Industrial Training Project Report On

"Architecture and Technology Stack for a Modern E-Commerce Platform"

Submitted in the partial fulfilment of the requirement for the award of degree of BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE BATCH (2023-2027)

Submitted to:-

Submitted by:-

AYUSH KUMAR

12301122

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING DAV UNIVERSITY JALANDHAR-PATHANKOT NATIONAL HIGHWAY, NH 44, SARMASTPUR PUNJAB 144012

DECLARATION

I, AYUSH KUMAR hereby declare that the work which is being presented in this project/training titled "Architecture and Technology Stack for a Modern E-Commerce Platform" by me, in partial fulfilment of the requirements for the award of Bachelor of Technology (B.Tech)
Degree in "Computer Science and Engineering " is an authentic record of my own work carried out under the guidance of MRS SHALINI SINGH &MR.LUVKUSH SINGH To the best of my knowledge, the matter embodied in this report has not been submitted to any other University/ Institute for the award of any degree or diploma.

AYUSH KUMAR, 12301122

ACKNOWLEDGEMENT

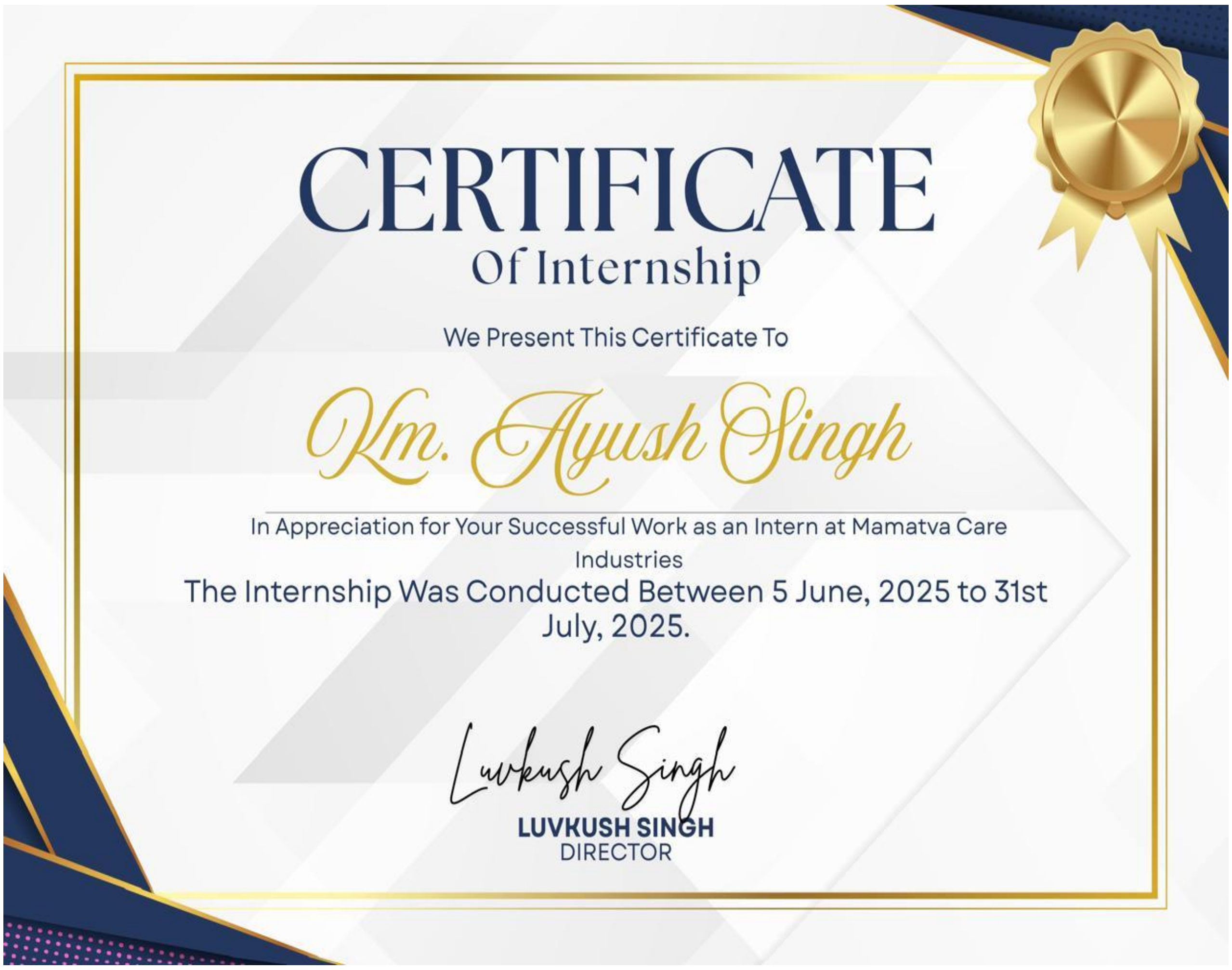
The completion of " Architecture and Technology Stack for a Modern E-Commerce Platform " stands as a testament to the collective dedication and collaborative effort. I extend my heartfelt appreciation to all those who contributed significantly to this project.

