INTRODUCTION ABOUT THE PROJECT

In today's digital-first economy, the retail sector has become highly competitive, with both consumers and retailers seeking efficient ways to connect in the online marketplace. The rise of e-commerce platforms has transformed the shopping experience, providing customers access to a global inventory while enabling businesses to reach a larger, more diverse audience. However, despite the abundance of general online marketplaces, many platforms struggle to offer the specialized, premium user experience that horology enthusiasts and modern shoppers demand. This is where **Shraddha Watch Store** comes in.

Shraddha Watch Store bridges the gap between quality timepieces and the consumers who seek them by offering a comprehensive, feature-rich platform for both shoppers and store administrators. Designed with a focus on usability, scalability, and performance, the platform provides a streamlined approach to online retail. Customers can explore various watch collections across categories such as Analog, Digital, Smartwatches, and Luxury, with personalized product recommendations, advanced search filters, and the ability to save items to a wishlist for later. Administrators can manage product inventories, process orders, and oversee customer interactions with ease.

The goal of **Shraddha Watch Store** is to create a one-stop solution that helps individuals find the perfect timepiece while assisting the business in managing sales efficiently. By leveraging modern technologies and best practices, the platform not only provides a seamless shopping experience but also ensures access to high-quality products that align with users' style and functional needs.

As the retail market becomes increasingly digital and the demand for convenient online shopping continues to rise, is built to meet these evolving consumer habits. Its responsive users on both sides of the transaction.

From a technical perspective, the platform's development is rooted in modern web development principles, utilizing **HTML5, CSS3, JavaScript, PHP, and MySQL** to create a scalable and secure environment. The project follows an Agile development approach, ensuring features are implemented iteratively with continuous feedback. The back-end architecture is built using the Model-View-Controller (MVC) design pattern, ensuring maintainability and ease of future enhancements.

By combining ease of use, performance, and a visually engaging user experience, **Shraddha Watch Store** seeks to stand out in the e-commerce market and become a trusted destination for watch lovers. Whether an individual is looking for a daily beater, a fashion statement, or a luxury investment, **Shraddha Watch Store** offers the tools and variety needed to succeed in today's competitive digital landscape.

OBJECTIVE AND SCOPE OF PROJECT

1) **Objectives**

The primary objective of this project is to develop **Shraddha Watch Store**, a robust and scalable e-commerce platform that digitizes the traditional watch buying experience. The specific objectives are as follows:

- **To centralize a diverse product inventory:** To provide a comprehensive digital catalog that hosts a wide range of watch collections—including analog, digital, smartwatches, and luxury timepieces—making it easier for customers to browse and compare products from different brands in one place.
- **To enhance user engagement through personalization:** To implement intelligent algorithms that offer personalized product recommendations based on user profiles, past purchase history, and browsing behavior, thereby increasing the likelihood of conversion and customer satisfaction.
- **To streamline the purchasing process:** To create a seamless "Add to Cart" and "Wishlist" mechanism that allows customers to easily manage their potential purchases, view total costs in real-time, and proceed to checkout with minimal friction.
- **To empower administrative control:** To provide administrators with a powerful dashboard for managing the product lifecycle. This includes adding new stock, updating prices, managing categories, and monitoring inventory levels to prevent stockouts or overstocking.
- **To ensure data security and trust:** To provide a secure environment for user data by implementing authentication protocols (login/signup) and secure payment processing, ensuring that customer details and transaction histories remain protected.
- **To optimize search and navigation:** To enhance the efficiency of the shopping experience by integrating advanced filtering and search algorithms. This allows users to narrow down products by specific attributes such as brand, strap material, water resistance, price range, and dial color.

2) **Scope**

The scope of the **Shraddha Watch Store** project encompasses the full software development lifecycle of an e-commerce application, focusing on three main modules: the User Interface (Front-end), the Administrator Dashboard (Back-end), and the Database Management System.

1. Customer/User Module Scope:

- **Account Management:** Development of secure registration and login systems, allowing users to create profiles, save shipping addresses, and view their order history.
- **Product Discovery:** Implementation of a responsive product catalog with high-quality images, detailed descriptions, and specifications.

- **Shopping Cart & Wishlist:** Functionalities that allow users to temporarily save items for purchase (Cart) or save them for future consideration (Wishlist), including the ability to modify quantities and remove items.
- **Order Tracking:** A feature enabling users to track the status of their orders from "Processing" to "Shipped" and "Delivered."

2. Administrator Module Scope:

- **Inventory Management:** Tools for the admin to perform CRUD (Create, Read, Update, Delete) operations on products, ensuring the website reflects accurate stock levels and pricing.
- **Order Processing:** A system for administrators to view incoming orders, verify payment status, update shipping details, and manage cancellations or returns.
- **User Management:** The ability to view registered users and manage accounts to maintain platform integrity.

3. Technical Scope:

- **Responsive Design:** Ensuring the website is fully functional and visually appealing across all device types, including desktop monitors, tablets, and mobile smartphones.
- **Secure Transactions:** Integration of reliable payment gateways to facilitate safe online monetary transactions.
- **Database Integrity:** Designing a relational database to efficiently store and retrieve complex data regarding products, users, orders, and transactions without redundancy.

INTRODUCTION OF HTML AND CSS

HTML (Hypertext Mark-up Language) and CSS (Cascading Style Sheets) are essential technologies that form the foundation of web development, enabling the creation of visually engaging and well-structured web pages. Much like Visual Basic in application development, HTML and CSS are integral in defining the structure and presentation of web content.

HTML (Hypertext Mark-up Language):

HTML serves as the fundamental framework of web pages, providing a structured mark-up language to define content and layout. Developers use HTML to create documents by utilizing tags that categorize and organize different elements on a webpage, such as headings, paragraphs, images, links, forms, and more.

CSS (Cascading Style Sheets):

CSS complements HTML by allowing developers to manage the visual styling of web pages. It introduces style rules that dictate how HTML elements should be displayed across various devices. By separating content from design, CSS streamlines the creation of consistent and visually appealing user interfaces.

Together, HTML and CSS provide the essential building blocks for crafting an engaging and user-friendly web experience. While HTML defines the structure, CSS enhances the design, offering a flexible and responsive layout. This synergy supports the principles of Rapid Application Development (RAD), simplifying the process of designing and styling web interfaces efficiently.

INTRODUCTION OF JAVASCRIPT

JavaScript is a versatile and powerful programming language primarily employed in web development. It is a key component of modern web browsers, empowering developers to create dynamic and interactive user interfaces. Often abbreviated as JS, JavaScript plays a pivotal role in enhancing web page functionality through client-side scripting.

Key Features of JavaScript:

- **Client-Side Scripting:**

JavaScript is predominantly used for client-side scripting, meaning it executes directly within the user's web browser. This capability enables developers to craft dynamic content, validate forms, handle events, and manipulate the Document Object Model

(DOM), allowing webpage content to be updated dynamically without the need for a page reload.

- **Object-Oriented:**

JavaScript is an object-oriented language, supporting core concepts such as encapsulation, inheritance, and polymorphism. These features allow developers to organize and modularize their code effectively, facilitating better code management and scalability.

- **Interactivity and Dynamic Content:**

JavaScript enhances web pages by enabling real-time responsiveness to user interactions. Developers can implement features like sliders, pop-ups, and form validations, while also facilitating dynamic content loading and updating, contributing to more engaging and user-friendly websites.

- **Event-Driven Programming:**

JavaScript follows an event-driven programming model, wherein developers define functions that execute in response to specific events such as button clicks, mouseovers, or form submissions. This paradigm improves user experience by making web pages more interactive and responsive to user actions.

- **Cross-Browser Compatibility:**

JavaScript is supported by all major web browsers, including Chrome, Firefox, Safari, and Edge, ensuring consistent functionality across diverse platforms and devices.

JavaScript is an indispensable technology in modern web development, frequently working alongside HTML and CSS to build interactive web applications. Over time, JavaScript has expanded beyond its initial use in web browsers, now also playing a significant role in server-side development (via Node.js), mobile app development, and other application domains.

INTRODUCTION OF PHP

PHP is a general-purpose scripting language primarily designed for web development. Initially created by Danish-Canadian programmer Rasmus Lerdorf in 1993 and officially released in 1995, PHP's reference implementation is now maintained by The PHP Group. Originally an acronym for "Personal Home Page," PHP now stands for the recursive term "PHP: Hypertext Preprocessor."

PHP code is typically processed on a web server through a PHP interpreter, which can be implemented as a module, daemon, or Common Gateway Interface (CGI) executable. The result of the interpreted PHP code—whether HTML, binary image data, or other types of

output—forms part of the HTTP response. Various web template systems, content management systems, and frameworks are available to streamline and facilitate this process.

While PHP is most commonly associated with web development, it is also capable of handling non-web-based programming tasks such as creating standalone graphical applications and controlling robotic drones. PHP code can also be executed directly from the command line, expanding its versatility.

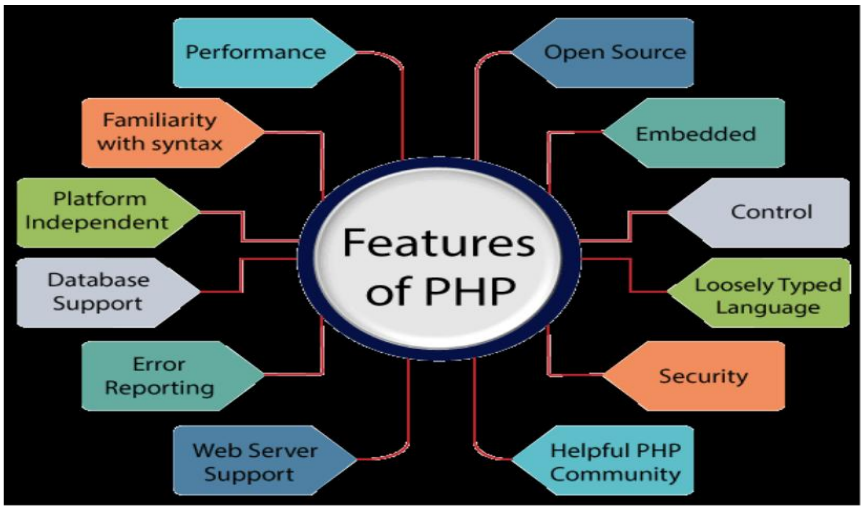
The standard PHP interpreter, powered by the Zend Engine, is open-source software released under the PHP License. PHP is widely portable and compatible with most web servers across various operating systems and platforms. Although PHP lacked a formal specification until 2014, its original implementation served as the de facto standard for

other implementations to follow. Since 2014, efforts have been undertaken to create a formal specification for PHP.

According to W3Techs, as of January 2023, PHP powers 77.8% of websites with known server-side programming languages. However, it is concerning that a significant portion of PHP users continue to utilize unsupported versions such as PHP 7 and even PHP 5, both of which no longer receive security updates and are known to have critical vulnerabilities.

PHP remains a powerful and reliable language for server-side scripting, with applications ranging from form data handling and dynamic content generation to database management, session handling, cookie management, and email processing. Furthermore, PHP offers numerous hash functions for encrypting user data, ensuring its security and robustness as a server-side scripting language. These capabilities make PHP an indispensable tool for modern web development.

Features of PHP:



- **Open Source:**

PHP is open-source software, meaning its source code is freely available. Developers can modify and develop PHP versions to suit their specific needs without any associated costs. All components of PHP are open for free download and use.

- **Familiarity with Syntax:**

PHP has a straightforward and easily understandable syntax, making it accessible for programmers of varying skill levels. The simplicity of the syntax allows for efficient coding and quick adoption by developers.

- **Embedded:**

PHP code can be seamlessly embedded within HTML tags and other scripts, enabling developers to mix PHP with standard HTML code effortlessly, enhancing the flexibility of web development.

- **Platform Independence:**

PHP is platform-independent, meaning it can be deployed across multiple operating systems such as Windows, macOS, Linux, and UNIX. A PHP application developed on one operating system can be executed on another without modification.

- **Performance:**

PHP executes scripts faster than many other programming languages, such as JSP and ASP. This is due to PHP's ability to use its own memory management, which reduces the server's workload and loading times, ultimately improving processing speed and overall performance.

- **Open Source:**

PHP is open-source software, meaning its source code is freely available. Developers can modify and develop PHP versions to suit their specific needs without any associated costs. All components of PHP are open for free download and use.

- **Familiarity with Syntax:**

PHP has a straightforward and easily understandable syntax, making it accessible for programmers of varying skill levels. The simplicity of the syntax allows for efficient coding and quick adoption by developers.

- **Embedded:**

PHP code can be seamlessly embedded within HTML tags and other scripts, enabling developers to mix PHP with standard HTML code effortlessly, enhancing the flexibility of web development.

- **Platform Independence:**

PHP is platform-independent, meaning it can be deployed across multiple operating systems such as Windows, macOS, Linux, and UNIX. A PHP application developed on one operating system can be executed on another without modification.

- **Database Support:**

PHP supports a wide range of databases, including MySQL, SQLite, ODBC, and others. This makes it a versatile language for applications requiring database interaction.

- **Error Reporting:**

PHP provides predefined error reporting constants, such as E_ERROR, E_WARNING, E_STRICT, and E_PARSE, that allow developers to generate error notices or warnings during runtime. This feature is essential for debugging and improving code reliability.

- **Loosely Typed Language:**

PHP is a loosely typed language, meaning that variables do not require explicit type declarations. The data type is automatically determined at runtime based on the value assigned to the variable, which simplifies development and reduces errors.

- **Web Server Support:**

PHP is compatible with a wide array of web servers, including Apache, Netscape, and Microsoft IIS, allowing developers to choose their preferred server environment without compatibility concerns.

- **Security:**

PHP provides robust security features designed to protect websites from threats and malicious attacks. Multiple layers of security are integrated into PHP, making it a reliable choice for developing secure websites.

- **Control:**

PHP is known for its efficiency and control, as it allows developers to accomplish complex tasks with relatively few lines of code. Additionally, PHP offers extensive control over website functionality, enabling developers to make quick adjustments and improvements when needed.

- **A Helpful PHP Community:**

PHP benefits from a large and active community of developers who regularly contribute to documentation, tutorials, FAQs, and other helpful resources. The wealth of community-driven knowledge makes learning PHP easier and more accessible for both beginners and experienced developers.

INTRODUCTION OF MYSQL

What is a Database?

A database is a specialized application designed to store, manage, and organize data in a structured manner. It provides distinct APIs that allow users to create, access, manage, search, and replicate the data it contains. While other types of data storage systems, such as file systems or large hash tables in memory, can be used, they generally do not offer the same speed and ease of data retrieval and manipulation as a dedicated database system.

In modern applications, relational database management systems (RDBMS) are commonly used to handle large volumes of data. The term "relational" refers to the way data is

organized into tables, where relationships between the data are established using primary keys and foreign keys.

Relational Database Management System (RDBMS)

An RDBMS is software that facilitates the implementation and management of relational databases. It enables the creation of databases with tables, columns, and indexes while ensuring the following functions:

- **Table Structure:** Allows the creation and organization of tables to store data.
- **Referential Integrity:** Maintains the integrity of relationships between tables using primary and foreign keys.
- **Index Management:** Automatically updates indexes to optimize data retrieval.
- **SQL Query Processing:** Interprets SQL queries and retrieves data by combining information from various tables.

RDBMS Terminology

Before exploring specific database systems like MySQL, it is important to understand some key terminology used in RDBMS:

- **Database:** A collection of related tables that store data.
- **Table:** A table is a collection of rows and columns, representing data in a structured format similar to a spreadsheet.
- **Column:** A column contains data of a specific type or category, such as customer names, addresses, or product prices.
- **Row:** A row (also known as a tuple, entry, or record) is a set of related data within a table, often representing an individual record.
- **Redundancy:** The practice of storing data multiple times to improve system performance. While redundancy can speed up access, it may introduce challenges in data consistency.
- **Primary Key:** A primary key is a unique identifier for each record in a table. It ensures that each row in a table is distinguishable from others, and no two rows can have the same primary key value.
- **Foreign Key:** A foreign key is used to create a relationship between two tables. It refers to the primary key of another table, ensuring referential integrity between the two tables.
- **Compound Key (Composite Key):** A compound key is a primary key that consists of two or more columns, used when a single column cannot uniquely identify records in a table.
- **Index:** An index in a database is a data structure that allows for faster retrieval of data, similar to an index at the back of a book. It speeds up query execution by reducing the amount of data that needs to be scanned.

- **Referential Integrity:** Referential integrity ensures that foreign key values in a table always point to valid, existing rows in another table, preventing orphaned records and maintaining data consistency.

What is SQL?

SQL (Structured Query Language) is the standard language used for managing and manipulating data in a relational database management system (RDBMS). It provides an interface for accessing, updating, deleting, and managing database structures and data. SQL is the foundation of interacting with databases, enabling users to perform a wide range of operations on the data they store.

SQL is divided into four main subsets:

1. **DDL (Data Definition Language):** DDL is used to define and manage the structure of a database. It allows the creation, alteration, and deletion of database objects such as tables, views, and schemas. Key commands include:
    - **CREATE**: To create new database objects.
    - **ALTER**: To modify existing database objects.
    - **DROP**: To delete database objects.
2. **DML (Data Manipulation Language):** DML deals with the manipulation of data within the database. It allows for adding, updating, deleting, and querying data. Common DML commands include:
    - **SELECT**: To retrieve data from one or more tables.
    - **INSERT**: To add new data into a table.
    - **UPDATE**: To modify existing data.
    - **DELETE**: To remove data from a table.

3. **DCL (Data Control Language):** DCL is used to control access to data within the database. It helps manage user permissions, allowing for secure access and data management. Main DCL commands include:
    - **GRANT**: To give a user specific access rights to database objects.
    - **REVOKE**: To remove previously granted permissions.
4. **TCL (Transaction Control Language):** TCL is used to manage transactions in a database. A transaction is a sequence of operations performed as a single unit. TCL ensures that transactions are handled properly, preserving data integrity. Key TCL commands include:
    - **COMMIT**: To save all changes made during the transaction.
    - **ROLLBACK**: To undo changes made during the transaction if an error occurs.
    - **SAVEPOINT**: To set a point within a transaction to which you can roll back.
    - **SET TRANSACTION**: To configure transaction properties.

