**High-Level Design (HLD)**

**System Architecture**

The Super Mall Web Application will be built using a client-server architecture with Firebase as the backend service for authentication, real-time database, and storage. The frontend will be developed using HTML, CSS (Tailwind CSS), and JavaScript.

**Components:**

1. **Frontend:**
   * User Interface (UI) built with HTML, CSS, and JavaScript.
   * User interactions handled using JavaScript.
   * Responsive design with Tailwind CSS.
2. **Backend:**
   * Firebase for database, authentication, and storage.
   * Cloud functions for server-side logic and business rules.

**Modules:**

1. **Admin Module:**
   * Login
   * Create/Manage Shop Details
   * Manage Offer Details
   * Manage Category & Floor
2. **User Module:**
   * Category-wise Details
   * Shop Listings
   * Product Offers
   * Product Comparison
   * Filtering
   * Shop-specific Offers
   * Floor Details
   * View Shop Details

**Low-Level Design (LLD)**

**Use Case Diagram**

The Use Case Diagram will depict the interactions between actors (Admin, User) and the system modules.

**Actors:**

* Admin
* User

**Use Cases for Admin:**

1. Login
2. Create Shop Details
3. Manage Shop Details
4. Manage Offer Details
5. Manage Category & Floor

**Use Cases for User:**

1. View Category-wise Details
2. View Shop Listings
3. View Product Offers
4. Compare Products
5. Filter Products
6. View Shop-specific Offers
7. View Floor Details
8. View Shop Details

**Class Design**

The Class Design will detail the classes and their relationships within the system.

**Classes:**

1. **User**
   * Attributes: userId, username, password, email, role
   * Methods: login(), viewCategories(), viewShops(), viewOffers(), compareProducts(), filterProducts(), viewShopDetails(), viewFloorDetails()
2. **Admin (inherits User)**
   * Methods: createShop(), manageShop(), manageOffers(), manageCategory(), manageFloor()
3. **Shop**
   * Attributes: shopId, shopName, shopDetails, category, floor
   * Methods: createShop(), updateShop(), deleteShop()
4. **Offer**
   * Attributes: offerId, shopId, offerDetails, startDate, endDate
   * Methods: createOffer(), updateOffer(), deleteOffer()
5. **Category**
   * Attributes: categoryId, categoryName
   * Methods: createCategory(), updateCategory(), deleteCategory()
6. **Floor**
   * Attributes: floorId, floorNumber, floorDetails
   * Methods: createFloor(), updateFloor(), deleteFloor()

**Sequence Diagram**

The Sequence Diagram will illustrate the interactions between objects over time for key use cases.

**Example: Admin Login and Create Shop**

1. **Admin Login:**
   * **Admin** -> **UI**: Enter login credentials
   * **UI** -> **Firebase Authentication**: Authenticate Admin
   * **Firebase Authentication** -> **UI**: Authentication success/failure
   * **UI** -> **Admin**: Display dashboard
2. **Create Shop:**
   * **Admin** -> **UI**: Enter shop details
   * **UI** -> **Firebase Database**: Store shop details
   * **Firebase Database** -> **UI**: Confirmation
   * **UI** -> **Admin**: Display success message

**Example: User View Shop Listings**

1. **User View Shop Listings:**
   * **User** -> **UI**: Request shop listings
   * **UI** -> **Firebase Database**: Retrieve shop listings
   * **Firebase Database** -> **UI**: Return shop listings
   * **UI** -> **User**: Display shop listings

**Detailed Diagrams**

**Use Case Diagram**

**Class Diagram**

**Sequence Diagram**

These diagrams should be created using appropriate tools such as Lucidchart, Microsoft Visio, or any other diagramming software.

**Additional Details**

**Logging**

* Use JavaScript or Python logging libraries for logging actions performed by the code.

**GitHub Repository**

* Maintain the code on a public GitHub repository.
* Include a README file with basic workflow and execution instructions.

**Testing**

* Include test cases for all functionalities.
* Ensure the code is modular, safe, testable, maintainable, and portable.

**Deployment**

* Host the application on a cloud platform with proper justification of the system design.
* Optimize the solution at the code level and architecture level.