My sincere gratitude goes to MRS.SHALINI SINGH for her invaluable guidance, unwavering support, and profound insights throughout the development journey. Her mentorship played a pivotal role in steering the project in the right direction and fostering an environment conducive to innovation and excellence. I express my appreciation to him for his unwavering commitment and collaborative spirit throughout the project's lifecycle. Her perspective played a crucial role in shaping and refining the project's concepts and execution.

Special thanks go to my family and friends for their unwavering encouragement and understanding during the challenging phases of this Endeavour. Their support was a constant source of motivation.

Lastly, I am grateful to all those individuals who, directly or indirectly, contributed to this project. Your support has been instrumental, and I am truly thankful for your contributions.

Training Certificate



CERTIFICATE

Of Internship

We Present This Certificate To

Km. Ayush Singh

In Appreciation for Your Successful Work as an Intern at Mamatva Care Industries

The Internship Was Conducted Between 5 June, 2025 to 31st July, 2025.

Luvkush Singh

LUVKUSH SINGH

DIRECTOR

COMPANY PROFILE

Mamatva Care Industries is an India-based manufacturing unit that produces cosmetics and essential home-care supplies. The company has recently launched its own brand, Mamatukare, which focuses on non-toxic, non-chemical, and eco-friendly products. The production and packaging processes are managed by Indian women through the Mamatukare Foundation, promoting women's empowerment and sustainable livelihoods. Mamatva Care Industries operates as a closely aligned, co-related company of the Mamatukare Foundation, combining ethical production with social impact.