MySQL

MySQL is a widely used relational database management system (RDBMS) developed and maintained by MySQL AB, a Swedish company. It is known for being fast, reliable, and easy to use, making it a popular choice for both small businesses and large enterprises. MySQL has several advantages that contribute to its growing popularity:

- **Open Source:** MySQL is released under an open-source license, meaning it is free to use and modify.

- **Powerful:** Despite being free, MySQL provides a range of features that rival some of the most expensive commercial database management systems. It can handle substantial databases and a variety of data processing tasks

- **Standardized SQL Support:** MySQL uses a standard form of SQL, making it compatible with most database applications.

- **Cross-Platform Compatibility:** MySQL can run on various operating systems, including Windows, Linux, and macOS, and supports multiple programming languages like PHP, PERL, C, C++, and Java.

- **Speed:** MySQL is known for its quick performance, especially with large datasets. It is optimized for high-speed operations.

- **Scalability:** MySQL can handle large databases with ease. It supports up to 50 million rows per table, with the theoretical file size limit reaching 8 million terabytes.

- **PHP Integration:** MySQL is often used alongside PHP, making it a popular choice for web development.

- **Customization:** Being open source, MySQL can be customized and extended to fit specific needs. Developers can modify the software to suit their particular environment or requirements.

DEFINITION OF PROBLEM FOR SHRADDHA WATCH STORE

The **Shraddha Watch Store** project addresses several critical challenges related to e-commerce functionality, user experience, and backend operations. These problems span across multiple areas such as search efficiency, inventory management, transaction security, and customer support, all of which impact the platform's overall effectiveness and satisfaction for both customers and administrators. Below are the primary problems the system aims to resolve:

1. User Experience and Interface:

- Ensuring intuitive navigation for customers to easily browse through diverse watch categories (Analog, Digital, Smartwatch).
- Addressing cluttered or complex design elements that could hinder the "Add to Cart" and checkout processes, leading to cart abandonment.

2. Product Search and Filter Functionality:

- Ensuring accurate, relevant, and fast search results based on user queries (e.g., searching for a specific brand or model number).
- Optimizing filtering mechanisms to allow customers to easily refine results based on price range, strap material, dial color, and movement type.

3. Inventory Accuracy and Maintenance:

- Keeping product listings updated and free from errors regarding stock availability and pricing to avoid customer frustration.
- Implementing a streamlined process for administrators to add new watches, update specifications, and manage stock levels in real-time.

4. Database Management and Performance:

- Ensuring that the database can handle a large volume of high-resolution product images and user data without compromising page load speeds.
- Optimizing database queries to avoid slowdowns and maintain quick response times, especially during high-traffic sales periods.

5. Security and Privacy:

- Protecting sensitive customer data, including shipping addresses, contact numbers, and order history.
- Ensuring secure handling of authentication credentials to prevent unauthorized account access.

6. Payment Gateway and Transaction Handling:

- Addressing potential billing discrepancies or payment failures during the checkout process.
- Ensuring a seamless integration with payment gateways to handle various payment modes (UPI, Cards, Net Banking) without errors.

7. Order Communication and Notifications:

- Streamlining the notification system to ensure customers receive timely updates regarding their order status (Placed, Shipped, Delivered).
- Addressing issues with delayed or lost order confirmations that could negatively impact customer trust.

8. Mobile Compatibility and Responsiveness:

- Ensuring a seamless mobile shopping experience for users accessing the store via smartphones or tablets, as a significant portion of e-commerce traffic is mobile-based.
- Maintaining layout integrity and touch-friendly interfaces across different screen sizes.

9. System Downtime and Reliability:

- Minimizing downtime or service interruptions that directly result in lost sales and revenue.
- Optimizing backend architecture to handle concurrent users efficiently to prevent server crashes.

10. Customer Support and Returns:

- Providing a structured mechanism for users to report issues with delivered products or request returns/exchanges.
- Reducing response times for admin intervention when customers face technical difficulties with their accounts or orders.

In sum, **Shraddha Watch Store** needs to address the above problems in a systematic and comprehensive manner to provide a smooth, secure, and effective online shopping platform. These challenges require a balanced approach to enhance the shopping experience, streamline administrative operations, and ensure the scalability of the platform for future business growth.

SYSTEM ANALYSIS

System analysis for **Shraddha Watch Store** involves understanding the requirements of both customers and store administrators. The system must provide a seamless online shopping experience while ensuring data security and efficient management of product inventory and orders. The system analysis phase includes gathering requirements through market research, analyzing similar e-commerce platforms, and identifying key functionalities that will make **Shraddha Watch Store** stand out in the competitive retail market. Here is the System Analysis for **Shraddha Watch Store** in points:

1. Problem Identification:

- Modern e-commerce platforms often lack niche-specific filtering and fail to provide an intuitive user experience for browsing complex categories like luxury or smartwatches.
- Administrators often face difficulties managing inventory levels accurately, while customers struggle to find specific watch models quickly due to poor search optimization.

2. User Requirements:

- **Customers:** Need a platform that provides high-quality product imagery, personalized recommendations, the ability to save items to a Wishlist, an easy-to-use Shopping Cart, and a responsive interface for mobile shopping.
- **Administrators:** Require features for listing new products with detailed specifications, managing stock levels, processing customer orders, and tracking sales data.

3. System Requirements:

- **Functional:**
    - User authentication (Login/Signup) and profile management (Address Book).
    - Product listing and search functionalities with advanced filtering options (Brand, Price, Strap Material).
    - "Add to Cart" and "Wishlist" features for customers to manage potential purchases.
    - Order management system for administrators to update shipping status (Pending, Shipped, Delivered).
- **Non-Functional:**
    - **Scalability:** The ability to handle a growing inventory and increasing number of concurrent users.
    - **Security:** robust protection of user data, especially sensitive information like passwords, contact details, and transaction history.
    - **Performance:** Optimization to ensure fast page load times (especially for image-heavy pages) and smooth navigation.

4. System Architecture:

- The system is based on a **Model-View-Controller (MVC)** architecture, separating the business logic from the user interface for better maintainability and scalability.
- The **Front-end** is developed using **HTML5, CSS3, and JavaScript**, providing a responsive and visually engaging experience across devices.
- The **Back-end** is developed using **PHP and MySQL**, enabling secure data storage, retrieval, and dynamic content management.

5. Data Flow:

- The system manages user input (such as product searches, cart additions, or order placements) through secure forms and interfaces, processes the data via PHP, and retrieves or stores the required information in the MySQL database.
- The MVC pattern ensures that data handling is done independently from the user interface, enhancing system modularity and making future updates easier.

6. User Interface and Experience:

- The system emphasizes visual appeal and ease of use, with intuitive navigation, clear product categorization, and minimal steps for customers to complete a purchase.
- **AJAX** is used to update data asynchronously (e.g., updating cart totals or filtering products), providing a smoother user experience without the need for full page reloads.

7. Security Considerations:

- The system is designed to protect user data with **encrypted passwords** (hashing) and **prepared SQL statements** to prevent SQL injection attacks.
- **SSL (Secure Sockets Layer)** is recommended for secure data transmission between the user's browser and the server, ensuring trust during transactions.

SYSTEM REQUIREMENTS

Minimum Hardware Requirements for Development

- **CPU:** Intel Core i3 or equivalent (modern processors provide better performance).
- **Memory (RAM):** 4 GB (2 GB is minimal and may not be sufficient for multitasking during development).
- **Hard Disk:** 512 GB (256 GB is minimal; additional storage is recommended for development flexibility).
- **System Type:** 32-bit or 64-bit Operating System (64-bit is preferred for handling more memory and compatibility with modern tools).

Minimum Software Requirements for Development

- **Operating System:**
  - **Windows:** Windows 10 or 11.
  - **Linux:** A recent version (e.g., Ubuntu 20.04 LTS or later).
- **Database:** MySQL 5.7 or later.
- **Server:** Apache (using XAMPP or a similar package for local development).
- **Scripting Language:** PHP 7.4 or later.
- **Frontend Technologies:** HTML5, CSS3, JavaScript (including jQuery and AJAX).
- **IDE:** Visual Studio Code or PHPStorm.
- **Screen Resolution:** 1920x1080 or higher (for better visibility and ease of development; 1024x768 is minimal).

Minimum Hardware and Software Requirements for Running the Website

On PCs (Desktops and Laptops)

- **CPU:** Intel Core i3 or equivalent (sufficient for running typical web applications).
- **Memory (RAM):** 4 GB or higher (more memory will improve performance, especially with higher traffic).
- **Hard Disk:** 256 GB or higher (adequate for website caching and data storage).
- **System Type:** 32-bit or 64-bit Operating System (64-bit is preferred for modern software compatibility and better performance).
- **Operating System:**
  - **Windows:** Windows 7 or later (Windows 10 or 11 preferred).
  - **Linux:** Ubuntu 16.04 LTS or later (e.g., Ubuntu 20.04 LTS or later).
- **Web Browser:** Modern browsers supporting HTML5, CSS3, and JavaScript (e.g., Google Chrome, Mozilla Firefox, Microsoft Edge, Safari).

On Mobile Devices (Phones and Tablets)

- **CPU:** Quad-core processor (e.g., Snapdragon 425 or equivalent) for smoother performance.
- **Memory (RAM):** 2 GB or higher (1 GB might be too limited for modern e-commerce interfaces).

- **Storage:** 16 GB or higher (for storing app data and caching content).
- **Operating System:**
  - **Android:** Version 8.0 (Oreo) or later.
  - **iOS:** Version 13 or later.
- **Browser:**
  - **Android:** Google Chrome or Mozilla Firefox (latest versions).
  - **iOS:** Safari (latest version) or Google Chrome (latest version).
- **Screen Resolution:** 720x1280 pixels or higher (HD resolution for adequate display of product images and content).
- **Internet Connection:** Reliable 4G or higher network speed (to ensure smooth access, image loading, and interaction with the website).

SYSTEM DESIGN AND CODING

INTRODUCTION:

Software design is a critical phase in the software engineering process for **Shraddha Watch Store**, laying the foundation for all subsequent development activities. It transcends development paradigms and application areas, serving as a common thread in the creation of every engineered system. Design is initiated after the specific requirements for the watch e-commerce platform have been specified and analyzed. This step forms the core of the development phase and precedes coding and testing. The primary goal of software design is to create a model or representation of the system that will eventually be implemented as a functional online store.

The central focus of the design is quality. Through the design process, the quality of **Shraddha Watch Store** is ensured by translating customer and administrator requirements into a structured framework that can be translated into code (PHP/MySQL). A robust design minimizes the risk of building unstable systems, ensuring the platform does not collapse under high traffic, remains easy to test, and meets the quality standards required for secure financial transactions.

In the design phase, detailed representations of data structures (such as product inventories and user profiles), program structures, and procedural steps are refined, reviewed, and documented. From both technical and project management perspectives, system design plays a pivotal role in guiding the engineering process toward the successful launch of the website.

INPUT DESIGN:

Input design is the process of converting user-oriented input into a computer-readable format. For **Shraddha Watch Store**, input design must be executed with meticulous attention to detail, as it involves capturing sensitive customer data and critical product information. The collection of input data often represents one of the most resource-intensive aspects of the system development process. Effective input design focuses on optimizing the methods of input collection—such as registration forms, checkout pages, and search bars—while ensuring the highest possible level of accuracy and usability.

The primary objectives of input design for the store are as follows:

1. **Cost-Effectiveness:** Create an input process that is efficient, reducing the time users spend filling out forms during checkout.
2. **Accuracy:** Achieve the highest possible accuracy in input data (e.g., validating shipping addresses and payment details) to minimize errors and delivery failures.
3. **User-Friendliness:** Ensure that the input process is intuitive, understandable, and straightforward for customers, facilitating seamless interaction with the website.

Input Data Considerations

When designing input data for **Shraddha Watch Store**, the goal is to create a process that is easy, logical, and error-free. The data entry operators (administrators adding watches) and end-users (customers) must be clear on the allocated space for each field, the correct sequence of

fields, and any required formatting. A well-designed input form provides these instructions explicitly to avoid ambiguity.

The input process typically involves the following stages:

- **Data Recording:** Capturing raw data from users (e.g., a customer entering their name and address during signup).
- **Data Transcription:** Not applicable in this direct digital entry system.
- **Data Conversion:** Converting user inputs into a format that the PHP backend can process.
- **Data Verification:** Checking the accuracy of input data against predefined criteria (e.g., ensuring email formats are valid).
- **Data Control:** Ensuring the integrity and security of the data throughout the process (e.g., sanitizing inputs to prevent SQL injection).
- **Data Transmission:** Transmitting data securely to the server for processing via SSL.
- **Data Correction:** Allowing users to correct errors in the input data (e.g., editing a shipping address before final payment).

A fundamental aspect of input design is to select data capture methods that reduce the number of stages, thereby decreasing the chances of errors. Input types for the watch store can be classified as:

- **External Input:** Data sourced from external entities, such as customers entering credit card details or search queries.
- **Internal Input:** Data generated internally, such as Order IDs or transaction timestamps.
- **Operational Input:** Data required for system operations, such as admin credentials for logging in.
- **Computerized Input:** Input processed by the system automatically, such as calculating total cart value based on selected items.
- **Interactive Input:** Input provided by users in real-time, such as selecting filters (e.g., "Analog Watches" or "Price: Low to High") to refine product views.

OUTPUT DESIGN:

Output design refers to the process of determining how the results of processing will be communicated to the users. For **Shraddha Watch Store**, outputs play a crucial role in conveying meaningful information, whether it represents a product catalog for customers or sales reports for administrators. Designing effective output involves organizing the presentation of watch specifications, images, and pricing in a manner that is clear, actionable, and accessible.

The outputs of the system are typically categorized as follows:

1. **External Outputs:** Information sent outside the system to users, such as Order Confirmation Emails, Digital Invoices, and Shipping Notifications.
2. **Internal Outputs:** Information generated within the system for internal use, such as Inventory Status Reports and Sales Analytics for the admin.
3. **Operational Outputs:** Outputs produced as part of routine system operations, such as error messages during failed logins or payment gateway status codes.

4. **Interactive Outputs:** Outputs that respond dynamically to user queries, such as search results displaying specific watch models or real-time stock availability (e.g., "Only 2 left in stock").
5. **Turnaround Outputs:** Outputs used by the system for processing and returned for further action, such as a "Review your Order" page before final submission.

The key to designing effective outputs for **Shraddha Watch Store** is ensuring that the right information is presented to the appropriate users in the right format. Outputs should be structured to provide clear communication that facilitates purchase decisions (for customers) and business management (for admins).

The design of screens and interfaces plays a critical role in making the outputs not only informative but also interactive. Users should be able to navigate through the watch collections, request detailed product views, and fulfill their needs by querying the system effectively. This process is vital for ensuring that users can access the data they require in an intuitive and seamless manner.

CONCLUSION

In the software engineering process for **Shraddha Watch Store**, system design, including both input and output design, is foundational for building a quality, reliable, and user-friendly e-commerce platform. Input design ensures that customer and product data is captured accurately and efficiently, while output design guarantees that users receive product details and order information in a well-structured and understandable manner. These steps are critical for the success of the online store, as they directly impact both the functionality and usability of the final product. By focusing on these design principles, developers can ensure that **Shraddha Watch Store** meets user needs, operates efficiently, and remains scalable for future growth

DATABASE DESIGN

TABLE 1: Address

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
user_id	int	NO		NULL	
full_name	varchar(100)	YES		NULL	
house_no	varchar(100)	YES		NULL	
area	varchar(200)	YES		NULL	
city	varchar(100)	YES		NULL	
pincode	varchar(10)	YES		NULL	
mobile	varchar(15)	YES		NULL	
is_default	tinyint(1)	YES		1	

TABLE 2: Admin

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
name	varchar(255)	NO		NULL	
email	varchar(255)	NO		NULL	
password	varchar(255)	NO		NULL	

TABLE 3: Cart

Field	Type	Null	Key	Default	Extra
cart_id	int	NO	PRI	NULL	auto_increment
user_id	int	NO		NULL	
product_id	int	NO		NULL	
quantity	int	NO		1	
price	decimal(10,2)	NO		NULL	
added_at	timestamp	NO		CURRENT_TIMESTAMP	DEFAULT_GENERATED

TABLE 4: Order_Items

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
order_id	int	NO		NULL	
product_id	int	NO		NULL	
quantity	int	NO		NULL	
price	int	NO		NULL	

TABLE 5: Orders

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
user_id	int	NO		NULL	
total_amount	int	NO		NULL	
address_id	int	NO		NULL	
payment_method	varchar(50)	YES		NULL	
created_at	timestamp	NO		CURRENT_TIMESTAMP	DEFAULT_GENERATED
status	enum('confirmed','processed','shipped','out_for...	NO		confirmed	
processed_at	datetime	YES		NULL	
shipped_at	datetime	YES		NULL	
out_for_delivery_at	datetime	YES		NULL	
delivered_at	datetime	YES		NULL	

TABLE 6: Reviews

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
product_id	int	NO	MUL	NULL	
rating	int	NO		NULL	
headline	varchar(255)	YES		NULL	
review_text	text	NO		NULL	
created_at	timestamp	NO		CURRENT_TIMESTAMP	DEFAULT_GENERATED

TABLE 7: Users

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
name	varchar(50)	NO		NULL	
email	varchar(50)	NO		NULL	
password	varchar(255)	NO		NULL	
phone	int	NO		NULL	

TABLE 8: Watches

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
title	varchar(255)	YES		NULL	
price	int	YES		NULL	
image	varchar(255)	YES		NULL	
description	varchar(255)	NO		NULL	
brand	varchar(255)	NO		NULL	
category	varchar(10)	NO		NULL	
quantity	int	NO		NULL	
orders_count	int	YES		0	

DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is a graphical tool used to describe and analyze the flow of data through the **Shraddha Watch Store** system. It focuses on the data flowing into the system, the processes that transform this data, and the data stores where information is kept. By visualizing these flows, we can ensure that the logical requirements of the e-commerce platform are met efficiently.

