Grocery Store

Problem definition

Modern Application Development - I

Last Updated: 23-05-28

Frameworks to be used

- Flask for application code
- Jinja2 templates + Bootstrap for HTML generation and styling
- SQLite for data storage
- All demos should be possible on a standalone platform like replit.com and should not require setting up new servers for database and frontend management

Grocery Store

- It is a multi-user app (one required admin/store manager and other users)
- Used for buying grocery
- User can buy many products for one or multiple sections
- Store manager can add section/category and products
- Each section/category will have
  - ID
  - Name etc.
- Each product will have
  - ID
  - Name
  - Manufacture/Expiry date
  - Rate per unit etc. (Rs/Kg, Rs/Litre)
- Every category can have a number of products
- System will automatically show the latest products added

Terminology

- Inventory
- Section/Category - List of products, amount etc.
- Product - Name, Price etc.
- Dynamic Pricing (optional) - Product prices can go up/down depending upon the season/demand

Similar Products in the Market:

1. BigBasket
    - Web, IOS and Android
2. Blinkit
    - Web, IOS and Android

- These are meant for exploring the idea and inspiration
- Don’t copy, get inspired

Core Functionality

- This will be graded
- Base requirements:
  - Admin/Store Manager login and User login
  - Category Management
  - Product Management
  - Buy products from one or multiple Categories
  - Search for Category/Product

Core - Admin and User Login

- Form for username and password for user
- Separate form for admin login
- You can either use a proper login framework, or just use a simple HTML form with username and password - we are not concerned with how secure the login or the app is
- Suitable model for user

Core - Inventory Management (Only for Store Manager)

- Create a new section/category
  - Storage should handle multiple languages - usually UTF-8 encoding is sufficient for this
- Edit a section/category
  - Change name/type or image
- Remove a section/category
  - With a confirmation from the admin

Core - Product management (Only for Store Manager)

- Create a new product
  - Storage should handle multiple languages - usually UTF-8 encoding is sufficient for this
- Edit a product
  - Change price/available quantity or image
- Remove a product
  - With a confirmation from the admin
- Allocate section while creating product

Core - Search for and Cart multiple products

- Ability to search sections based on section/category
- Ability to search products based on price, manufacture date etc.
- Ability to add multiple products in a cart (may or may not belong to same category)

Core - Buy Products

- Display all the products available for a given category to the users
- Ability to buy multiple products from one or multiple sections.
- Ability to show out of stock for the products that are not available.
- Ability to show the total amount to be paid for the transaction.

Recommended (graded)

- APIs for interaction with sections and products
  - CRUD on sections
  - CRUD on products
  - Additional APIs for getting the sections/products to display
- Validation
  - All form inputs fields - text, numbers, dates etc. with suitable messages
  - Backend validation before storing / selecting from database

Optional

- Styling and Aesthetics
- Proper login system
- Export section/product engagement (number of products bought, frequently bought products, most demanded sections)
- Predict demand of a product based on the previous trends
- Provide promo codes for the first transaction of new user. (gives say x% discount on the total amount on the first purchase)

Evaluation

- Report (not more than 2 pages) describing models and overall system design
  - Include as PDF inside submission folder
- All code to be submitted on portal
- A brief (2-3 minute) video explaining how you approached the problem, what you have implemented, and any extra features
  - This will be viewed during or before the viva, so should be a clear explanation of your work
- Viva: after the video explanation, you are required to give a demo of your work, and answer any questions
  - This includes making changes as requested and running the code for a live demo
  - Other questions that may be unrelated to the project itself but are relevant for the course

Instructions

- This is a live document and will be updated with more details and FAQs (possibly including suggested wireframes, but not specific implementation details) as we proceed.
- We will freeze the problem statement on or before 14th May, beyond which any modifications to the statement will be communicated via proper announcements.
- The project has to be submitted as a single zip file.