Voice: +91- 8081986701

Website: mamatvatwa care.org

Contents

1 Introduction: Defining the E-Commerce Technology Landscape (Section 1/15) 1

2 Frontend Foundation: HTML5 Structure (The Skeleton) (Section 2/15) 1

3 Styling and Aesthetics: CSS3 and Responsive Design (Section 3/15) 2

4 Interactivity and Logic: JavaScript (ES6+) Core (Section 4/15) 2

5 The Single-Page Application (SPA) vs. Multi-Page Application (MPA) Model (Section 5/15) 3

6 Backend as a Service (BaaS): Introduction to Firebase (Section 6/15) 3

7 Data Persistence: Firestore NoSQL Database (Section 7/15) 4

8 User Identity: Firebase Authentication Flow (Section 8/15) 4

9 E-commerce Feature Deep Dive: Product Catalog Management (Section 9/15) 5

10 E-commerce Feature Deep Dive: Shopping Cart Logic (Client-Side State) (Section 10/15) 5

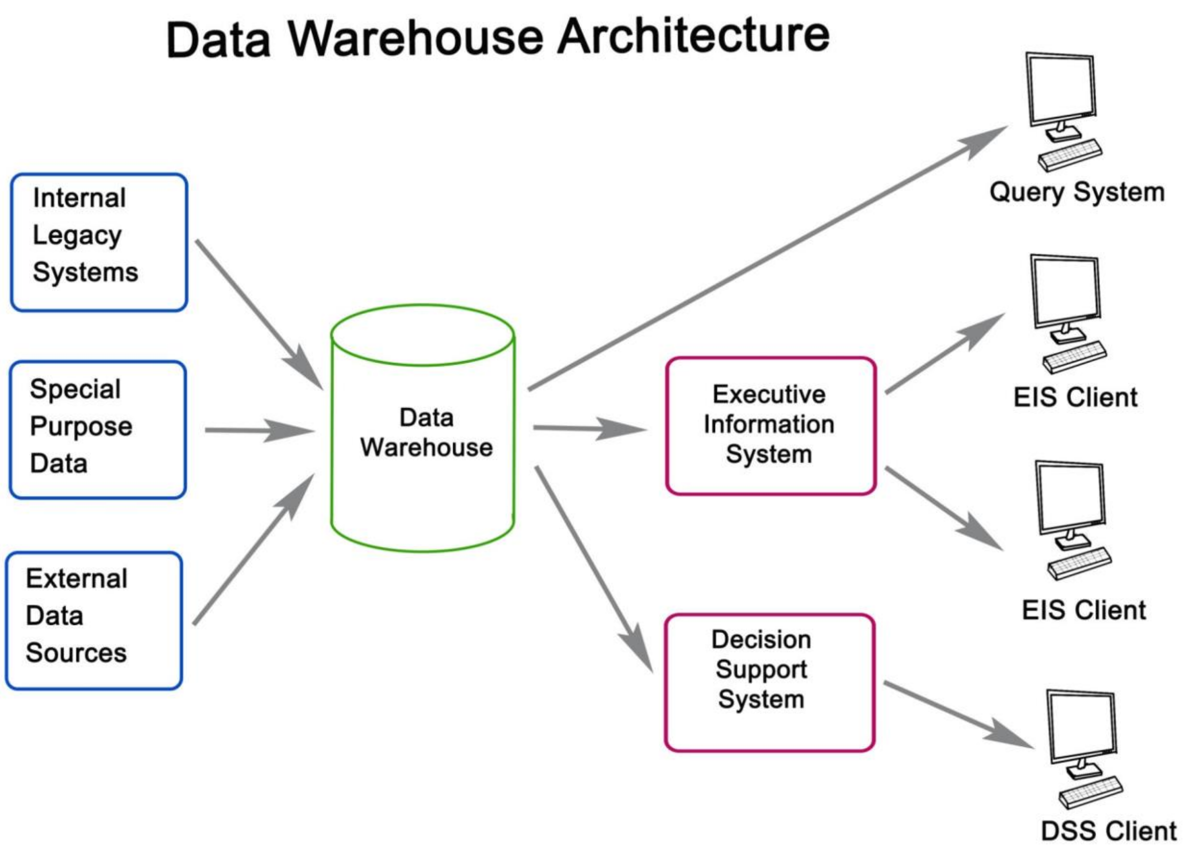
11 Frontend Development Best Practices (Performance & Security) (Section 11/15) 5

12 State Management Strategies (Beyond Vanilla JS) (Section 12/15) 6

13 Deployment and Hosting Considerations (Section 13/15) 6

14 API Design and Microservices (Conceptual Expansion) (Section 14/15) 7

15 Future Enhancements and Scalability (Section 15/15) 7



Frontend Foundation: HTML5 Structure (The Skeleton) (Section 2/15)

HTML5 serves as the semantic backbone of the application. Its primary role is to structure the content, ensuring accessibility and search engine optimization (SEO).

Key HTML5 Features Used:

- **Semantic Elements:** Using elements like <header>, <nav>, <main>, <section>, and <footer> instead of generic <div> elements improves code readability and helps screen readers and search engines understand the document structure.

- **Media Support:** The use of <img> tags for product listings, often enhanced with srcset and <picture> tags for responsive image delivery, although not fully visible in the snippet.

- **Form Validation:** Built-in validation attributes (e.g., required, email, pattern) are critical for ensuring data integrity before client-side JavaScript intervention, such as in the newsletter form.

- **Custom Data Attributes (data-*):** These are essential for connecting the visual elements to JavaScript logic, such as attaching product IDs to "Add to Cart" buttons for easy retrieval during transaction processing.

Styling and Aesthetics: CSS3 and Responsive Design (Section 3/15)

CSS3 dictates the visual presentation, ensuring the site is aesthetically pleasing and functional across all devices. The design emphasizes a clean, modern e-commerce aesthetic.