DFDs are of two types:

1. Physical DFD:

The physical DFD is a model of the current system (if any exists) or the physical implementation of the new system. It is used to ensure that the actual flow of documents and data in the real world is clearly understood.

2. Logical DFD:

The logical DFD is a model of the proposed **Shraddha Watch Store** system. It clearly shows the requirements on which the new system should be built, focusing on *what* the system does rather than *how* it does it technically.

Notation Used In DFD:

There are four simple notations used to complete DFDs:

- **Data Flow:**

  Represented by an arrow ($\rightarrow$), this indicates the direction of data movement. It connects processes, external entities, and data stores. For example, a "Product Order" flows from the Customer to the Order Processing system.

- **External Entity:**

  Represented by a rectangle or square, this denotes a source or destination of data outside the system boundaries. In **Shraddha Watch Store**, the key external entities are the **Customer** (who places orders) and the **Administrator** (who manages inventory).

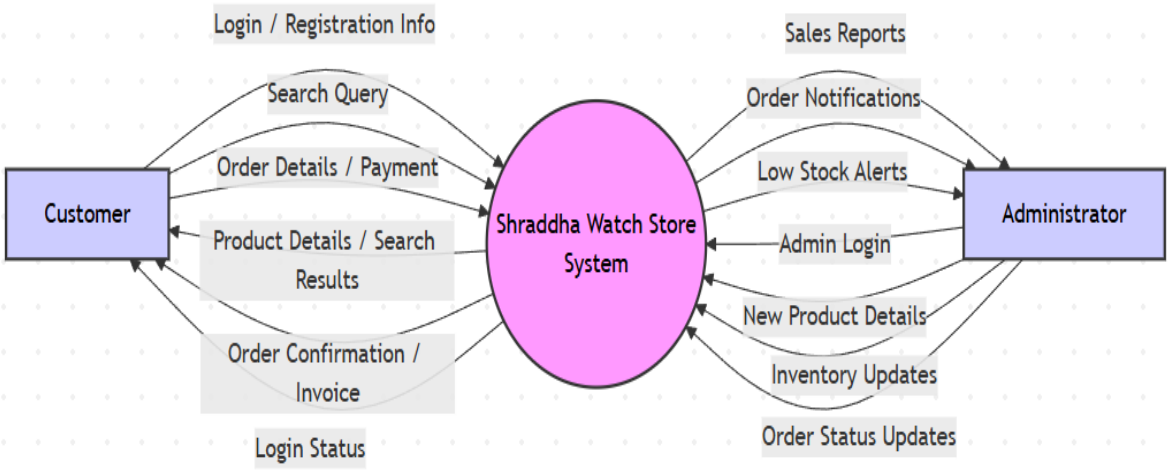
- **Process:**

  Represented by a circle or a rounded rectangle, this denotes a function or transformation that converts inputs into outputs. Examples include "Verify Login," "Calculate Total Price," or "Update Inventory."

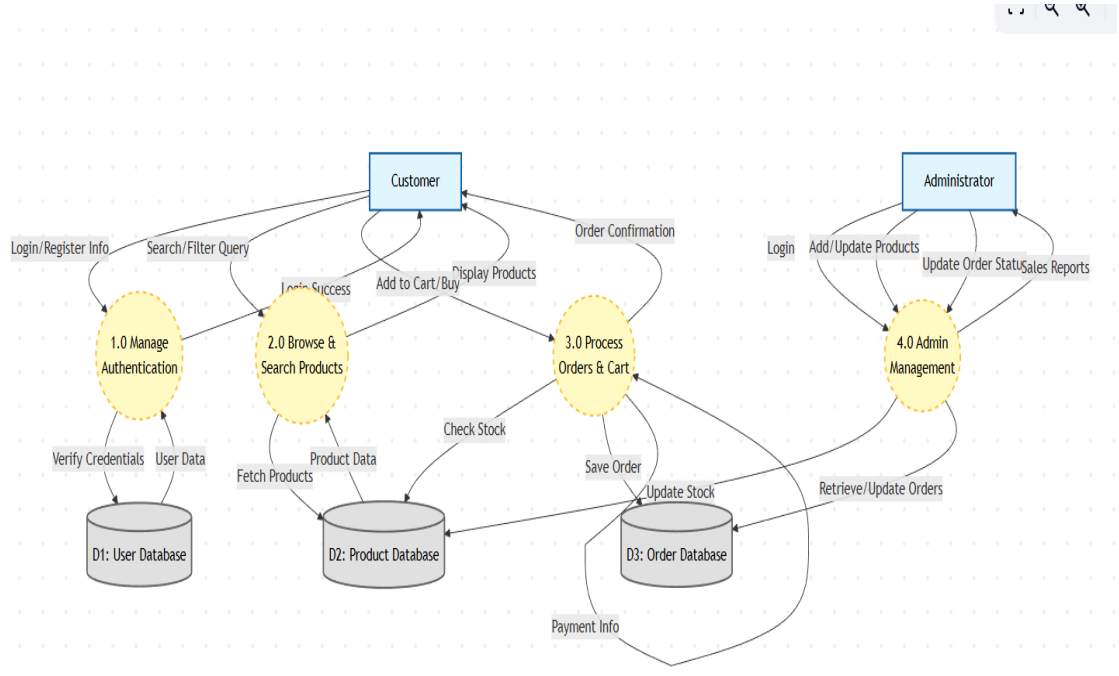
- **Data Store:**

Represented by an open-ended rectangle or parallel lines ($=$), this indicates a repository where data is stored for later use. Examples include the **Product Database**, **User Profiles**, and **Order History**.

DFD Level 0:



DFD Level 1:



Here is the **Entity Relationship Diagram (ERD)** section for **Shraddha Watch Store**, including the definitions and the diagram code.

ENTITY RELATIONSHIP DIAGRAM

An Entity Relationship Diagram (ERD) is a graphical representation that depicts the logical structure of a database. It visualizes the relationships between different entities (tables) within the **Shraddha Watch Store** database, ensuring that data is organized efficiently and without redundancy.

The core components of an E-R Diagram are:

1. Entity: An entity is a real-world object or concept that is distinguishable from all other objects. In the context of **Shraddha Watch Store**, key entities include:

- **User:** Represents the customer or admin accessing the system.
- **Product:** Represents the watches available for sale.
- **Order:** Represents a purchase made by a user.
- **Category:** Represents the classification of watches (e.g., Analog, Digital).

2. Attribute: Attributes are the specific properties or details that describe an entity. For example:

- **User Entity:** Name, Email, Password, Address.
- **Product Entity:** Watch Name, Brand, Price, Stock Quantity.

3. Relationship: A relationship is an association among several entities. For example:

- A **User** *places* an **Order**.
- An **Order** *contains* multiple **Products**.
- A **Product** *belongs to* a **Category**.

4. Weak Entity: A Weak Entity is an entity that does not have a primary key of its own. It cannot be uniquely identified by its own attributes alone and depends on the existence of another entity (known as the "Owner" or "Strong" entity).

- **Symbol:** Represented by a double-bordered rectangle.
- **Example in Shraddha Watch Store:** The **Order Details** (or Order Items) entity is a weak entity. It cannot exist without a corresponding **Order**. If an order is deleted, the specific details (like "2 units of Casio G-Shock") associated with that order are also removed.

5. Attributes: An attribute is a property or characteristic of an entity. It describes the data we want to store for each entity.

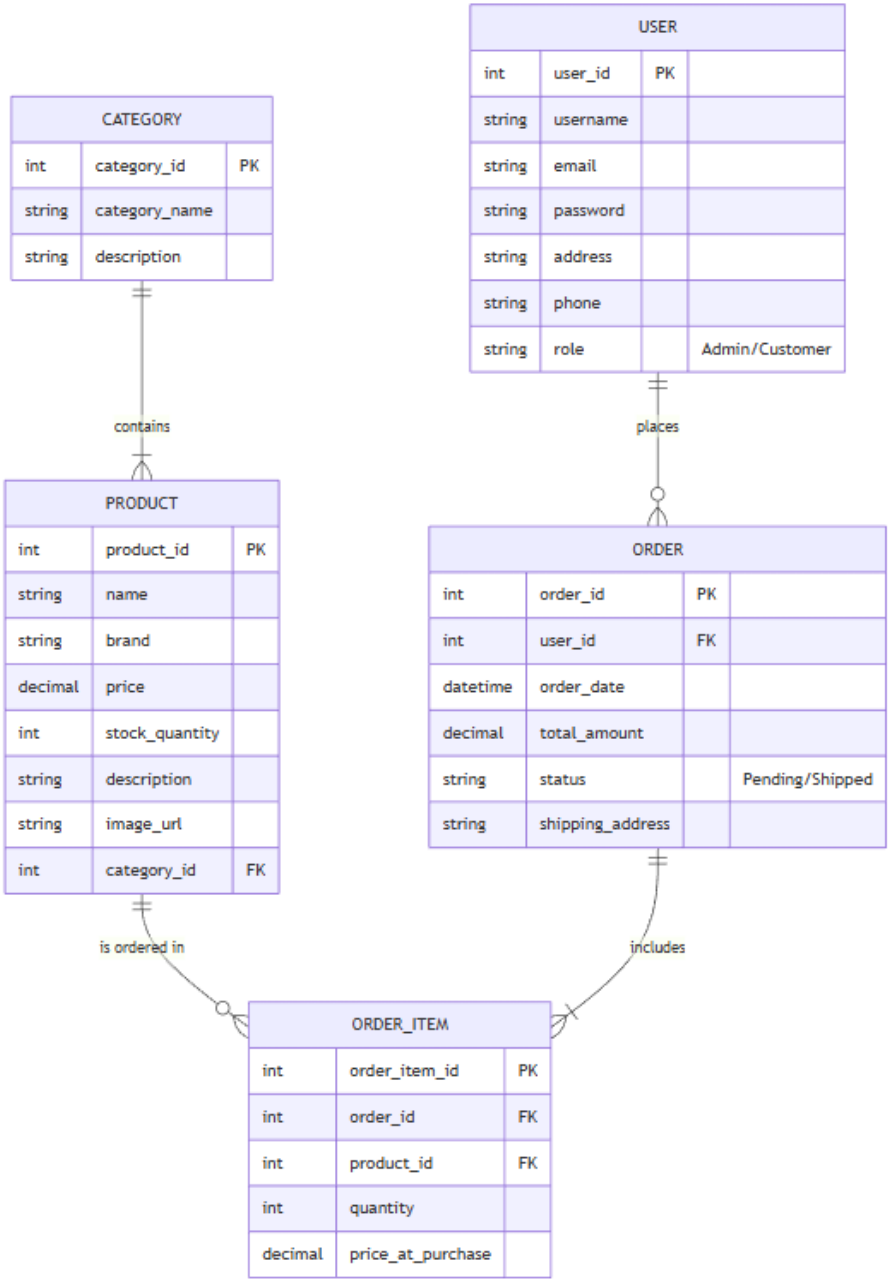
- **Symbol:** Represented by an oval or ellipse.

- **Example in Shraddha Watch Store:**
    - For the **Product (Watch)** entity, attributes include: `Watch_ID`, `Brand_Name`, `Price`, `Dial_Color`, and `Strap_Material`.
    - For the **User** entity, attributes include: `User_ID`, `Email`, `Phone_Number`, and `Shipping_Address`.

Types of Attributes:

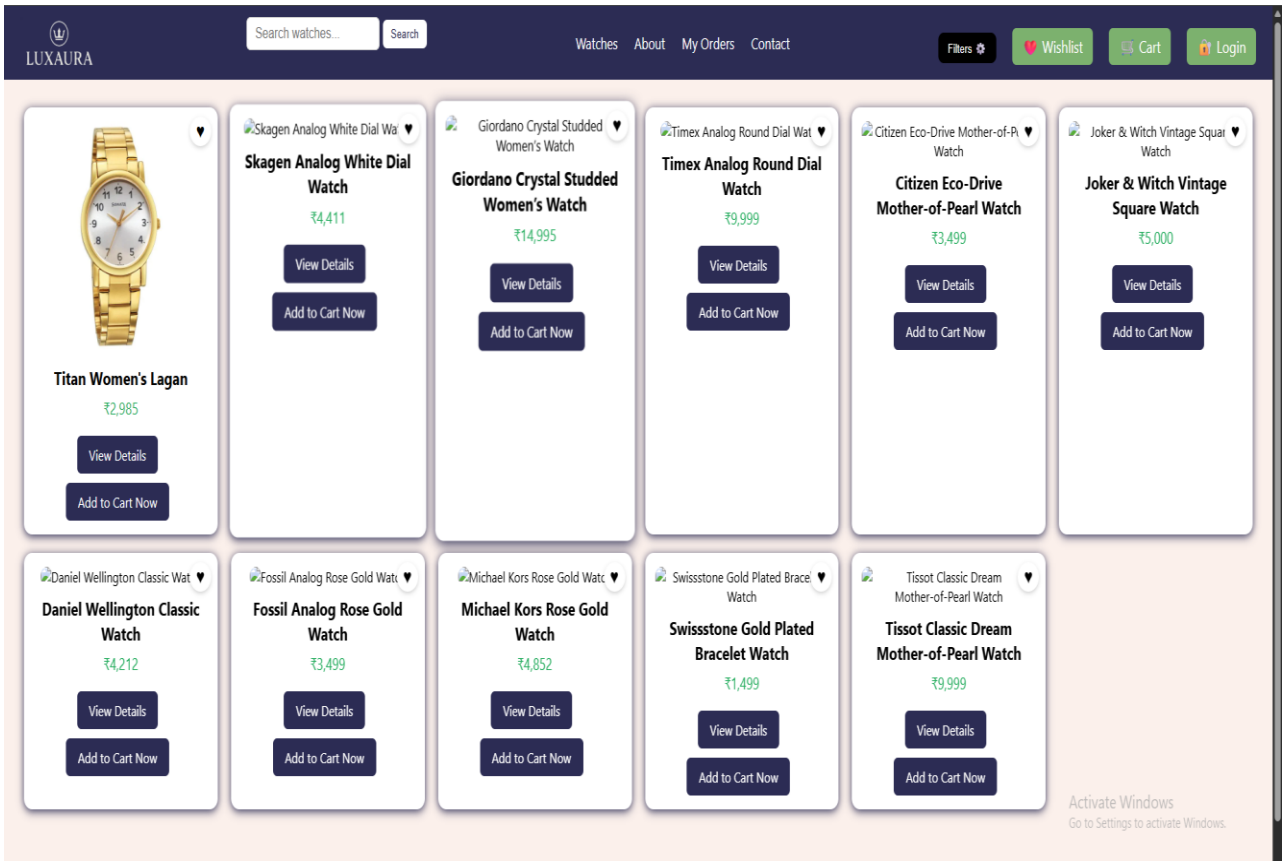
- **Key Attribute:** An attribute that uniquely identifies an entity (e.g., `Order_ID`).
- **Composite Attribute:** An attribute that can be divided into sub-parts (e.g., `Address` can be divided into `City`, `State`, `Zip Code`).
- **Derived Attribute:** An attribute whose value is derived from other attributes (e.g., `Total_Order_Value` is calculated from `Price` × `Quantity`).

