation.
25 Jan	03 Feb	Integrating Every Modules.
04 Feb	08 Feb	Testing.
09 Feb	09 Feb	Deploying.

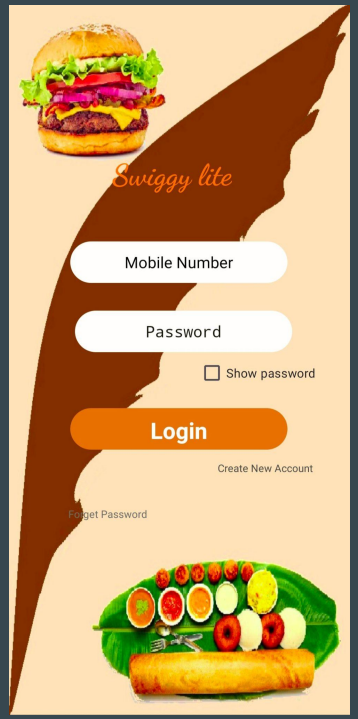
Challenges:

1. Tracking in Real-Time:
    a. Ensuring accurate and seamless real-time tracking for users to monitor the status and location of their orders.
2. Designing API from Scratch:
    a. Developing a robust and scalable API architecture tailored to the specific needs of the food delivery system.
3. Optimizing Delivery Path for Delivery Person:
    a. Creating algorithms to optimize delivery routes, minimizing travel time and enhancing the efficiency of delivery personnel.
4. Thinking from the Perspective of All Three Stakeholders:
    a. Addressing the diverse needs and preferences of consumers, vendors, and delivery persons to create an inclusive and user-centric experience.
5. Designing Efficient Database:
    a. Developing an efficient and well-structured database to handle the complexities of catalog management, user data, and order information.
6. Improving User Experience:
    a. Continuously enhancing the app's design, features, and interactions to provide a seamless, enjoyable, and user-friendly experience.

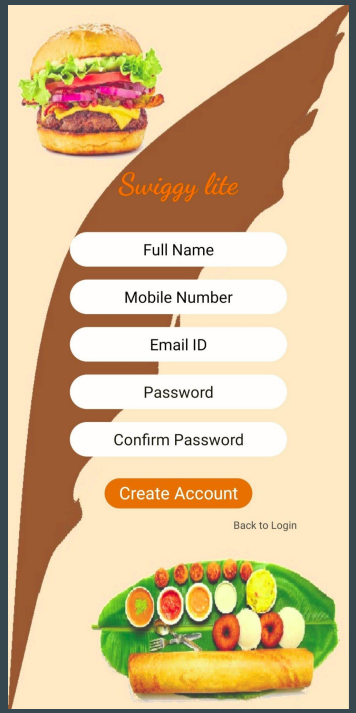
Future Developments

1. Enhanced User Experience:
    a. Refinement of User Interface (UI): Continuously improve the app's visual appeal and ease of navigation.
    b. User Feedback Integration: Implement mechanisms to collect and act upon user feedback for ongoing enhancements.
2. Payment Innovation:
    a. Contactless Payment Expansion: Introduce support for additional contactless payment methods.
3. Add-on:
    a. Selected food items like cakes or pizzas will come with addon like selecting decorations to be added on cakes or selecting extra topping.
4. Notification:
    a. Notifications about limited time offers given by food vendors.
    b. Notification warning if stocks of items are running out.
5. Web Application:
    a. We are going to creating cross platform web application for devices that run on other os such as ios, windows or linux.

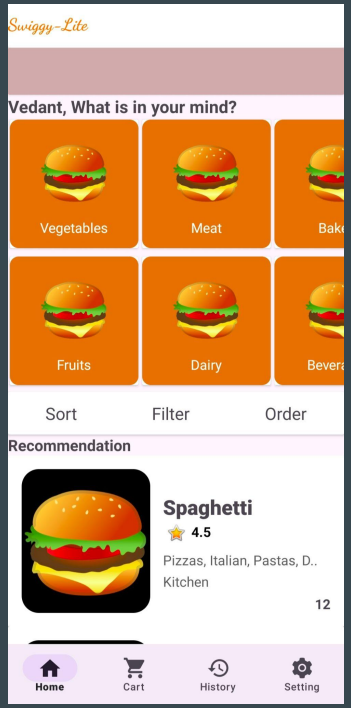
Prototype



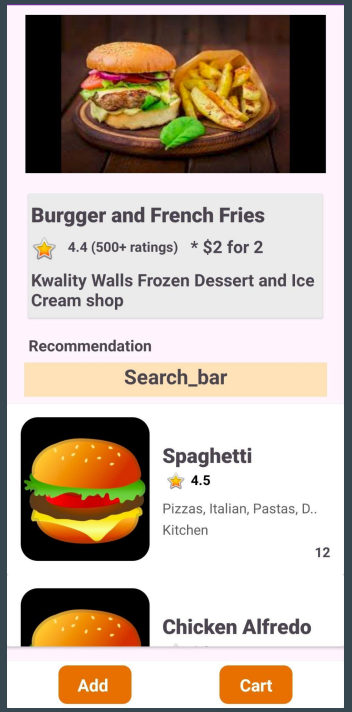
Login Page



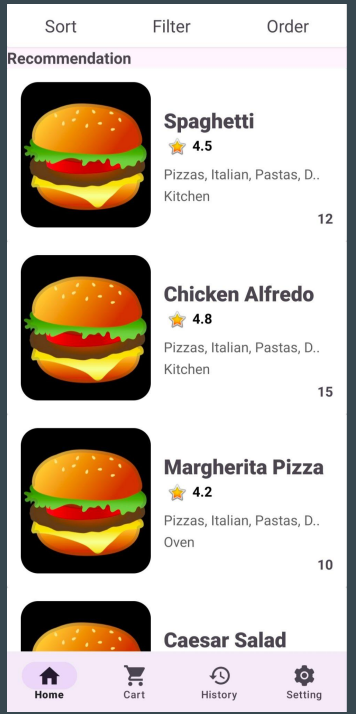
Register Page



Home Page



Item Detail Page



Category Page

Thank You