Core CSS3 Technologies and Concepts:

- **CSS Variables (Custom Properties):** The :root block defines variables like --primary: #1a1a1a; and --accent: #e74c3c;. This technique centralizes the color palette, allowing for rapid, site-wide theme changes and reducing repetitive code.
- **Flexbox and Grid Layouts:**
  - **Flexbox:** Used for one-dimensional layouts (e.g., aligning navigation items, cart contents).
  - **CSS Grid:** Used for two-dimensional layouts (e.g., the product display (.product-list) and the footer layout). This ensures the product grid adapts seamlessly to different screen sizes.
- **Transitions and Animations:** The var(--transition: all 0.3s ease;) variable is used to create smooth visual feedback, like the scrolled state on the navbar and hover effects on product cards, enhancing the perceived performance and user experience (UX).
- **Media Queries:** These are the backbone of **Responsive Web Design (RWD)**. They allow the CSS to apply different styles based on device characteristics (e.g., width, height, orientation), ensuring optimal layout for mobile phones, tablets, and desktops.

1 Introduction: Defining the E-Commerce Technology Landscape (Section 1/15)

The AYUNIK e-commerce application, built upon foundational web technologies, represents a crucial intersection of design, data management, and user interaction. To elaborate on the system ”in use,” we must look beyond the basic files (index.html, script.js, style.css) and analyze the underlying architecture that enables scalability, security, and a real-time shopping experience.

A typical modern web application, including this e-commerce platform, operates on a multitiered architecture, where responsibilities are separated to improve maintainability and scalability.

1. **Presentation Tier (Frontend):** The user interface (HTML, CSS, JavaScript) that runs in the customer’s browser.
2. **Application Tier (Backend Logic):** The server-side code that handles business logic, processing requests, and interacting with the database.
3. **Data Tier (Database):** The storage mechanism for all application data (products, users, orders, cart state).

For lean, modern applications, the Application and Data Tiers are often condensed into a **Backend as a Service (BaaS)** model, which is the path utilized by incorporating Firebase.

2 Frontend Foundation: HTML5 Structure (The Skeleton) (Section 2/15)

HTML5 serves as the semantic backbone of the application. Its primary role is to structure the content, ensuring accessibility and search engine optimization (SEO).

Key HTML5 Features Used:

- **Semantic Elements:** Using elements like <header>, <nav>, <main>, <section>, and <footer> instead of generic <div> elements improves code readability and helps screen readers and search engines understand the document structure.
- **Media Support:** The use of <img> tags for product listings, often enhanced with srcset and <picture> tags for responsive image delivery, although not fully visible in the snippet.
- **Form Validation:** Built-in validation attributes (e.g., required, email, pattern) are critical for ensuring data integrity before client-side JavaScript intervention, such as in the newsletter form.
- **Custom Data Attributes (data-*):** These are essential for connecting the visual elements to JavaScript logic, such as attaching product IDs to ”Add to Cart” buttons for easy retrieval during transaction processing.

3 StylingandAesthetics: CSS3andResponsiveDesign(Section3/15)

CSS3 dictates the visual presentation, ensuring the site is aesthetically pleasing and functional across all devices. The design emphasizes a clean, modern e-commerce aesthetic.

Core CSS3 Technologies and Concepts:

- **CSS Variables (Custom Properties):** The :root block defines variables like --primary: #1a1a1a; and --accent: #e74c3c;. This technique centralizes the color palette, allowing for rapid, site-wide theme changes and reducing repetitive code.
- **Flexbox and Grid Layouts:**
  - **Flexbox:** Used for one-dimensional layouts (e.g., aligning navigation items, cart contents).
  - **CSS Grid:** Used for two-dimensional layouts (e.g., the product display (.product-list) and the footer layout). This ensures the product grid adapts seamlessly to different screen sizes.
- **Transitions and Animations:** The var(--transition: all 0.3s ease;) variable is used to create smooth visual feedback, like the scrolled state on the navbar and hover effects on product cards, enhancing the perceived performance and user experience (UX).
- **Media Queries:** These are the backbone of **Responsive Web Design (RWD)**. They allow the CSS to apply different styles based on device characteristics (e.g., width, height, orientation), ensuring optimal layout for mobile phones, tablets, and desktops.

4 Interactivity and Logic: JavaScript (ES6+) Core (Section 4/15)

JavaScript (specifically ECMAScript 2015/ES6 and newer features) is the engine that drives all dynamic behavior and client-side logic.