ENTITY RELATIONSHIP DIAGRAM



INPUT FORMS, PAGES AND CODING

Home Page:

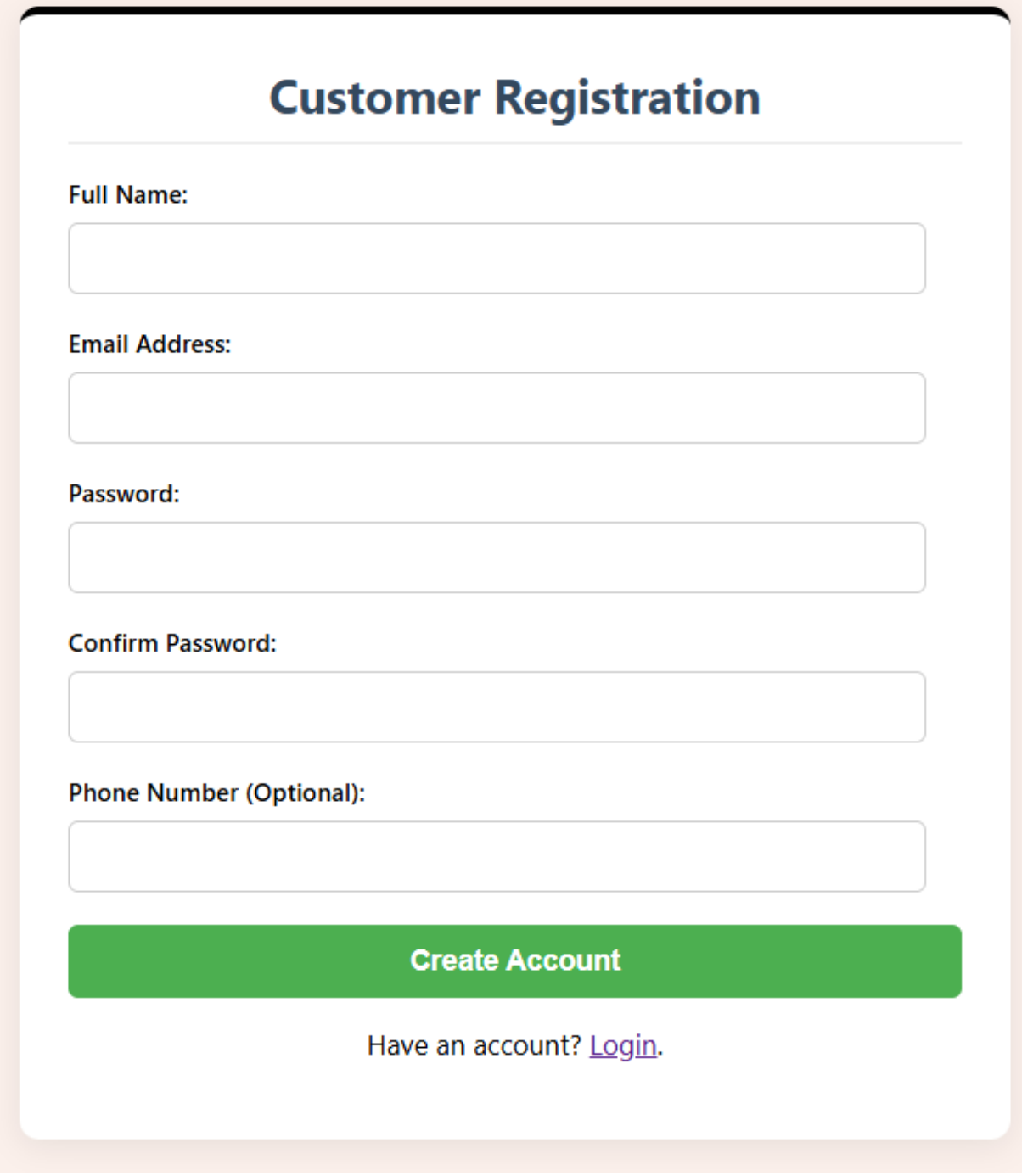


Code:

```
<?php
include "partials/conn.php";
?>

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8" />
    <meta-name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Watch Store</title>
    <link rel="stylesheet" href="style.css">
</head>

<body>
    <?php
    include "nav.php";
```

```
    ?>


    <?php
    $sql = "SELECT * FROM watches";
    $result = $conn->query($sql);
    ?>


    <!-- WATCH GRID -->
    <div class="watch-container">

        <?php while ($row = $result->fetch_assoc()) { ?>

    <div class="watch-card">
        <a href="add_to_wishlist.php?id=<?= $row['id'] ?>">
            <button class="heart-btn">♥</button>
        </a>

        <img src="img/<?= $row['image'] ?>" alt="<?= $row['title'] ?>">

        <div class="watch-title"><?= $row['title'] ?></div>
        <div class="watch-price">₹<?= number_format($row['price']) ?></div>

        <?php if ($row['quantity'] < 1) { ?>
            <!-- OUT OF STOCK -->
            <div class="out-of-stock">Out of Stock</div>
            <a href="watch.php?id=<?= $row['id'] ?>" class="btn">View Details</a>
            <button class="btn disabled" disabled>Add to Cart</button>
        <?php } else { ?>
            <!-- IN STOCK -->
            <a href="watch.php?id=<?= $row['id'] ?>" class="btn">View Details</a>
            <a href="add_to_cart.php?id=<?= $row['id'] ?>" class="btn">Add to Cart Now</a>
        <?php } ?>

    </div>

<?php } ?>


    </div>

    <!-- FOOTER -->
    <footer>
        © 2025 WatchStore. All rights reserved.
    </footer>

</body>

</html>
```

Register Form:



Code:

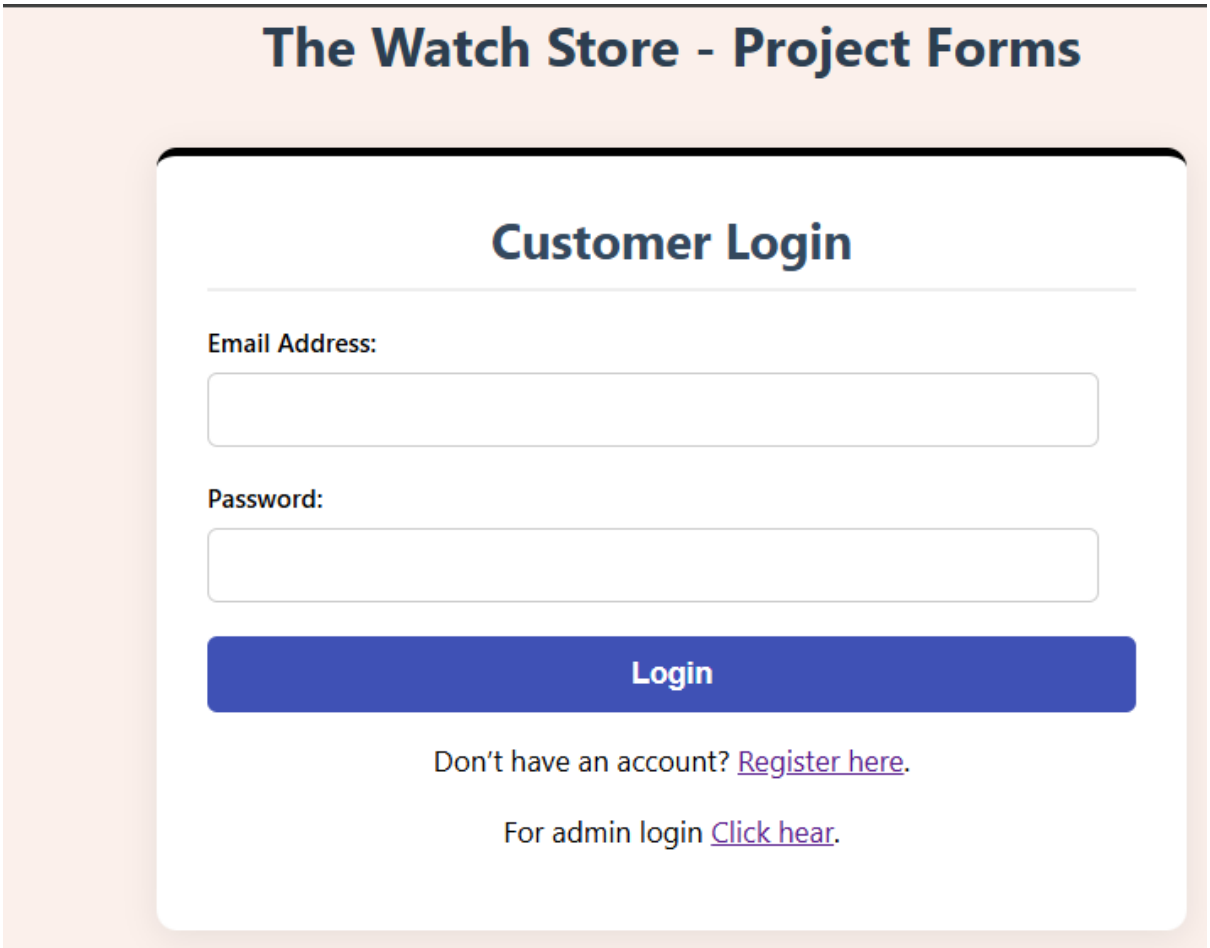
```
<?php
include "partials/conn.php";  // ✅ Separate connection file included

$message = "";
```

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {

   $name     = trim($_POST['full_name']);
   $email    = trim($_POST['email']);
   $password = $_POST['password'];
   $confirm  = $_POST['confirm_password'];
   $phone    = trim($_POST['phone']);

   // 1. Basic Validation
   if (empty($name) || empty($email) || empty($password) || empty($confirm)) {
      $message = "❌ All required fields must be filled!";
   }
   // 2. Validate Email
   elseif (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
      $message = "❌ Invalid email format!";
   }
   // 3. Password Match
   elseif ($password !== $confirm) {
      $message = "❌ Password and Confirm Password do not match!";
   }
   // 4. Password Strength (at least 8 characters, one uppercase, one number)
   elseif (!preg_match('/^(?=.*[A-Z])(?=.*\d).{8,}$/', $password)) {
      $message = "❌ Password must be at least 8 characters, include one uppercase letter and one number!";
   }
   // 5. Optional: Validate Phone
   elseif (!empty($phone) && !preg_match('/^\+?[0-9]{7,15}$/', $phone)) {
      $message = "❌ Invalid phone number!";
   }
   else {
      // 6. Check Email Exists
      $check = $conn->prepare("SELECT id FROM user WHERE email = ?");
      $check->bind_param("s", $email);
      $check->execute();
      $check->store_result();

      if ($check->num_rows > 0) {
         $message = "❌ Email is already registered!";
      } else {
         // 7. Insert
         $hashed_pass = password_hash($password, PASSWORD_DEFAULT);

         $stmt = $conn->prepare("INSERT INTO user (name, email, password, phone) VALUES (?, ?, ?, ?)");
```

```
        $stmt->bind_param("ssss", $name, $email, $hashed_pass, $phone);

        if ($stmt->execute()) {
            $message = "✅ Account created successfully! <a href='login.php'>Login Now</a>";
        } else {
            $message = "❌ Something went wrong!";
        }
    }
    $check->close();
  }
}

?>

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Customer Registration</title>
<link rel="stylesheet" href="style.css">
<script>
function validateForm() {
  const name = document.getElementById('reg_name').value.trim();
  const email = document.getElementById('reg_email').value.trim();
  const password = document.getElementById('reg_password').value;
  const confirm = document.getElementById('reg_confirm_password').value;
  const phone = document.getElementById('reg_phone').value.trim();
  let msg = "";

  if (!name || !email || !password || !confirm) {
    msg = "❌ All required fields must be filled!";
  } else if (!/^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email)) {
    msg = "❌ Invalid email format!";
  } else if (password !== confirm) {
    msg = "❌ Passwords do not match!";
  } else if (!/^(?=.*[A-Z])(?=.*\d).{8,}$/.test(password)) {
    msg = "❌ Password must be at least 8 characters, include one uppercase letter and one number!";
  } else if (phone && !/^\+?[0-9]{7,15}$/.test(phone)) {
    msg = "❌ Invalid phone number!";
  }

  if (msg) {
    alert(msg);
```

```
        return false;
    }
    return true;
}
</script>
</head>
<body>

<div class="form-container">
    <h2>Customer Registration</h2>

    <?php if ($message != "") { ?>
        <p style="text-align:center; background:#eee; padding:10px; border-radius:5px;">
            <?= $message ?>
        </p>
    <?php } ?>

    <form action="" method="post" onsubmit="return validateForm()">
        <label for="reg_name">Full Name:</label>
        <input type="text" id="reg_name" name="full_name" required>

        <label for="reg_email">Email Address:</label>
        <input type="email" id="reg_email" name="email" required>

        <label for="reg_password">Password:</label>
        <input type="password" id="reg_password" name="password" required>

        <label for="reg_confirm_password">Confirm Password:</label>
        <input type="password" id="reg_confirm_password" name="confirm_password" required>

        <label for="reg_phone">Phone Number (Optional):</label>
        <input type="text" id="reg_phone" name="phone">

        <button type="submit" class="button success">Create Account</button>
    </form>

    <p style="text-align:center; margin-top:15px;">
        Have an account? <a href="login.php">Login</a>.
    </p>
</div>

</body>
</html>
```

Login Form:



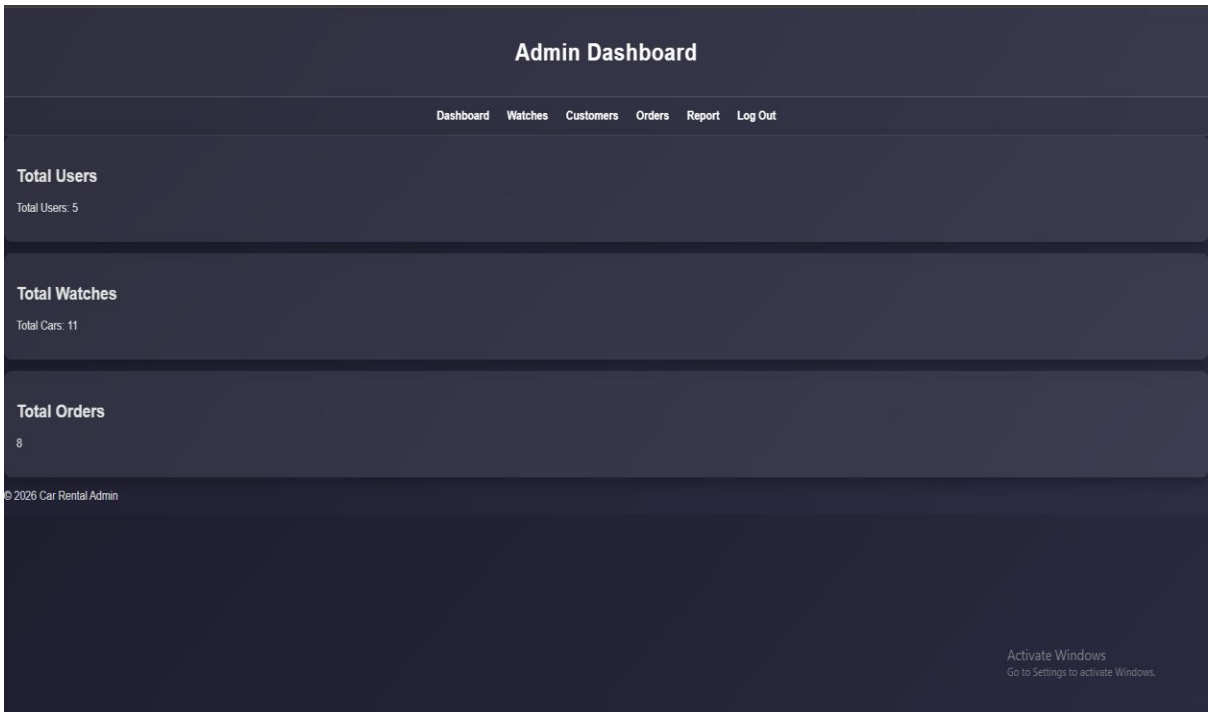
Code:

```
<?php
session_start();
include "partials/conn.php";

$message = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {

    $email    = $_POST['email'];
    $password = $_POST['password'];

    // Check if user exists
    $stmt = $conn->prepare("SELECT id, name, password FROM user WHERE email =
?");
    $stmt->bind_param("s", $email);
    $stmt->execute();
```

```
    $result = $stmt->get_result();

    if ($result->num_rows === 1) {

        $row = $result->fetch_assoc();

        // Verify hashed password
        if (password_verify($password, $row['password'])) {

            // Create session
            $_SESSION['user_id']   = $row['id'];
            $_SESSION['user_name'] = $row['full_name'];

            header("Location: index.php");
            exit();
        } else {
            $message = "✕ Incorrect password!";
        }

    } else {
        $message = "✕ No account found with this email!";
    }
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Customer Login</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>

<h1>The Watch Store - Project Forms</h1>

<div class="form-container">
    <h2>Customer Login</h2>

    <!-- Show message -->
    <?php if ($message != "") { ?>
        <p    style="text-align:center;    background:#f7dcdc;    padding:10px;    border-
radius:5px;">
            <?= $message ?>
        </p>
    <?php } ?>

    <form action="" method="post">
```

```
        <label for="login_email">Email Address:</label>
        <input type="email" id="login_email" name="email" required>

        <label for="login_password">Password:</label>
        <input type="password" id="login_password" name="password" required>

        <button type="submit" class="button">Login</button>
    </form>

    <p style="text-align:center; margin-top:15px;">
        Don’t have an account? <a href="SignUp.php">Register here</a>.
    </p>

    <p style="text-align:center; margin-top:15px;">
        For admin login <a href="admin/adminlogin.php">Click hear</a>.
    </p>
</div>

</body>
</html>
```

Admin Dashboard:

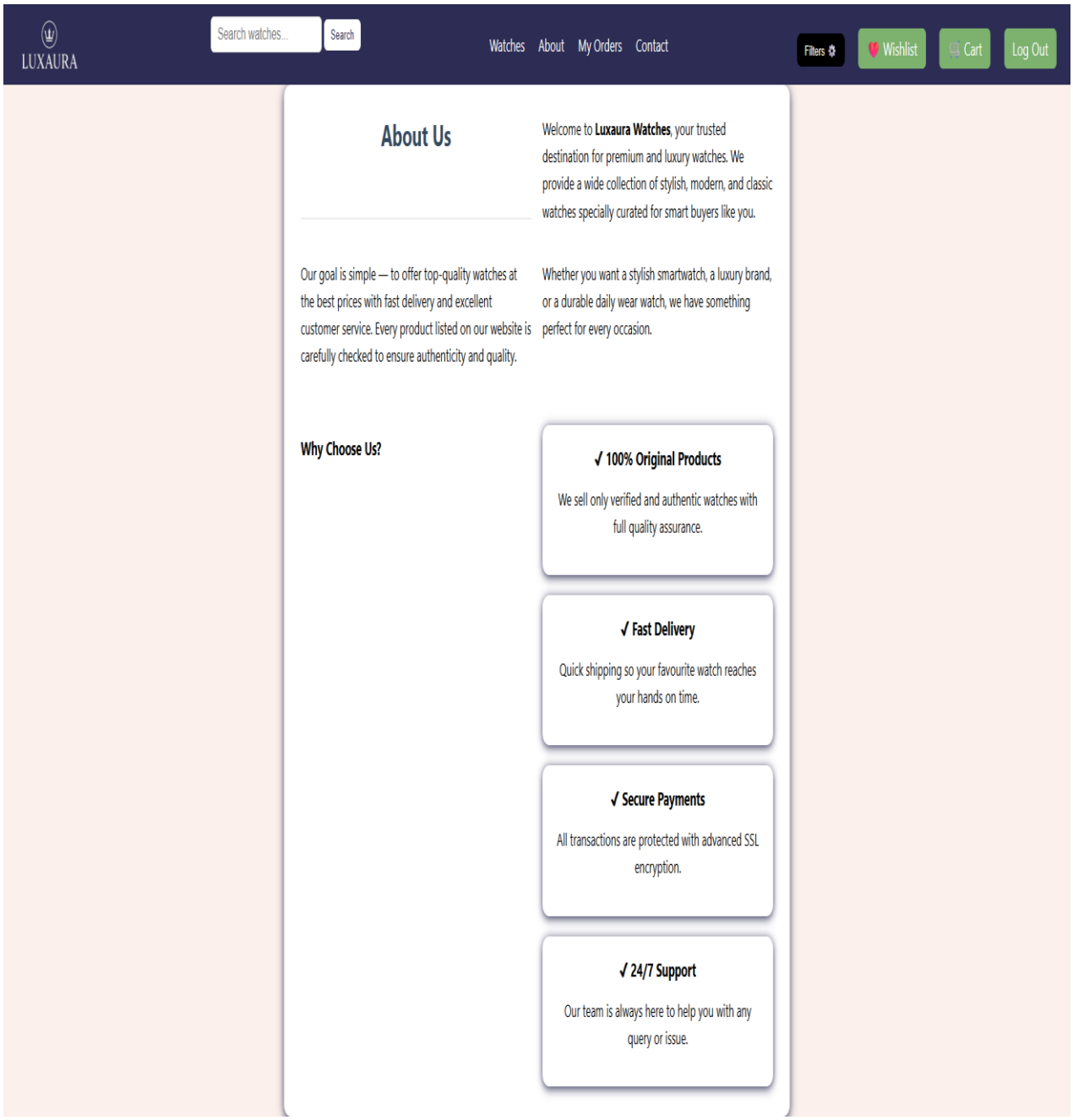


Code:

