System Design Document

Project: Supermarket Price Comparison App

By: A-Team

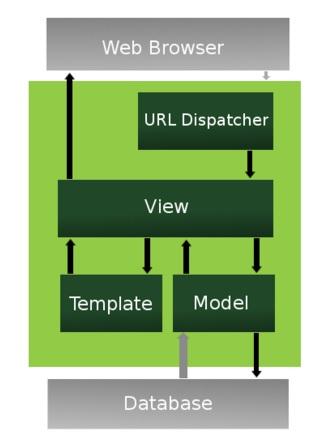
# Architecture

## Web Framework

Django is a high-level Python based web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers it helps with web development, so that you can focus on the app without reinventing the wheel. Django follows the “Batteries included” philosophy, providing robust features out of the box. It takes care of ORM, authentication, URL routing and form handling. The framework emphasizes reusability, less code and the principle of DRY (Don’t repeat yourself).

Django follows the Model-View-Controller (MVC) architecture pattern, but with some differences.

The “controller” in Django is represented by the framework itself while the “view” corresponds to the actual web pages. This leads to the commonly used term “Model-View-Template” (MVT) for Django applications.



# Data Access Design

## Model Design (Data Model)

In the context of the supermarket price comparison app the “model” part of the MVC pattern will involve defining the various entities needed to represent the data stored in the database. This includes:

* **Supermarkets** Attributes: Name, Location, Opening Hours, etc
* **Products** Attributes: Name, Brand, Category, Description, etc.
* **Prices** Attributes: Product, Supermarket, Price, Date, etc.
* **Users** Attributes: Username, Email, Password, Location, etc.
* **Shopping Lists** Attributes: User, Products, Date Created.

**Relationships:**

* A **Supermarket** can have many **Products** with different **Prices**
* **Products** are linked to **Prices** across multiple **Supermarkets**
* **Users** can create multiple **Shopping Lists** containing various **Products**.

## Functional Decomposition

Key entities in the data model include **Supermarkets**, **Products, Prices and Users**. The following functions are identified from user stories.

* **Users:** Register, Login, Create Shopping List, Compares Prices, View Past Comparison.
* **Supermarkets:** Manage Product Listings, Update Prices
* **Products:** Add to Shopping List, Compare Across Supermarkets
* **Prices:** Track Price History, Display Current Prince

The functions help identify the Views needed, such as:

* **User Registration View**
* **Login View**
* **Shopping List View**
* **Price Comparison View**
* **Product Detail View**
* **Supermarket Management View**

## Database Investigation

*List which database management systems you investigated, and which is more appropriate for your web app. Justify your decision.*

# Security Design

## Framework Security

We are using Django as a web framework for the application. It has several in-build security features which we will leverage to protect the project against vulnerabilities. Some of the features are:

* Cross-Site Forgery (CSRF) Protection
* Cross-Site Scripting (XSS) Protection
* SQL Injection Prevention
* Clickjacking Protection
* Authentication

## Security Mechanisms

To protect the users, we will implement the following measures:

* User Authentication and Password management
  + We will rely on Django’s user model and authentication system to securely store and validate user data. The passwords will be hashed.
* CSRF Protection
  + Django’s middleware (CSRF) will be enabled, and all forms will include a CSRF token.
* SQL Injection Prevention
  + We will use Django’s ORM for all database queries to ensure automatic protection against SQL injection.
* Clickjacking Protection
  + We will use Django’s X-Frame-Options middleware to prevent the site from being framed by other websites.

## User Roles and Permissions

* Admin Role
  + Full access to manage users, product, stores and view price history data
  + Can update and delete any records
  + Access to view reports and logs.
* Registered User Role
  + Can add products to their cart, view prices and perform comparisons.
  + Can view their own store preferences and history
  + Cannot access or modify admin data.
* Guest Role
  + Access to view product prices and perform comparisons but can’t save preferences or add items to their cart.
  + Limited functionality needs to register for full access.

We will use Djangos built-in User Model along with group permissions system to manage these roles. Each user role will be assigned to a group with specific permissions, and access control will be enforced via Django’s @permission\_required decorator.

# User Interface Design

## View Design (UI List)

*This is the “view” part of the MVC pattern.*

*List the user interfaces needed to perform each user story.*

### User Story 1

As a user, I want to add my products to a Cart that will display the total of my shopping. I also want to remove or update the quantities in the cart. I want to access the cart from any screen and want to know if the product is my cart already when looking at products.

### User Story 2

As a user, I want to use my location so that the app will only look at the supermarkets in my area.

## UI Design

Record palette, fonts, look and feel, bootstrap used, etc. for your web app.

Include a wireframe or mockup for the landing page to illustrate the style you are aiming for.