Key JavaScript Technologies and Implementation:

- **DOM Manipulation:** Functions like displayProducts() and updateCartUI() dynamically generate or modify the HTML structure based on application state or data fetched from the backend.
- **Event Listeners:** document.querySelectorAll('.add-to-cart') and subsequent addEventListener(' ...) calls define how the application responds to user actions (e.g., adding an item, opening the cart).
- **Asynchronous Operations (fetch and async/await):** Crucial for interacting with the Firebase backend (data retrieval, user sign-in). All interaction with the server is non-blocking, ensuring the UI remains responsive.
- **Module System:** Although a single file application, a real-world application would use JavaScript modules (import/export) to organize code (e.g., separating authentication logic from cart logic), improving maintainability.
- **Data Formatting:** The formatINR() function demonstrates using the modern Intl.NumberFormat API for localization, ensuring prices are displayed correctly for the Indian Rupee market.

5 TheSingle-PageApplication(SPA)vs. Multi-PageApplication(MPA) Model (Section 5/15)

While the provided files suggest a more traditional MPA structure (loading new HTML files for different views), the modern trend is towards SPAs, which offer a faster, more fluid user experience.

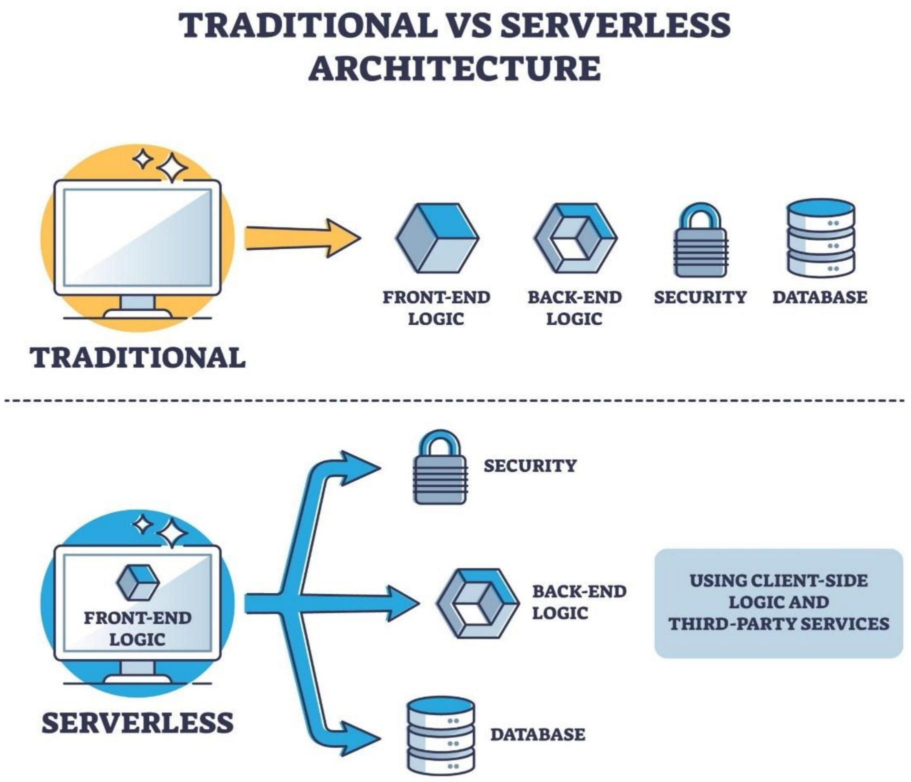
- **MPA (Traditional):** Every user action that changes the view (e.g., clicking on a product category) requires a full page reload from the server. Pros: Better SEO out-of-the-box. Cons: Slow transitions.

Data Persistence: Firestore NoSQL Database (Section 7/15)

Firestore is the primary data storage technology for this application. It is a flexible, scalable, cloud-hosted NoSQL database.

Firestore Key Concepts:

- **Collections and Documents:** Data is structured hierarchically. A Collection (e.g., products, users, carts) contains Documents (single records, e.g., a specific product item). Documents contain fields (key-value pairs).
- **Real-time Synchronization (onSnapshot):** This is Firestore's most powerful feature for e-commerce. Instead of manually polling the server, the client establishes a persistent connection. When the data changes on the server (e.g., a product's stock count is updated), the client automatically receives the update instantly, crucial for a live shopping experience.
- **Mandatory Data Structure:** Following security rules, application data must be structured:
  - **Public Data:** Stored in /artifacts/{appId}/public/data/{collection} (e.g., Product Catalog).
  - **Private User Data:** Stored in /artifacts/{appId}/users/{userId}/{collection} (e.g., User's Cart, Order History).