```
 <!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Admin Dashboard</title>
  <link rel="stylesheet" href="style.css">
</head>

<body>

  <header class="header">
    <h1>Admin Dashboard</h1>
  </header>

  <?php include("partials/nav.php"); ?>
  <?php include("../partials/conn.php"); ?>

  <section>
    <div class="dashboard-card">
      <h2>Total Users</h2>
      <?php
```

```
    $sql = "SELECT * FROM user";
    $result= $conn->query($sql);
    $totalUsers = mysqli_num_rows($result);
    // Replace this with your logic to fetch actual total users
    echo "<p>Total Users: " . $totalUsers . "</p>";
    ?>
   </div>

   <div class="dashboard-card">
    <h2>Total Watches</h2>
    <?php

    $sql = "SELECT * FROM watches";
    $result= $conn->query($sql);
    $totalCars=mysqli_num_rows($result);
    // Replace this with your logic to fetch actual total cars
    echo "<p>Total Cars: " . $totalCars . "</p>";
    ?>
   </div>

   <div class="dashboard-card">
    <h2>Total Orders</h2>
    <?php
    // Example PHP logic to fetch and display rental history content
    // Assuming $rentalHistoryContent contains the retrieved rental history
    $sql = "SELECT * FROM orders";
    $result= $conn->query($sql);
    $rentalHistoryContent=mysqli_num_rows($result);
    echo "<p>" . $rentalHistoryContent . "</p>";
    ?>
   </div>
  </section>

  <footer>
   <p>&copy;
    <?php echo date("Y"); ?> Car Rental Admin
   </p>
  </footer>

 </body>

</html>
```

About Page:



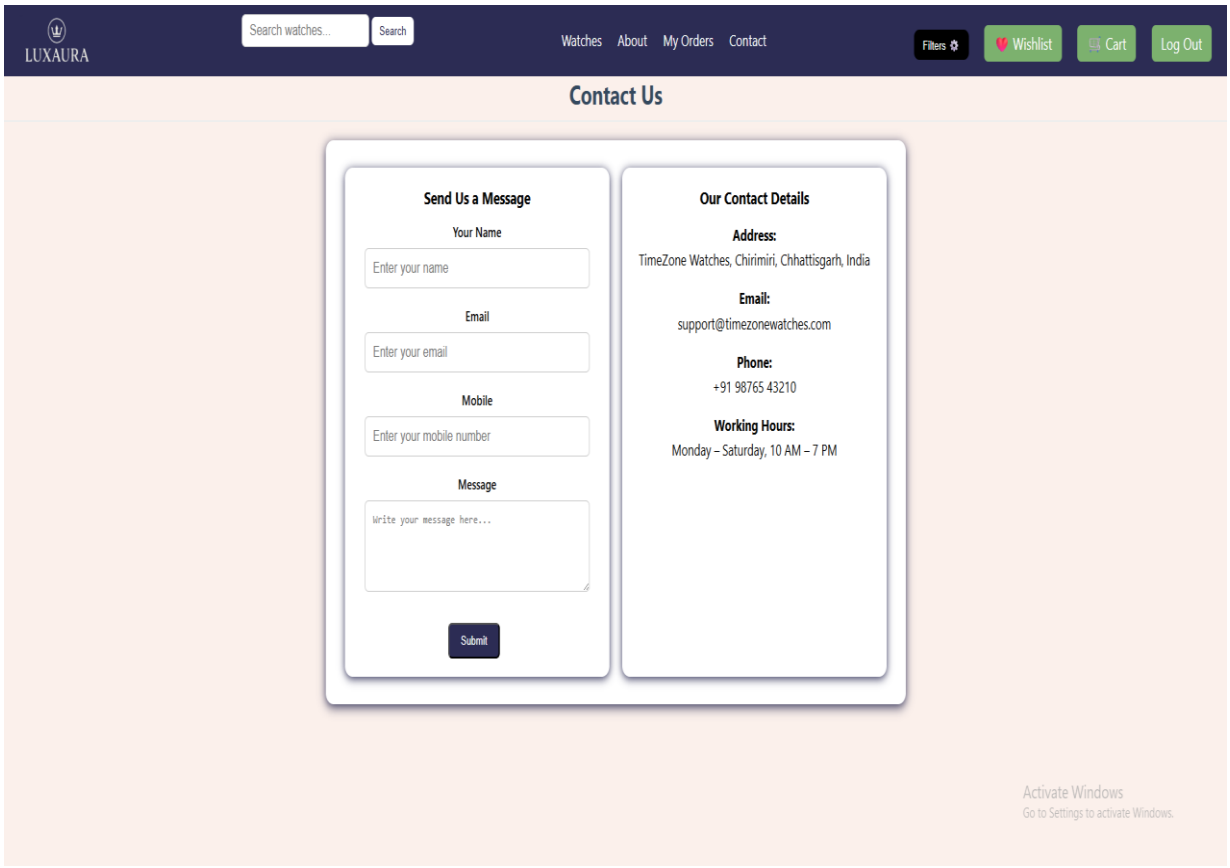
Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>About Us</title>
<link rel="stylesheet" href="style.css">
</head>
<body>

<?php
include "nav.php";
?>

<div class="container">
  <h2>About Us</h2>

  <p>
    Welcome to <strong>Luxaura Watches</strong>, your trusted destination for premium and luxury watches.
    We provide a wide collection of stylish, modern, and classic watches specially curated for smart buyers like you.
  </p>

  <p>
    Our goal is simple — to offer top-quality watches at the best prices with fast delivery and excellent customer service.
    Every product listed on our website is carefully checked to ensure authenticity and quality.
  </p>

  <p>
    Whether you want a stylish smartwatch, a luxury brand, or a durable daily wear watch,
    we have something perfect for every occasion.
  </p>

  <h3 style="margin-top: 30px;">Why Choose Us?</h3>

  <div class="features">
    <div class="card">
      <h3>✓ 100% Original Products</h3>
      <p>We sell only verified and authentic watches with full quality assurance.</p>
    </div>

    <div class="card">
      <h3>✓ Fast Delivery</h3>
      <p>Quick shipping so your favourite watch reaches your hands on time.</p>
    </div>

    <div class="card">
```

```
        <h3>✓ Secure Payments</h3>
        <p>All transactions are protected with advanced SSL encryption.</p>
      </div>

      <div class="card">
        <h3>✓ 24/7 Support</h3>
        <p>Our team is always here to help you with any query or issue.</p>
      </div>
    </div>

</div>

</body>
</html>
```

Contact Page:

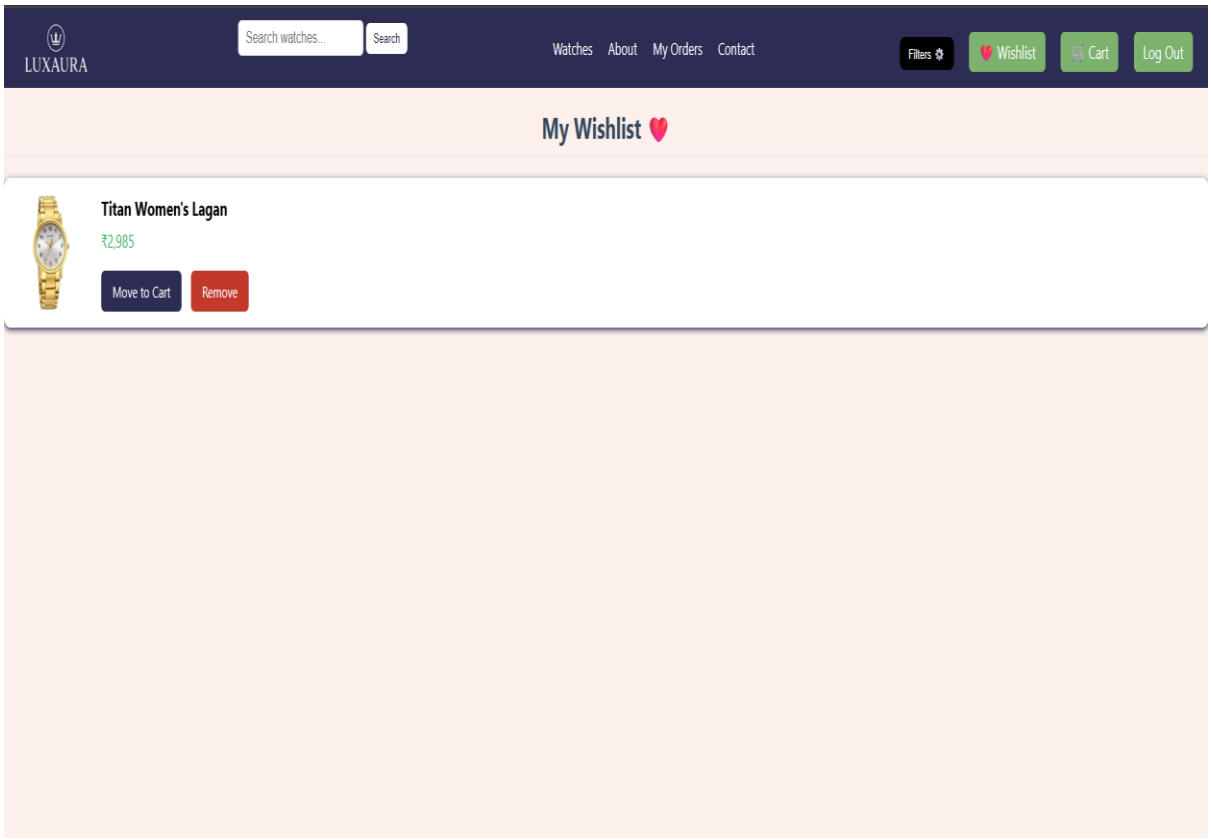


Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Contact Us</title>
<link rel="stylesheet" href="style.css">
</head>
<body>

<?php
include "nav.php";
?>

<h2>Contact Us</h2>

<div class="container">
```

```
    <!-- Contact Form -->
    <div class="card">
      <h3>Send Us a Message</h3>

      <label>Your Name</label>
      <input type="text" placeholder="Enter your name">

      <label>Email</label>
      <input type="email" placeholder="Enter your email">

      <label>Mobile</label>
      <input type="text" placeholder="Enter your mobile number">

      <label>Message</label>
      <textarea placeholder="Write your message here..."></textarea>

      <button class="btn">Submit</button>
    </div>

    <!-- Contact Info -->
    <div class="card info">
      <h3>Our Contact Details</h3>

      <p><strong>Address:</strong><br>
      TimeZone Watches, Chirimiri, Chhattisgarh, India</p>

      <p><strong>Email:</strong><br>
      support@timezonewatches.com</p>

      <p><strong>Phone:</strong><br>
      +91 98765 43210</p>

      <p><strong>Working Hours:</strong><br>
      Monday – Saturday, 10 AM – 7 PM</p>
    </div>

  </div>

  </body>
  </html>
```

Wishlist Page:

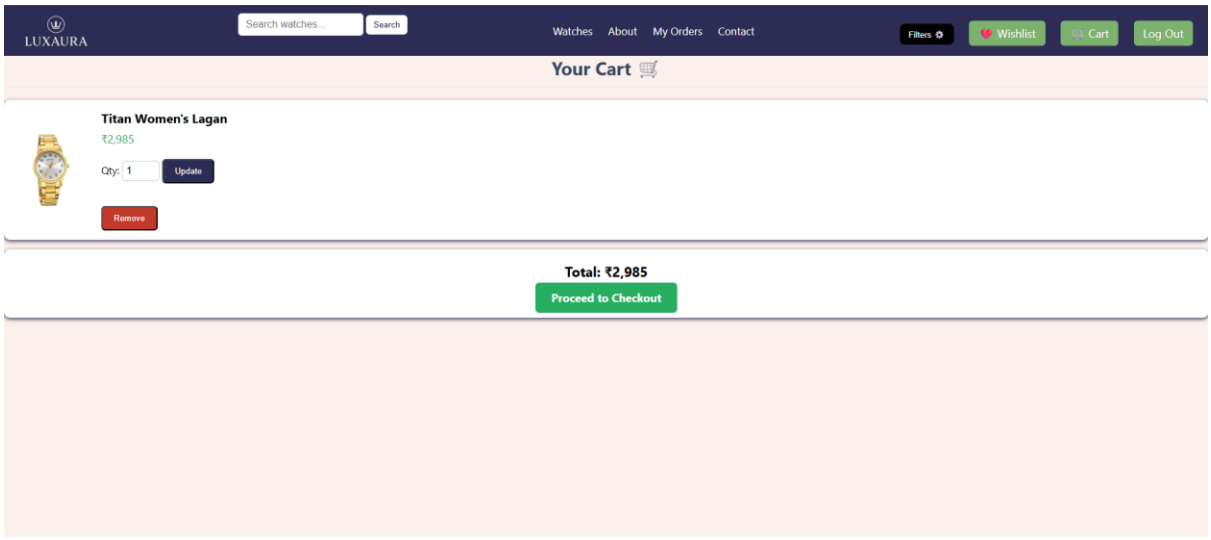


Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Wishlist</title>
   <link rel="stylesheet" href="style.css">
</head>
<body>
<?php include "nav.php";
include "partials/login_verification.php"; ?>
<?php
include "partials/conn.php";
$user_id = $_SESSION['user_id']; // TEMP

// Fetch wishlist items joined with watch details
$sql = "SELECT wishlist.id AS wish_id, watches.title, watches.price, watches.image, watches.id AS product_id
```

```
        FROM wishlist
        JOIN watches ON wishlist.product_id = watches.id
        WHERE wishlist.user_id = $user_id";

$result = $conn->query($sql);
?>

<h2 style="text-align:center; margin-top:20px;">My Wishlist ❤</h2>

<?php
if ($result->num_rows > 0) {

   while ($row = $result->fetch_assoc()) {
      echo "
      <div class='wishlist-item'>
         <img src='img/Watch.webp'>
         <div class='info'>
            <div class='name'>{$row['title']}</div>
            <div class='price'>₹" . number_format($row['price']) . "</div>

            <a                                                    class='btn'
href='add_to_cart.php?id={$row['product_id']}'>Move to Cart</a>
            <a                          class='btn                        remove'
href='remove_wish.php?id={$row['wish_id']}'>Remove</a>
         </div>
      </div>
      ";
   }

} else {
   echo "<p style='text-align:center; margin-top:20px;'>Your wishlist is
empty ☹</p>";
}
?>

</body>
</html>
```

Cart Page:

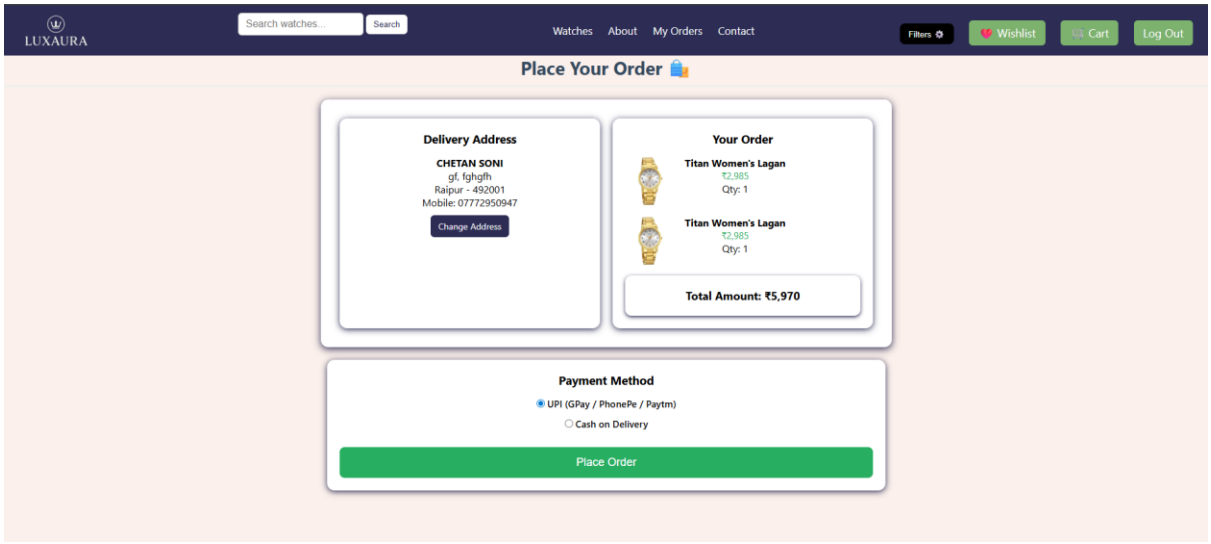


Code:

```
<?php
include "nav.php";
include "partials/login_verification.php";

include "partials/conn.php";
$user_id = $_SESSION['user_id'];

$sql = "SELECT cart.*, watches.title, watches.image, watches.price
        FROM cart
        JOIN watches ON cart.product_id = watches.id
        WHERE cart.user_id = '$user_id'";

$result = $conn->query($sql);

$total = 0;
?>
<!DOCTYPE html>
<html>
<head>
    <title>Your Cart</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>

<h2>Your Cart 🛒</h2>
```

```
<?php
if ($result->num_rows > 0) {

   while ($row = $result->fetch_assoc()) {
      $total += $row['price'] * $row['quantity'];
?>
      <div class="cart-item">
         <img src="img/Watch.webp">
         <div class="info">
            <div class="name"><?= $row['title'] ?></div>
            <div class="price">₹<?= number_format($row['price']) ?></div>
            <div class="qty-box">
               Qty:
               <form action="update_cart.php" method="post" style="display:inline-block;">
                  <input type="hidden" name="cart_id" value="<?= $row['cart_id'] ?>">
                  <input type="number" name="quantity" value="<?= $row['quantity'] ?>" min="1">
                  <button class="btn">Update</button>
               </form>
            </div>
            <form action="remove_cart.php" method="post">
               <input type="hidden" name="cart_id" value="<?= $row['cart_id'] ?>">
               <button class="btn remove">Remove</button>
            </form>
         </div>
      </div>
<?php
   }
} else {
   echo "<h3>Your cart is empty ☹</h3>";
}
?>

<?php if ($total > 0) { ?>
<div class="total-box">
   <div class="total">Total: ₹<?= number_format($total) ?></div>
   <a href="order.php" class="checkout-btn">Proceed to Checkout</a>
</div>
<?php } ?>

</body>
</html>
```

