**KitaabShop Application Architecture and Design**

**1. Overview:**

KitaabShop is an e-commerce web application where users can browse, search, and purchase books. The architecture follows a classic web application model built on Flask as the backend framework, using SQLite as the database. It follows the MVC (Model-View-Controller) pattern to separate concerns and make the system scalable and maintainable.

**2. High-Level Architecture:**

**Client-Side (Frontend):**

* **Technologies:** HTML, CSS (for UI consistency), JavaScript (for form validation, search, etc.)
* **Functionality:**
  + Users interact with various pages like Home, Product Listing, Product Description, Cart, and Profile pages.
  + Responsive design using CSS media queries.
  + Pages are rendered dynamically with data coming from the Flask backend.
  + User actions like adding to cart, login, registration, and purchase are submitted to the server for processing.

**Server-Side (Backend):**

* **Technologies:** Flask (Python Web Framework), SQLite (Database)
* **Architecture Pattern:** MVC (Model-View-Controller)
* **Functionality:**
  + Handles routing and logic for various user interactions (adding to cart, registration, login, etc.).
  + Manages sessions for authenticated users.
  + Interacts with the SQLite database to fetch, update, and manipulate data.
  + Routes incoming requests to the corresponding view templates and passes appropriate data to the views.
  + Processes payments, user profile changes, and orders.

**3. Folder Structure:**

The folder structure of the application organizes different components for maintainability and scalability.



**4. Application Components and Design:**

**Frontend (View Layer):**

* **HTML Pages (Jinja Templating):**
  + Pages are dynamically rendered using Jinja2 templating engine, which inserts data (like books, cart items, etc.) into the HTML.
  + Each page corresponds to different routes in Flask and presents data fetched from the backend.
  + **Main Pages:**
    - **Home Page (home.html):** Displays search bar, categories, and featured books.
    - **Product Page (productDescription.html):** Shows book details and options to add to cart.
    - **Cart Page (cart.html):** Displays the user's selected items and the total price.
    - **Profile Pages (profileHome.html, editProfile.html, etc.):** Allow users to view and update their details.
* **CSS for Styling:**
  + Custom CSS ensures the application has a cohesive and familiar look.
  + Each page has its specific styles, and some shared styles are in topStyle.css for uniform navigation and branding.
  + Pages are responsive and adapt to various screen sizes using media queries.

**Backend (Controller Layer):**

* **Routing (Flask):**
  + The application uses Flask's routing to define various endpoints, such as /, /cart, /profile, etc.
  + Each route corresponds to a function in main.py that handles the request, interacts with the database (if needed), and renders the appropriate template with data.
* **Controllers (in main.py):**
  + **Home Controller:** Fetches books and genres to display on the homepage.
  + **Product Controller:** Fetches specific product details and renders the product description page.
  + **Cart Controller:** Handles adding items to the cart, removing items, and calculating the total price.
  + **Profile Controller:** Handles user-specific actions like viewing profile, editing profile, and changing the password.
* **Session Management:**
  + User sessions are maintained to track user login status, cart items, and profile data.

**Database (Model Layer):**

* **SQLite Database:**
  + **Tables:**
    - **Users:** Stores user information (ID, email, password, profile details).
    - **Books:** Stores book details (ID, title, author, price, stock).
    - **Cart:** Stores cart information with a mapping between users and their selected books.
    - **Orders:** Stores order details for each user, including the total price and order status.
  + **Database Connection:**
    - Managed through the database.py script using SQLite's sqlite3 library.
    - Queries for fetching data (books, cart items) and updates (adding to cart, placing orders) are handled within the controllers in main.py.

**5. Functional Flow:**

**User Registration and Login:**

* **Registration:**
  + User submits details via the registration form.
  + The backend checks for existing users, hashes the password, and stores the new user in the users table.
* **Login:**
  + User provides email and password.
  + Authentication is done through OTP (One time Password). OTP is sent to user’s email address. User can enter that OTP on next page
  + The backend validates the credentials, and if successful, a session is created for the user.
* **Forgot Password**
  + User can opt for forgot password option if required.
  + Password reset link will be sent to user’s mail address.

**Book Search and Product Viewing:**

* The user searches for books using the search bar.
* The search query is sent to the server, which retrieves matching books from the books table.
* The user can click on a book to view its details on the product page.

**Cart and Checkout:**

* **Adding to Cart:**
  + When a user adds a book to their cart, the cart table is updated with the user ID and book ID.
* **Checkout:**
  + On the checkout page, the user confirms the order, and the system updates the orders table.
  + After a successful order, the cart is cleared.

**Profile Management:**

* **View Profile:**
  + Users can view their profile details, including email, name, address, etc.
* **Edit Profile:**
  + Users can update their profile information, which is stored in the users table.
* **Change Password:**
  + The user submits their old and new passwords. The backend verifies the old password and updates it to the new password after hashing.