- **SPA (Modern Approach):** The entire application loads once, and subsequent view changes are handled by JavaScript manipulating the DOM. This requires a **Routing Library** (like React Router, Vue Router, or custom vanilla JS routing) to manage the browser history and display the correct component/view.
- **AYUNIK's Approach (Implied):** The reliance on heavy JavaScript for dynamic updates (like cart and products) indicates a transition towards SPA patterns, where most of the work is done client-side after the initial load.

6 BackendasaService(BaaS):IntroductiontoFirebase(Section6/15)

For a startup or small-to-medium e-commerce platform, using a Backend as a Service (BaaS) like Firebase drastically reduces development time by abstracting away server maintenance and complex API creation. This is evidenced by the mandatory use of the __firebase_config and __initial_auth_token global variables. **Firebase Core Services:**

- **Authentication:** Handles user sign-up, sign-in, and session management.
- **Firestore:** A NoSQL database for real-time data storage.
- **Storage:** For storing large files like product images.
- **Hosting:** Provides fast, secure hosting for the static frontend files (HTML, CSS, JS)

7 Data Persistence: Firestore NoSQL Database (Section 7/15)

Firestore is the primary data storage technology for this application. It is a flexible, scalable, cloud-hosted NoSQL database. **Firestore Key Concepts:**

- **Collections and Documents:** Data is structured hierarchically. A Collection (e.g., products, users, carts) contains Documents (single records, e.g., a specific product item). Documents contain fields (key-value pairs).
- **Real-time Synchronization (onSnapshot):** This is Firestore's most powerful feature for ecommerce. Instead of manually polling the server, the client establishes a persistent connection. When the data changes on the server (e.g., a product's stock count is updated), the client automatically receives the update instantly, crucial for a live shopping experience.
- **Mandatory Data Structure:** Following security rules, application data must be structured:
  - **Public Data:** Stored in /artifacts/{appId}/public/data/{collection} (e.g., Product Catalog).
  - **Private User Data:** Stored in /artifacts/{appId}/users/{userId}/{collection} (e.g., User's Cart, Order Histor

8 User Identity: Firebase Authentication Flow (Section 8/15)

Authentication is the gateway to personalized e-commerce features (saving a cart, viewing order history). The environment uses a custom token for authentication. **Authentication Process (firebase-auth.js):**

1. **Initialization:** Firebase Auth is initialized: auth = getAuth(app);.
2. **Custom Token Sign-In:** The application uses the __initial_auth_token (a pre-generated, secure token) to sign the user in non-interactively using signInWithCustomToken(auth, __initial_auth_token).
3. **Anonymous Fallback:** If the custom token is missing, the application defaults to signInAnonymously(a to ensure all users, even guests, have a unique userId to link their cart to.
4. **State Management:** The onAuthStateChanged listener is vital. It detects when a user logs in or out and updates the client-side userId, which is then used to construct the correct Firestore paths for loading personalized data.

9 E-commerce Feature Deep Dive: Product Catalog Management (Section 9/15)

The efficiency of displaying thousands of products relies on efficient data fetching and rendering. **Technology in Use:**

- **Data Model:** The products collection in Firestore holds all necessary data (Name, Price, Category, Image URL, Stock).
- **Querying:** A dedicated JavaScript function (displayProducts in the provided HTML) fetches the product data. For a large catalog, this query would use pagination and filters (e.g., query(collection(db, 'products'), where('category', '==', 'Tops'), limit(20))) to prevent slow loading times.
- **Dynamic Rendering:** The JavaScript iterates over the fetched array of product documents and dynamically generates the HTML product cards. This separation of data (Firestore) and presentation (JavaScript + HTML template strings) is a fundamental best practice.
- **Image Handling:** Product image URLs are stored in Firestore, but the actual images are typically hosted on a Content Delivery Network (CDN) or Firebase Storage for high-speed delivery.

10 E-commerceFeatureDeepDive: ShoppingCartLogic(Client-Side State) (Section 10/15)

The shopping cart is a critical feature that manages temporary, session-specific data. **Cart Implementation:**

- **Data Structure:** The cart is typically an array of objects, where each object represents a cart item and includes: {key (Product ID + Size), productId, name, size, price, quantity}.
- **Persistence:** In the provided structure, the cart state is stored **privately** in Firestore under the path /artifacts/{appId}/users/{userId}/carts. This ensures the cart state is preserved even if the user closes the browser or switches devices.
- **Real-time Update:** When the cart is modified (add/remove item), the application updates the Firestore document. The onSnapshot listener on the cart collection immediately triggers the updateCartUI() function in all connected browser sessions, ensuring data consistency.
- **Client-Side Functions:**
  - addToCart(item): Handles adding a new item or incrementing the quantity of an existing item.
  - removeFromCart(key): Handles deleting an item.