Order Page:



Code:

```
<?php

include "nav.php";
include "partials/conn.php";

/* ========================
  LOGIN CHECK
======================== */
$user_id = $_SESSION['user_id'] ?? 0;

if (!$user_id) {
   header("Location: login.php");
   exit;
}

$sql = "SELECT
           cart.product_id,
           cart.quantity AS cart_qty,
           watches.title,
           watches.image,
           watches.price,
           watches.quantity AS stock
        FROM cart
        INNER JOIN watches ON watches.id = cart.product_id
        WHERE cart.user_id = ?";

$stmt = $conn->prepare($sql);
```

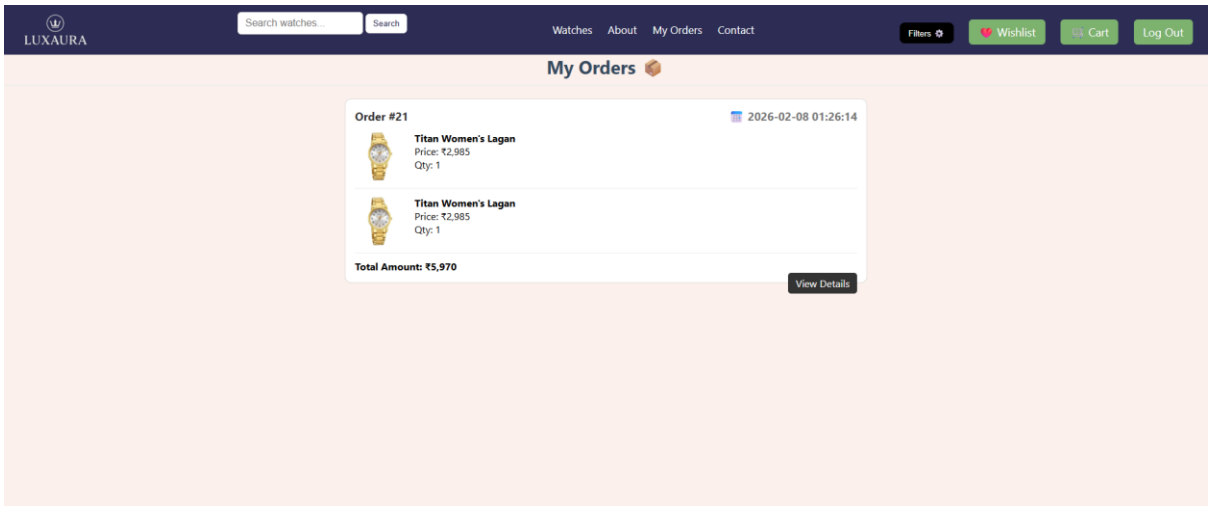
```
$stmt->bind_param("i", $user_id);
$stmt->execute();
$result = $stmt->get_result();

/* ========================
  CALCULATE TOTAL & STOCK
======================== */
$total = 0;
$outOfStock = false;
$cartItems = [];

if ($result->num_rows > 0) {

   while ($item = $result->fetch_assoc()) {

      $stock   = (int)$item['stock'];
      $cartQty = (int)$item['cart_qty'];

      if ($stock <= 0 || $stock < $cartQty) {
         $outOfStock = true;
      }

      $item['itemTotal'] = $item['price'] * $cartQty;
      $total += $item['itemTotal'];
      $cartItems[] = $item;
   }

} else {
   $outOfStock = true;
}

/* ========================
  FETCH DEFAULT ADDRESS
======================== */
$addrSql = "SELECT * FROM address
         WHERE user_id = ? AND is_default = 1
         LIMIT 1";

$addrStmt = $conn->prepare($addrSql);
$addrStmt->bind_param("i", $user_id);
$addrStmt->execute();
$address = $addrStmt->get_result()->fetch_assoc();
?>

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
```

```
<title>Place Order</title>
<link rel="stylesheet" href="style.css">
</head>

<body>

<h2>Place Your Order 🛍️</h2>

<div class="container">

  <!-- DELIVERY ADDRESS -->
  <div class="card">
    <h3>Delivery Address</h3>

    <?php if ($address) { ?>
      <div class="address-box">
        <strong><?= htmlspecialchars($address['full_name']) ?></strong><br>
        <?= htmlspecialchars($address['house_no']) ?>,
        <?= htmlspecialchars($address['area']) ?><br>
        <?= htmlspecialchars($address['city']) ?> -
        <?= htmlspecialchars($address['pincode']) ?><br>
        Mobile: <?= htmlspecialchars($address['mobile']) ?>
      </div>
      <a href="address.php" class="change-btn">Change Address</a>
    <?php } else { ?>
      <p>No address added!</p>
      <a href="address.php" class="change-btn">Add Address</a>
    <?php } ?>
  </div>

  <div class="card">
    <h3>Your Order</h3>

    <?php if (!empty($cartItems)) { ?>
      <?php foreach ($cartItems as $item) { ?>

        <div class="item">
          <img src="img/Watch.webp" alt="Product">

          <div>
            <div class="item-name"><?= htmlspecialchars($item['title']) ?></div>
            <div class="item-price">₹<?= number_format($item['price']) ?></div>
            <div>Qty: <?= $item['cart_qty'] ?></div>

            <?php if ((int)$item['stock'] < (int)$item['cart_qty']) { ?>
              <div style="color:red;font-weight:bold;">
                Out of Stock
```

```
                    </div>
                <?php } ?>
            </div>
        </div>

        <?php } ?>

        <div class="total-box">
            Total Amount: ₹<?= number_format($total) ?>
        </div>

    <?php } else { ?>
        <p>Your cart is empty!</p>
    <?php } ?>
</div>

</div>

<!-- PAYMENT SECTION -->
<div class="card" style="max-width:850px;margin:20px auto;">
    <h3>Payment Method</h3>

    <form action="place_order.php" method="post">

        <label class="payment-option">
            <input type="radio" name="payment_method" value="upi"
checked>
            UPI (GPay / PhonePe / Paytm)
        </label>

        <label class="payment-option">
            <input type="radio" name="payment_method" value="cod">
            Cash on Delivery
        </label>
        <?php if (!$outOfStock && $address) { ?>
            <button type="submit" class="place-btn">Place Order</button>
        <?php } else { ?>
            <button type="button" class="place-btn" disabled
                style="background:#ccc;cursor:not-allowed;">
                Cannot Place Order
            </button>
            <p style="color:red;margin-top:10px;">
                Please fix out-of-stock items or add address.
            </p>
        <?php } ?>
    </form>
</div>

</body>
</html>
```

My Orders Page:

s



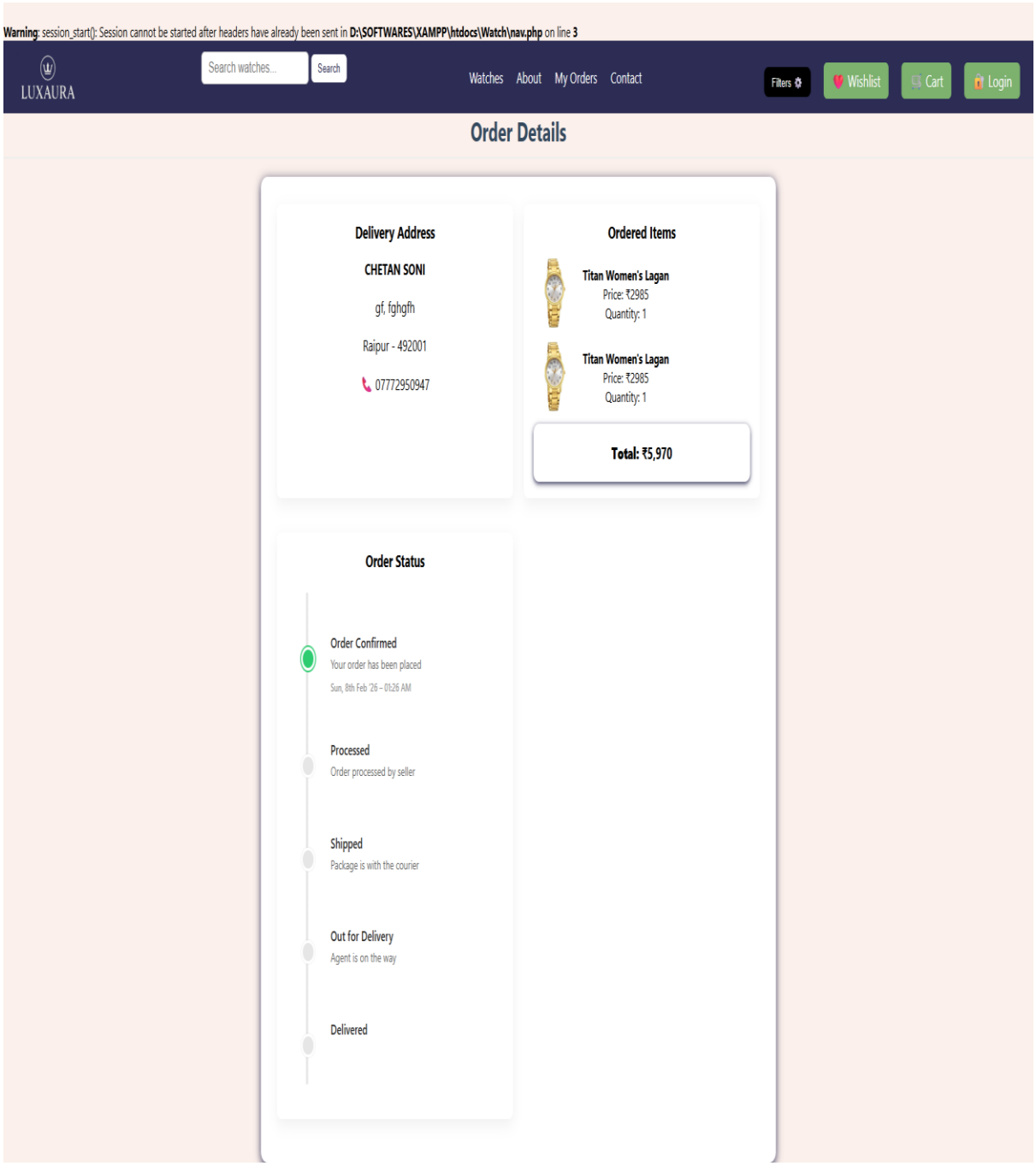
Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>My Orders</title>
<link rel="stylesheet" href="style.css">

<style>
.order-box {
    background: #fff;
    padding: 15px;
    margin: 15px auto;
    border-radius: 10px;
    max-width: 800px;
    border: 1px solid #ddd;
}

.order-header {
    font-size: 18px;
    color: #333;
    font-weight: bold;
}

.order-item {
    display: flex;
    gap: 15px;
    margin-top: 12px;
```

```
    border-bottom: 1px solid #eee;
    padding-bottom: 10px;
}

.order-item img {
    width: 80px;
    border-radius: 6px;
}

.order-total {
    margin-top: 10px;
    font-weight: bold;
    color: #000;
}

.btn-details {
    background: #333;
    color: #fff;
    padding: 6px 12px;
    border-radius: 5px;
    text-decoration: none;
    float: right;
}
</style>

</head>
<body>

<?php include "nav.php";
include "partials/login_verification.php"; ?>
<?php
include "partials/conn.php";

// Dummy user_id = 1
$user_id = $_SESSION['user_id'];

// Fetch all orders of the user
$ordersSQL = "SELECT * FROM orders WHERE user_id = '$user_id'
ORDER BY id DESC";
$ordersResult = $conn->query($ordersSQL);
?>

<h2 style="text-align:center;">My Orders 🎁</h2>

<?php
if ($ordersResult->num_rows > 0) {

    while ($order = $ordersResult->fetch_assoc()) {
        $order_id = $order['id'];
```

```
        // Fetch order items
        $itemsSQL = "
            SELECT order_items.*, watches.title, watches.image
            FROM order_items
            JOIN watches ON order_items.product_id = watches.id
            WHERE order_items.order_id = '$order_id'
        ";
        $itemsResult = $conn->query($itemsSQL);
?>
        <div class="order-box">
            <div class="order-header">
                Order #<?= $order_id ?>
                <span style="float:right; color:#777;">📅 <?= $order['created_at'] ?></span>
            </div>

            <?php while ($item = $itemsResult->fetch_assoc()) { ?>
                <div class="order-item">
                    <img src="img/Watch.webp">
                    <div>
                        <div><strong><?= $item['title'] ?></strong></div>
                        <div>Price: ₹<?= number_format($item['price']) ?></div>
                        <div>Qty: <?= $item['quantity'] ?></div>
                    </div>
                </div>
            <?php } ?>

            <div class="order-total">
                Total Amount: ₹<?= number_format($order['total_amount']) ?>
            </div>

            <a href="order_details.php?id=<?= $order_id ?>" class="btn-details">View Details</a>
        </div>

<?php
    }
} else {
    echo "<p style='text-align:center; font-size:18px;'>No orders found!</p>";
}
?>

</body>
</html>
```

Order Details Page:



Code:

```
<?php
include "partials/conn.php";

if (!isset($_GET['id'])) {
    die("Invalid Order ID");
```

```
}
$order_id = $conn->real_escape_string($_GET['id']); // small safety improvement

// 1. Fetch Order Details
$orderSQL = "SELECT orders.*, address.full_name, address.house_no, address.area, address.city, address.pincode, address.mobile
        FROM orders
        JOIN address ON orders.address_id = address.id
        WHERE orders.id = '$order_id'";

$orderResult = $conn->query($orderSQL);
if ($orderResult->num_rows == 0) {
   die("Order not found");
}
$order = $orderResult->fetch_assoc();

// 2. Fetch Order Items
$itemsSQL = "
   SELECT order_items.*, watches.title, watches.image
   FROM order_items
   JOIN watches ON order_items.product_id = watches.id
   WHERE order_items.order_id = '$order_id'
";

$itemsResult = $conn->query($itemsSQL);

if (!$itemsResult) {
   die("Error fetching order items: " . $conn->error);
}


// 3. Define Logic for Timeline Progress
$stages = [
   'confirmed'       => 1,
   'processed'       => 2,
   'shipped'         => 3,
   'out_for_delivery' => 4,
   'delivered'       => 5
];

// Get current status from DB
$current_status = $order['status'] ?? 'confirmed';

// Convert status to a number (e.g., 'shipped' becomes 3)
$current_step_number = $stages[$current_status] ?? 1;

// progress percent for gradient fill (0..100)
$total_steps = count($stages);
if ($total_steps <= 1) {
```

```
    $progress_percent = 100;
} else {
    // This treats the first step as 0% and last step as 100%.
    $progress_percent = (($current_step_number - 1) / ($total_steps - 1))
* 100;
}
$progress_percent = round($progress_percent, 2);
?>
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Order Details</title>
    <link rel="stylesheet" href="style.css">
    <style>
        /* ====== Reliable two-column timeline (no overlap) ====== */

        /* container for timeline rows */
        .timeline {
            position: relative;
            margin: 12px 0;
            --left-col-width: 72px;      /* total width reserved for rail/dot
column */
            --rail-x: calc(var(--left-col-width) / 2); /* center x for the rail */
            --dot-size: 20px;
        }

        /* vertical rail: single element spanning full timeline height
           gradient uses --progress custom property injected inline from
PHP */
        .timeline::before {
            content: "";
            position: absolute;
            left: var(--rail-x);
            top: 8px;
            bottom: 8px;
            width: 4px;
            border-radius: 4px;
            z-index: 1;
            background: linear-gradient(
                to bottom,
                #2ecc71 0%,
                #2ecc71 var(--progress),
                #e6e6e6 var(--progress),
                #e6e6e6 100%
            );
        }

        /* each row is a 2-column layout: left fixed, right flexible */
        .timeline-item {
```

```
            display: flex;
            align-items: flex-start;
            gap: 16px;
            padding: 18px 12px;
            min-height: 56px; /* ensure some vertical room */
            position: relative;
        }

        /* left column: holds the dot and keeps width fixed */
        .timeline-item .left {
            width: var(--left-col-width);
            display: flex;
            align-items: center;       /* vertically center dot relative to content
*/
            justify-content: center;   /* horizontally center inside left area */
            position: relative;
            z-index: 2; /* place above rail gradient */
        }