11 Frontend Development Best Practices (Performance & Security) (Section 11/15)

A professional e-commerce application must adhere to best practices for speed and user safety.

- **Cross-Site Scripting (XSS) Prevention:** Since the application uses template literals (${...}) to generate HTML dynamically, strict sanitation is mandatory in a full-scale app to ensure user-generated content (e.g., product reviews) cannot execute malicious scripts.
- **Performance Optimization:**
  - **Minification & Bundling:** Combining and compressing the HTML, CSS, and JS files to reduce load time.
  - **Lazy Loading:** Only loading product images and content as the user scrolls them into the viewport.
  - **Resource Hints:** Using <link rel="preload"> to inform the browser about critical resources needed early in the loading process.
- **Accessibility (A11y):** Ensuring the site is usable by people with disabilities (e.g., proper use of ARIA attributes, keyboard navigation support).

12 StateManagementStrategies(BeyondVanillaJS)(Section12/15)

For a complex app, managing data and state across multiple components becomes challenging with plain vanilla JavaScript. Advanced applications move to dedicated frameworks. **Alternative State Management Technologies:**

- **React/Angular/Vue.js:** These frameworks provide component-based architecture and powerful native state management features.
- **Redux/Zustand (for React):** Libraries that centralize the application state into a single store, making state changes predictable and debuggable.
- **Global Context (for React):** A pattern used to share data (like user info, cart state, or Firebase instances) across the entire component tree without prop-drilling.

Moving to one of these frameworks would significantly streamline complex features like checkout flow, personalized recommendations, and sophisticated filtering.

13 Deployment and Hosting Considerations (Section 13/15)

The delivery mechanism is critical for application availability and speed. **Deployment Stack:**

- **Static Asset Hosting (Firebase Hosting):** The HTML, CSS, and JavaScript files are static assets served globally via a Content Delivery Network (CDN). This ensures low latency regardless of the user's location.
- **CI/CD Pipeline:** A Continuous Integration/Continuous Deployment system (e.g., GitHub Actions, Cloud Build) automates the process of testing the code, building the assets, and deploying them to Firebase Hosting upon every code merge, ensuring rapid feature release.
- **Caching Strategy:** Configuring HTTP headers (like Cache-Control) to tell the user's browser how long to store static assets, reducing repeated data transfers.

14 API Design and Microservices (Conceptual Expansion) (Section 14/15)

While Firebase abstracts away much of the API layer, a truly scalable e-commerce platform often uses dedicated APIs for specialized tasks. **Microservice Architecture:**

- **Payment Gateway Service:** A dedicated, secure service (often via Stripe, PayPal) that handles all sensitive payment processing. **The frontend never directly touches credit card data.**
- **Inventory Service:** A standalone service that manages stock levels and reserves items during the checkout process, ensuring atomicity across transactions. This prevents issues like overselling.
- **Recommendation Service:** An AI/ML service that uses user behavior data to suggest relevant products, connecting to the main application via a dedicated API endpoint.

15 Future Enhancements and Scalability (Section 15/15)

To prepare AYUNIK for significant growth and feature expansion, the following steps would be taken:

1. **Introduce Server-Side Rendering (SSR) or Static Site Generation (SSG):** Using Next.js or Nuxt.js to pre-render the product pages. This dramatically improves initial load time and ensures best-in-class SEO.
2. **Scale Firestore:** Implement sharding or use complex data structures to handle millions of simultaneous users and product listings efficiently.
3. **Advanced Analytics:** Integrating tools like Google Analytics 4 (GA4) or Mixpanel to track every user action (product views, add-to-cart clicks, checkout abandonment) to drive business decisions.
4. **Web Workers:** Offloading heavy computational tasks (like complex product filtering or large data processing) from the main browser thread to a background thread to prevent UI freezing.