        /* the dot itself */
        .timeline-item .dot {
            width: var(--dot-size);
            height: var(--dot-size);
            border-radius: 50%;
            background: #e6e6e6;
            border: 3px solid #fff;      /* inner white ring */
            box-shadow: 0 0 0 1px rgba(0,0,0,0.04);
            display: inline-block;
            flex: 0 0 auto;
            margin-top:30px;
            margin-left: -5px;
        }

        /* active dot */
        .timeline-item .dot.active {
            background: #2ecc71;
            box-shadow: 0 0 0 2px rgba(46,204,113,0.85);

        }

        /* right column: content block */
        .timeline-item .content {
            flex: 1 1 auto;
            /* ensure text is left-aligned inside timeline, even if global CSS
centers headers */
            text-align: left;
            padding-right: 6px;
              margin-left: 50px;

        }
```

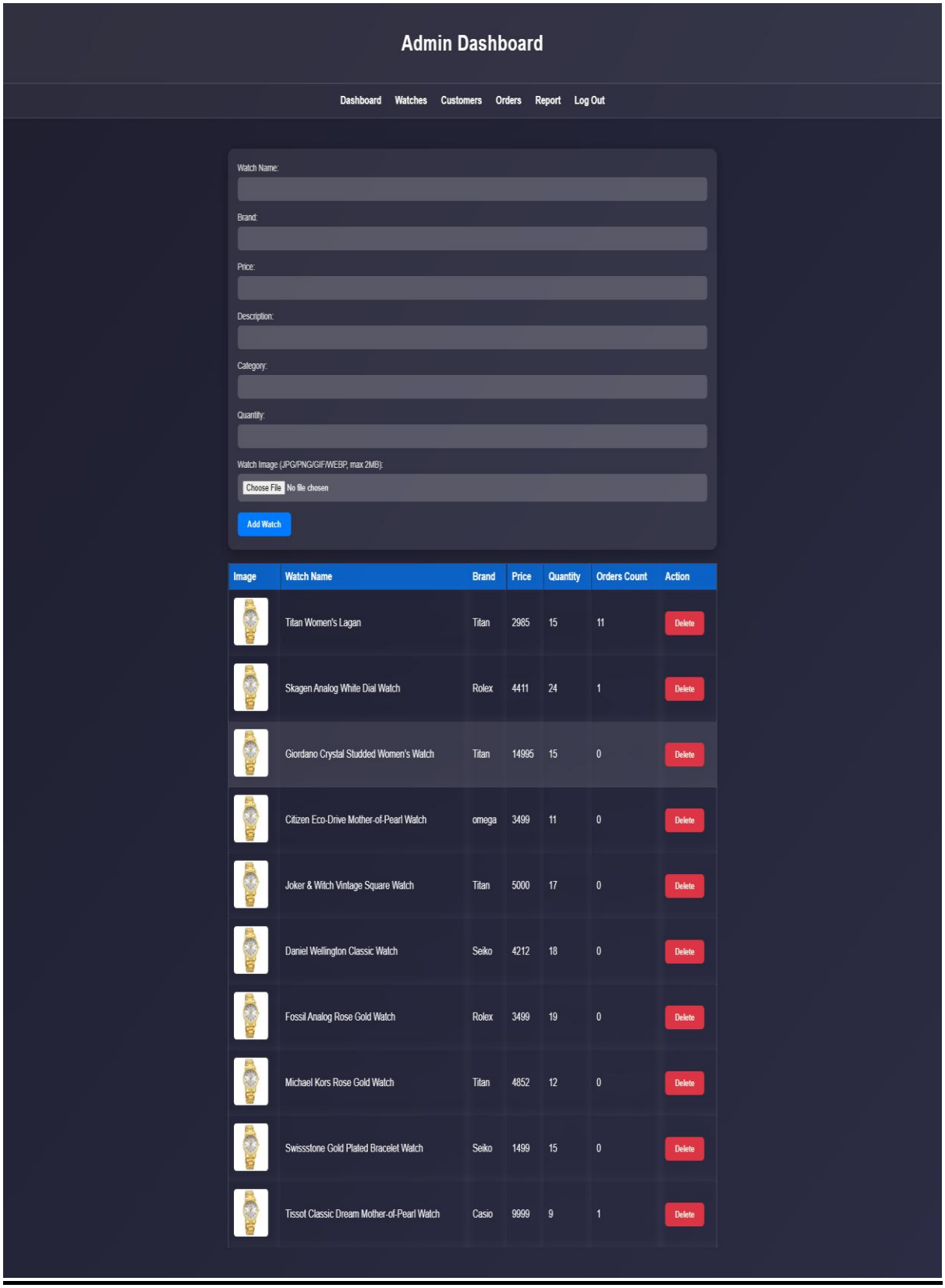
```
      /* typography inside timeline (keeps text tidy) */
      .timeline-item h4 {
         margin: 0 0 6px 0;
         font-size: 16px;
         font-weight: 600;
         color: #222;
         line-height: 1.1;
      }

      .timeline-item p {

         margin: 0 0 8px 0;
         color: #666;
         font-size: 14px;
         line-height: 1.35;

      }

      .timeline-item .date {
         font-size: 12px;
         color: #999;
         font-weight: 500;
      }

      /* small screens adjust left column width & dot size */
      @media (max-width: 600px) {
         .timeline {
            --left-col-width: 64px;
            --dot-size: 18px;
         }
      }

      /* visually separate the card container so it looks like your other cards */
      .card { padding: 18px; margin-bottom: 18px; border-radius: 8px; background: #fff; box-shadow: 0 6px 18px rgba(0,0,0,0.06); }

      /* small helper so ordered items don't break layout */
      .item { display:flex; gap:12px; margin-bottom:12px; align-items:center; }

      /* ensure timeline headings inside your card don't inherit weird global styles */
      .card h3 { margin-top: 0; }
   </style>
</head>
<body>

   <?php include "nav.php"; ?>
```

```
<h2 style="text-align:center;">Order Details</h2>

<div class="container" style="max-width:900px; margin:auto;">

    <div class="card">
        <h3>Delivery Address</h3>
        <p><strong><?= htmlspecialchars($order['full_name']) ?></strong></p>
        <p><?= htmlspecialchars($order['house_no']) ?>, <?= htmlspecialchars($order['area']) ?></p>
        <p><?= htmlspecialchars($order['city']) ?> - <?= htmlspecialchars($order['pincode']) ?></p>
        <p>📞 <?= htmlspecialchars($order['mobile']) ?></p>
    </div>

    <div class="card">
        <h3>Ordered Items</h3>

        <?php while ($item = $itemsResult->fetch_assoc()) { ?>
        <div class="item">
            <img src="img/<?= htmlspecialchars($item['image']) ?>" width="100" alt="">
            <div>
                <div><strong><?= htmlspecialchars($item['title']) ?></strong></div>
                <div>Price: ₹<?= htmlspecialchars($item['price']) ?></div>
                <div>Quantity: <?= htmlspecialchars($item['quantity']) ?></div>
            </div>
        </div>
        <?php } ?>

        <div class="total-box">
            <strong>Total:</strong> ₹<?= number_format($order['total_amount']) ?>
        </div>
    </div>

    <div class="card">
        <h3>Order Status</h3>

        <!-- timeline: --progress injected inline from PHP -->
        <div class="timeline" style="--progress: <?= $progress_percent ?>%;">
            <!-- Step 1 -->
            <div class="timeline-item">
                <div class="left">
                    <span class="dot <?= ($current_step_number >= 1) ? 'active' : '' ?>"></span>
```

```
                </div>
                <div class="content">
                    <h4>Order Confirmed</h4>
                    <p>Your order has been placed</p>
                    <div class="date"><?= (new DateTime($order['created_at']))->format('D, jS M ’y – h:i A') ?></div>
                </div>
            </div>

            <!-- Step 2 -->
            <div class="timeline-item">
                <div class="left">
                    <span class="dot <?= ($current_step_number >= 2) ? 'active' : '' ?>"></span>
                </div>
                <div class="content">
                    <h4>Processed</h4>
                    <p>Order processed by seller</p>
                    <?php if (!empty($order['processed_at'])): ?>
                    <div class="date"><?= (new DateTime($order['processed_at']))->format('D, jS M ’y – h:i A') ?></div>
                    <?php endif; ?>
                </div>
            </div>

            <!-- Step 3 -->
            <div class="timeline-item">
                <div class="left">
                    <span class="dot <?= ($current_step_number >= 3) ? 'active' : '' ?>"></span>
                </div>
                <div class="content">
                    <h4>Shipped</h4>
                    <p>Package is with the courier</p>
                    <?php if (!empty($order['shipped_at'])): ?>
                    <div class="date"><?= (new DateTime($order['shipped_at']))->format('D, jS M ’y – h:i A') ?></div>
                    <?php endif; ?>
                </div>
            </div>

            <!-- Step 4 -->
            <div class="timeline-item">
                <div class="left">
                    <span class="dot <?= ($current_step_number >= 4) ? 'active' : '' ?>"></span>
                </div>
                <div class="content">
                    <h4>Out for Delivery</h4>
```

```
                    <p>Agent is on the way</p>
                    <?php if (!empty($order['out_for_delivery_at'])): ?>
                    <div class="date"><?= (new DateTime($order['out_for_delivery_at']))->format('D, jS M ’y – h:i A') ?></div>
                    <?php endif; ?>
                  </div>
                </div>

                <!-- Step 5 -->
                <div class="timeline-item">
                  <div class="left">
                    <span class="dot <?= ($current_step_number >= 5) ? 'active' : '' ?>"></span>
                  </div>
                  <div class="content">
                    <h4>Delivered</h4>
                    <?php if ($current_status == 'delivered'): ?>
                    <p>Enjoy your watch!</p>
                    <?php endif; ?>
                    <?php if (!empty($order['delivered_at'])): ?>
                    <div class="date"><?= (new DateTime($order['delivered_at']))->format('D, jS M ’y – h:i A') ?></div>
                    <?php endif; ?>
                  </div>
                </div>
              </div> <!-- /.timeline -->

            </div> <!-- /.card -->

          </div> <!-- /.container -->

        </body>
        </html>
```

Admin Watches Page:



Code:

```
 <?php
include("../partials/conn.php");

// NOTE: Make sure your `Watches` table has an `image` column (VARCHAR).
// If it doesn't, run something like:
// ALTER TABLE Watches ADD COLUMN image VARCHAR(255) AFTER description;

$upload_dir = __DIR__ . '/../img/';   // Server path



if ($_SERVER["REQUEST_METHOD"] == "POST") {
   // Retrieve form data (use trim to avoid leading/trailing spaces)
   $Title = trim($_POST['title']);
   $Brand = trim($_POST['Brand']);
   $Price = trim($_POST['Price']);
   $Discription = trim($_POST['Discription']);
   $C = trim($_POST['C']);
   $Quantity = intval($_POST['Quantity']);
   $orders_count = intval($_POST['orders_count']);



   // Image upload handling (optional)
   $image_path_for_db = NULL;
   if (isset($_FILES['image']) && $_FILES['image']['error'] !== UPLOAD_ERR_NO_FILE) {
      $file = $_FILES['image'];

      // Basic validation
      $allowed_types = ['image/jpeg', 'image/png', 'image/gif', 'image/webp'];
      $max_size = 100 * 1024 * 1024; // 2 MB

      if ($file['error'] !== UPLOAD_ERR_OK) {
         echo "<script>alert('Image upload error. Please try again.')</script>";
      } elseif (!in_array(mime_content_type($file['tmp_name']), $allowed_types)) {
         echo "<script>alert('Invalid image type. Only JPG, PNG, GIF, WEBP allowed.')</script>";
      } elseif ($file['size'] > $max_size) {
         echo "<script>alert('Image too large. Max 100 MB allowed.')</script>";
      } else {
         // Generate unique filename
```

```
            $ext = pathinfo($file['name'], PATHINFO_EXTENSION);
            $safeName = preg_replace('/[^a-zA-Z0-9-_\.]/', '_',
pathinfo($file['name'], PATHINFO_FILENAME));
            $newFilename = $safeName . '_' . time() . '.' . $ext;
            $destination = $upload_dir . $newFilename;

            if (move_uploaded_file($file['tmp_name'], $destination)) {
               $image_path_for_db = $newFilename;

            } else {
               echo "<script>alert('Failed to move uploaded
image.')</script>";
            }
         }
      }

      // Insert into database using prepared statement to avoid SQL
injection
      // Note: ensure the Watches table has columns: title, Brand, Price,
description, image
      $stmt = $conn->prepare(
         "INSERT INTO `Watches`
    (`title`, `Brand`, `Price`, `quantity`, `description`, `image`, `category`)
    VALUES (?, ?, ?, ?, ?, ?, ?)"
      );

      if ($stmt) {
         // If no image uploaded, pass NULL
         $imgParam = $image_path_for_db !== NULL ?
$image_path_for_db : NULL;
         // Bind parameters (s = string)
         $stmt->bind_param(
            "sssisss",
            $Title,
            $Brand,
            $Price,
            $Quantity,
            $Discription,
            $imgParam,
            $C
         );

         if ($stmt->execute()) {
            echo "<script>alert('Watch added successfully!');</script>";
            // Redirect after a tiny delay to allow alert to show
            echo "<script>setTimeout(function(){ window.location =
'Watches.php'; }, 200);</script>";
            exit;
         } else {
            echo "Error executing query: " . htmlspecialchars($stmt->error);
```

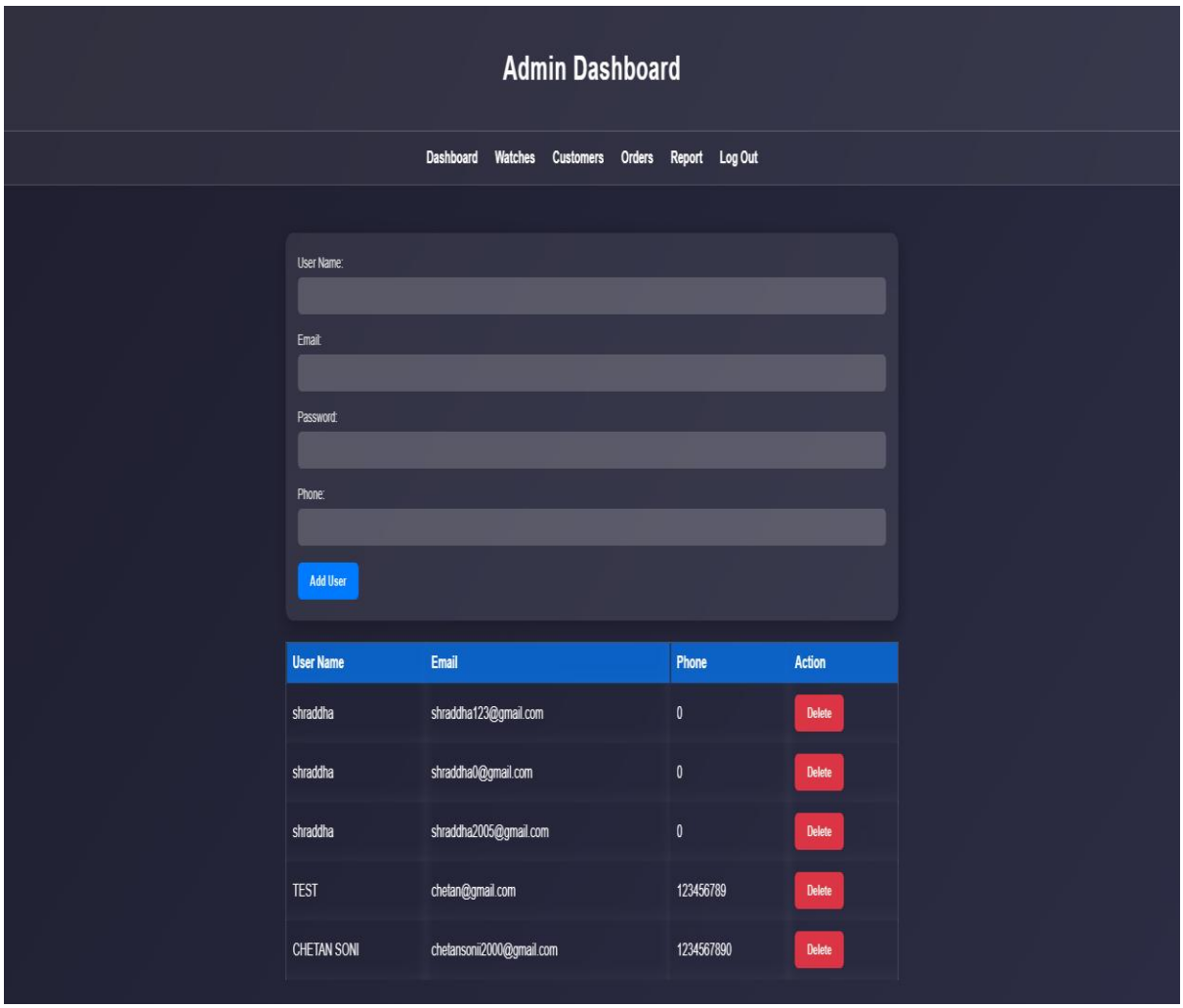
```
        }
        $stmt->close();
    } else {
        echo "Prepare failed: " . htmlspecialchars($conn->error);
    }
}

// SQL query to retrieve data from the Watches table
$sql = "SELECT * FROM Watches";
$result = $conn->query($sql);

// Handle delete (GET param id)
if (isset($_GET['id'])) {
    $id = intval($_GET['id']);

    // Before deleting, try to remove image file if exists
    $sel = $conn->prepare("SELECT image FROM Watches WHERE id
= ?");
    if ($sel) {
        $sel->bind_param("i", $id);
        $sel->execute();
        $sel->bind_result($imageToDelete);
        if ($sel->fetch()) {
            if ($imageToDelete) {
                $fullPath = __DIR__ . '/../img/' . $imageToDelete;
                if (file_exists($fullPath)) {
                    @unlink($fullPath); // delete file (suppress warning)
                }
            }
        }
        $sel->close();
    }