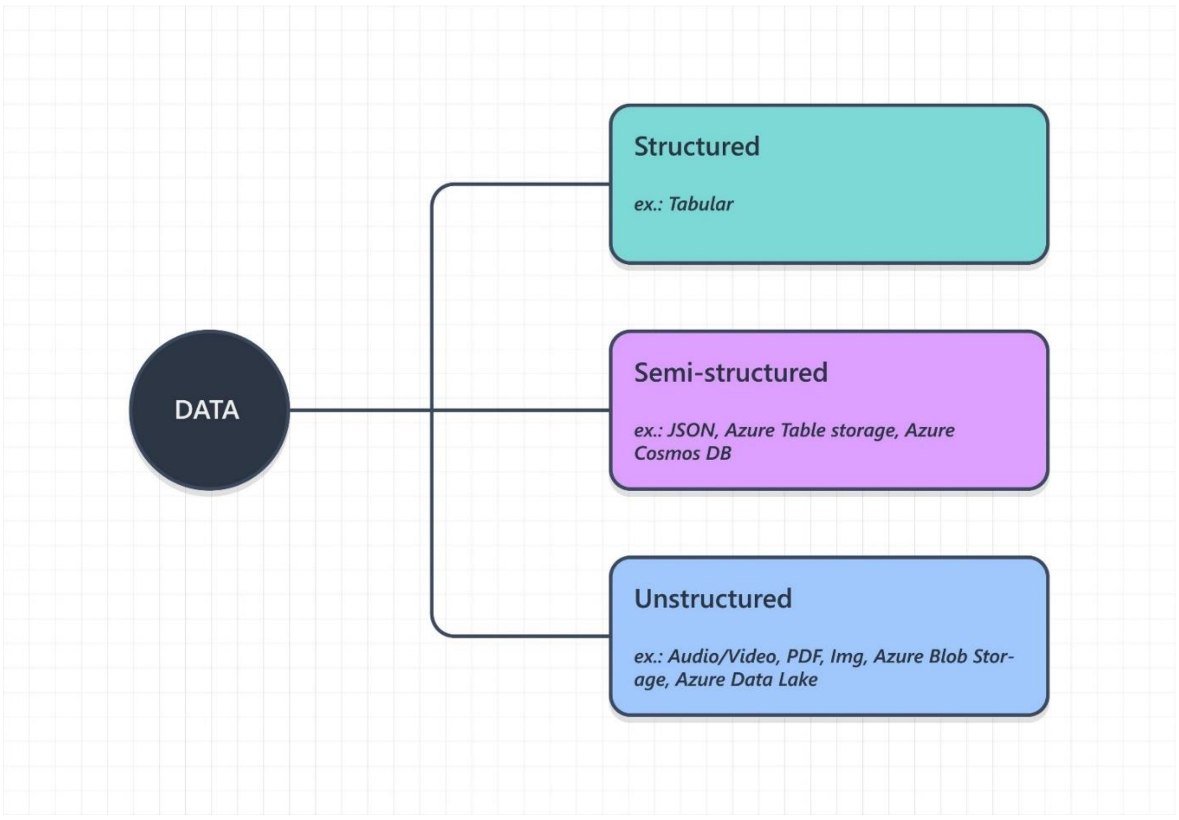
The current structure serves as an excellent foundation, but leveraging these advanced technologies is the path to building a high-performance, globally scalable e-commerce ecosystem.

Data Persistence: Firestore NoSQL Database (Section 7/15)

Firestore is the primary data storage technology for this application. It is a flexible, scalable, cloud-hosted NoSQL database.

Firestore Key Concepts:

- **Collections and Documents:** Data is structured hierarchically. A Collection (e.g., products, users, carts) contains Documents (single records, e.g., a specific product item). Documents contain fields (key-value pairs).
- **Real-time Synchronization (onSnapshot):** This is Firestore's most powerful feature for e-commerce. Instead of manually polling the server, the client establishes a persistent connection. When the data changes on the server (e.g., a product's stock count is updated), the client automatically receives the update instantly, crucial for a live shopping experience.
- **Mandatory Data Structure:** Following security rules, application data must be structured:
  - **Public Data:** Stored in /artifacts/{appId}/public/data/{collection} (e.g., Product Catalog).


DATA
Structured
ex.: Tabular
Semi-structured
ex.: JSON, Azure Table storage, Azure Cosmos DB
Unstructured
ex.: Audio/Video, PDF, Img, Azure Blob Storage, Azure Data Lake