    $del = $conn->prepare("DELETE FROM Watches WHERE id = ?");
    if ($del) {
        $del->bind_param("i", $id);
        if ($del->execute()) {
            echo "<script>alert('Watch deleted successfully!');</script>";
            echo     "<script>setTimeout(function(){     window.location     =
'Watches.php'; }, 200);</script>";
            exit;
        } else {
            echo "Error deleting record: " . htmlspecialchars($del->error);
        }
        $del->close();
    } else {
        echo "Prepare failed: " . htmlspecialchars($conn->error);
    }
}
?>
```

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Add Watch - Admin Dashboard</title>
  <link rel="stylesheet" href="style.css" />
  <style>
    /* small inline improvements for image thumbnails */
    .thumb {
      max-width: 100px;
      max-height: 70px;
      object-fit: cover;
      border-radius: 6px;
      border: 1px solid rgba(255, 255, 255, 0.08);
    }
  </style>
</head>

<body class="body">

  <header class="header">
    <h1>Admin Dashboard</h1>
  </header>

  <?php include("partials/nav.php"); ?>

  <section class="section">
    <!-- enctype is required for file uploads -->
    <form method="POST" id="addWatchForm" class="form" enctype="multipart/form-data">
      <label class="label" for="title">Watch Name:</label>
      <input type="text" id="title" name="title" required class="input">

      <label class="label" for="Brand">Brand:</label>
      <input type="text" id="Brand" name="Brand" required class="input">

      <label class="label" for="Price">Price:</label>
      <input type="text" id="Price" name="Price" required class="input">

      <label class="label" for="Discription">Description:</label>
      <input type="text" id="Discription" name="Discription" required class="input">

      <label class="label" for="C">Category:</label>
```

```
        <input type="text" id="C" name="C" required class="input">

        <label class="label" for="Quantity">Quantity:</label>
        <input type="number" id="Quantity" name="Quantity" min="0"
required class="input">


        <label class="label" for="image">Watch Image
(JPG/PNG/GIF/WEBP, max 2MB):</label>
        <input type="file" id="image" name="image" accept="image/*"
class="input">

        <button type="submit" class="button">Add Watch</button>
      </form>

      <table class="table">
        <thead>
          <tr>
            <th>Image</th>
            <th>Watch Name</th>
            <th>Brand</th>
            <th>Price</th>
            <th>Quantity</th>
            <th>Orders Count</th>
            <th>Action</th>
          </tr>
        </thead>
        <tbody>
          <?php
          if ($result && $result->num_rows > 0) {
            while ($row = $result->fetch_assoc()) {
              // Use htmlentities or htmlspecialchars to prevent XSS
              $id = (int) $row['id'];
              $title = htmlspecialchars($row['title']);
              // try both Brand or brand depending on your column
              $brand = isset($row['Brand']) ?
htmlspecialchars($row['Brand']) : (isset($row['brand']) ?
htmlspecialchars($row['brand']) : '');
              $price = htmlspecialchars($row['Price'] ?? $row['price']
?? '');
              $image = !empty($row['image']) ?
htmlspecialchars($row['image']) : null;
              $quantity = (int)($row['quantity'] ?? 0);
              $orders_count = (int)($row['orders_count'] ?? 0);

              ?>
              <tr>
                <td>
                  <?php if ($image && file_exists(__DIR__ . '/' .
$image)): ?>
```

```
                        <img src="../img/Watch.webp" alt="watch" class="thumb">
                    <?php elseif ($image): ?>
                        <!-- if stored image reference exists but file missing, still attempt to load -->
                        <img src="../img/Watch.webp" alt="watch" class="thumb">
                    <?php else: ?>
                        <span style="color:#aaa;font-size:13px;">No image</span>
                    <?php endif; ?>
                  </td>
                  <td><?php echo $title; ?></td>
                  <td><?php echo $brand; ?></td>
                  <td><?php echo $price; ?></td>
                  <td><?php echo $quantity; ?></td>
                  <td><?php echo $orders_count; ?></td>


                  <td>
                    <a href="Watches.php?id=<?php echo $id; ?>" onclick="return confirm('Delete this watch?');">
                      <button class="button delete" type="button">Delete</button>
                    </a>
                  </td>
                </tr>
              <?php }
            } else { ?>
              <tr>
                <td colspan="5" style="text-align:center;color:#ccc;">No watches found.</td>
              </tr>
            <?php } ?>
          </tbody>
        </table>
    </section>

</body>

</html>
```

Users Page:



Code:

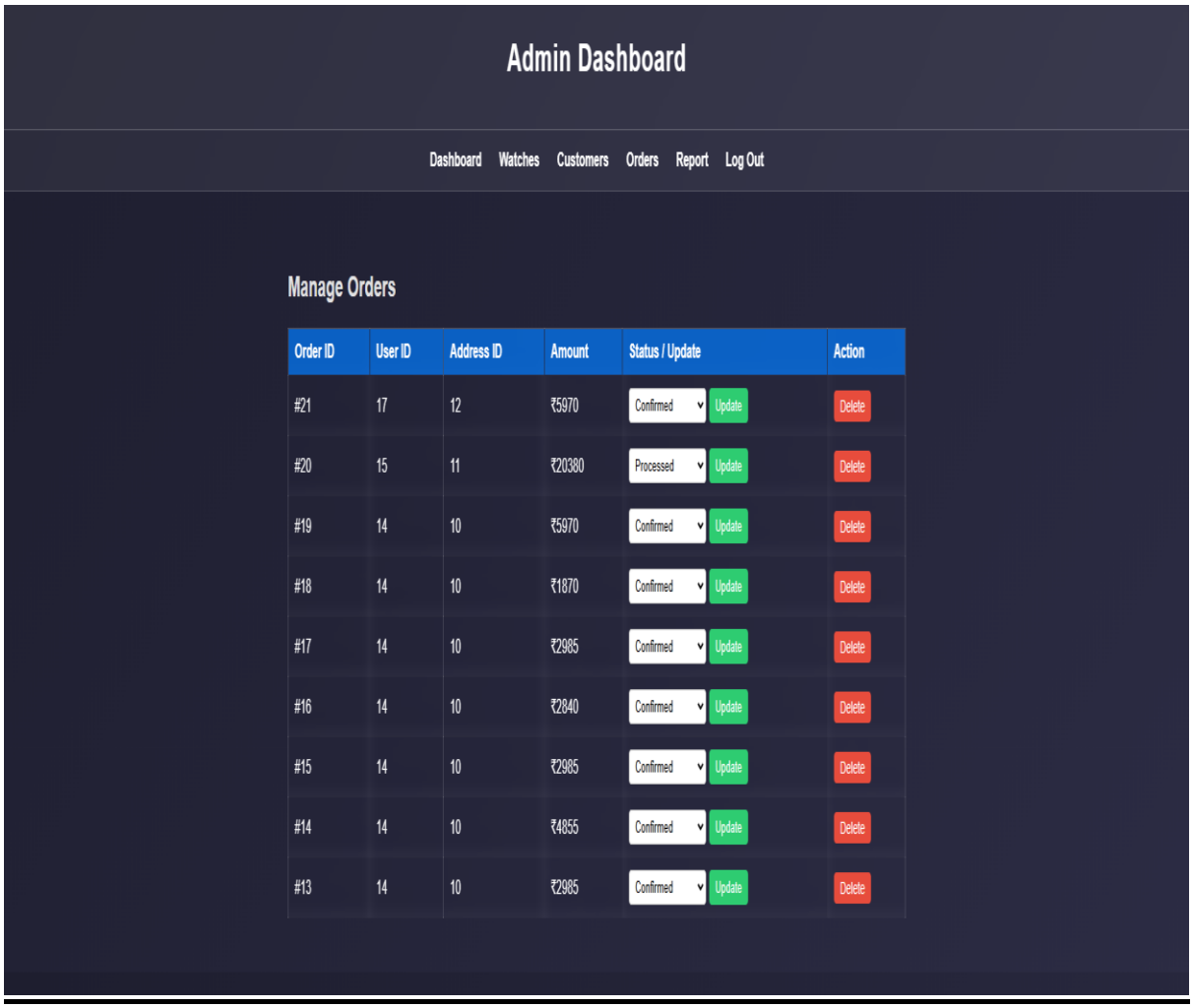
```
<?php
include("../partials/conn.php");

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Retrieve form data
    $username = $_POST['Name'];
    $email = $_POST['email'];
    $password = $_POST['password'];
    $Phone = $_POST['Phone'];
```

```
   if (!empty($username) && !empty($email) && !empty($password))
{


      $hashed_pass = password_hash($password,
PASSWORD_DEFAULT);

      $sql = "INSERT INTO user (name, email, password, phone)
VALUES ('$username', '$email', '$hashed_password', '$Phone')";

      if ($conn->query($sql) === TRUE) {
         echo "<script>alert('user added successful!')</script>";
         header("location: customer.php");

      } else {
         echo "Error: " . $sql . "<br>" . $conn->error;
      }

      $conn->close();
   }
}

// SQL query to retrieve data from the user table
$sql = "SELECT * FROM user"; // Change 'user' to your actual table
name

$result = $conn->query($sql);
if(isset($_GET['id'])) {
   $id = intval($_GET['id']);

   $sql = "DELETE FROM user WHERE id = $id"; // Change 'user' to
your actual
   if ($conn->query($sql) === TRUE) {
      echo "<script>alert('User Deleted successful!')</script>";
      header("location: customer.php");
   } else {
      echo "Error deleting record: " . $conn->error;
   }
}
?>

<!DOCTYPE html>
<html lang="en">

<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-
scale=1.0">
   <title>Add Car - Admin Dashboard</title>
   <link rel="stylesheet" href="style.css">
```

```
</head>

<body class="body">

   <header class="header">
      <h1>Admin Dashboard</h1>
   </header>

   <?php include("partials/nav.php"); ?>

   <section class="section">
      <form method="POST" id="addCarForm" class="form">
         <label for="Name" class="label">User Name:</label>
         <input type="text" id="Name" name="Name" required
class="input">

         <label for="email" class="label">Email:</label>
         <input type="text" id="email" name="email" required
class="input">

         <label for="password" class="label">Password:</label>
         <input type="text" id="password" name="password" required
class="input">

         <label for="phone" class="label">Phone:</label>
         <input type="text" id="Phone" name="Phone" required
class="input">

         <button type="submit" class="button">Add User</button>
      </form>

      <table class="table">
         <thead>

            <tr>
               <th>User Name</th>
               <th>Email</th>
               <th>Phone</th>
               <th>Action</th>
            </tr>
         </thead>
         <tbody>
            <?php

            if ($result->num_rows > 0) {
               // Output data of each row
               while ($row = $result->fetch_assoc()) {
                  ?>
                  <tr>
                     <td>
```

```
                    <?php echo " . $row['name'] . " ?>
                  </td>
                  <td>
                    <?php echo " . $row['email'] . " ?>
                  </td>
                  <td>
                    <?php echo " . $row['phone'] . " ?>
                  </td>

                    <td><a href="customer.php?id=<?php echo " . $row['id'] . " ?>"><button class="button delete">Delete</button></a></td>

                </tr>
              <?php }
            } ?>
          </tbody>
        </table>
    </section>

</body>

</html>
```

Manager Orders Page:



Code:

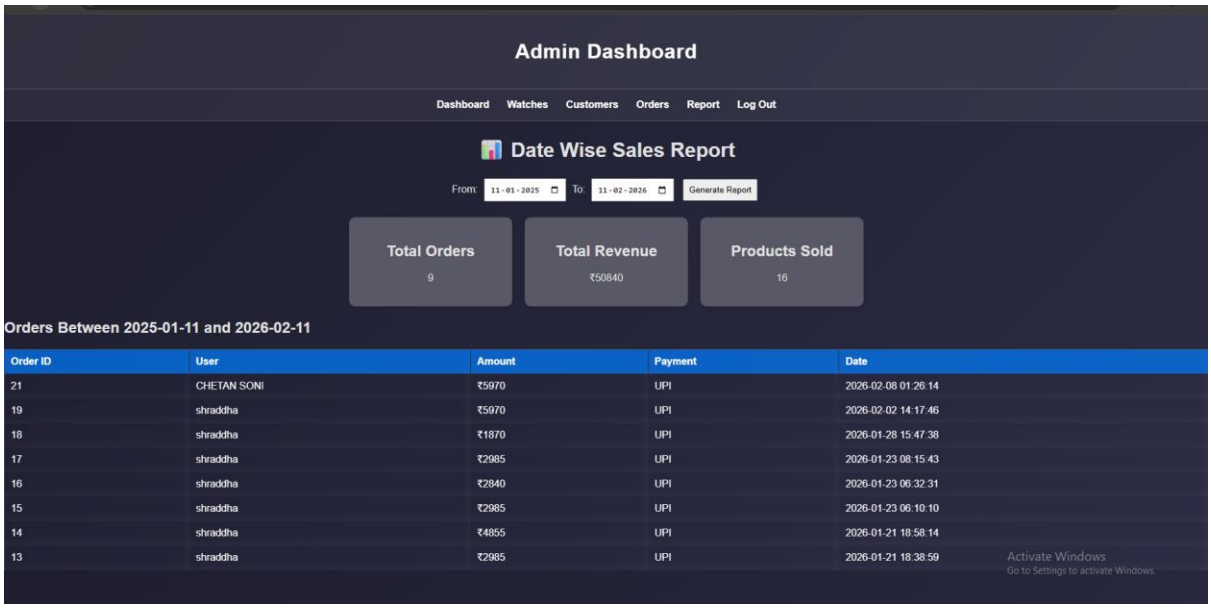
```
<?php
include("../partials/conn.php");

if (isset($_POST['update_status'])) {
    $order_id = intval($_POST['order_id']);
    $new_status = $_POST['status'];

    // Base update query
    $sql = "UPDATE orders SET status = '$new_status' ";

    // Add specific timestamp based on the new status
    if ($new_status == 'processed') {
```

```
        $sql .= ", processed_at = NOW() ";
    } elseif ($new_status == 'shipped') {
        $sql .= ", shipped_at = NOW() ";
    } elseif ($new_status == 'out_for_delivery') {
        $sql .= ", out_for_delivery_at = NOW() ";
    } elseif ($new_status == 'delivered') {
        $sql .= ", delivered_at = NOW() ";
    }

    $sql .= " WHERE id = $order_id";

    if ($conn->query($sql) === TRUE) {
        echo "<script>alert('Status updated successfully!');
window.location.href='order.php';</script>";
    } else {
        echo "<script>alert('Error updating status: " . $conn->error .
"');</script>";
    }
}

// --- 2. HANDLE DELETE ---
if (isset($_GET['delete_id'])) {
    $id = intval($_GET['delete_id']);
    $sql = "DELETE FROM orders WHERE id = $id";
    if ($conn->query($sql) === TRUE) {
        echo "<script>alert('Record deleted successfully!');
window.location.href='order.php';</script>";
    } else {
        echo "Error deleting record: " . $conn->error;
    }
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Manage Orders - Admin Dashboard</title>
  <link rel="stylesheet" href="style.css">
  <style>
      /* Simple CSS to make the status dropdown look good */
      .status-form {
          display: flex;
          gap: 5px;
      }
      .status-select {
          padding: 5px;
          border-radius: 4px;
```

```
        border: 1px solid #ccc;
      }
      .btn-update {
        background-color: #2ecc71;
        color: white;
        border: none;
        padding: 5px 10px;
        cursor: pointer;
        border-radius: 4px;
      }
      .btn-delete {
        background-color: #e74c3c;
        color: white;
        padding: 5px 10px;
        text-decoration: none;
        border-radius: 4px;
        font-size: 14px;
      }
  </style>
</head>
<body class="body">

  <header class="header">
    <h1>Admin Dashboard</h1>
  </header>

  <?php include("partials/nav.php"); ?>

  <section class="section">
    <h2>Manage Orders</h2>
    <table class="table">
      <thead>
      <tr>
        <th>Order ID</th>
        <th>User ID</th>
        <th>Address ID</th>
        <th>Amount</th>
        <th>Status / Update</th> <th>Action</th>
      </tr>
      </thead>
      <tbody>
      <?php
      $sql = "SELECT * FROM orders ORDER BY created_at DESC"; //
Show newest first
      $result = $conn->query($sql);

      if ($result->num_rows > 0) {
        while ($row = $result->fetch_assoc()) {
          $current_status = $row['status'];
      ?>
```

```
        <tr>
          <td>#<?= $row['id'] ?></td>
          <td><?= $row['user_id'] ?></td>
          <td><?= $row['address_id'] ?></td>
          <td>₹<?= $row['total_amount'] ?></td>

          <td>
            <form method="POST" class="status-form">
              <input type="hidden" name="order_id" value="<?= $row['id']
?>">

              <select name="status" class="status-select">
                <option value="confirmed" <?=
$current_status=='confirmed' ? 'selected' : '' ?>>Confirmed</option>
                <option value="processed" <?=
$current_status=='processed' ? 'selected' : '' ?>>Processed</option>
                <option value="shipped" <?=
$current_status=='shipped' ? 'selected' : '' ?>>Shipped</option>
                <option value="out_for_delivery" <?=
$current_status=='out_for_delivery' ? 'selected' : '' ?>>Out for
Delivery</option>
                <option value="delivered" <?=
$current_status=='delivered' ? 'selected' : '' ?>>Delivered</option>
                <option value="cancelled" <?=
$current_status=='cancelled' ? 'selected' : '' ?>>Cancelled</option>
              </select>

              <button type="submit" name="update_status" class="btn-
update">Update</button>
            </form>
          </td>

          <td>
            <a href="order.php?delete_id=<?= $row['id'] ?>"
onclick="return confirm('Are you sure?')" class="btn-
delete">Delete</a>
          </td>

        </tr>
      <?php }
      } else {
        echo "<tr><td colspan='6'>No orders found</td></tr>";
      }
      ?>
      </tbody>
    </table>
  </section>

</body>
</html>
```

Reports Page:



Code:

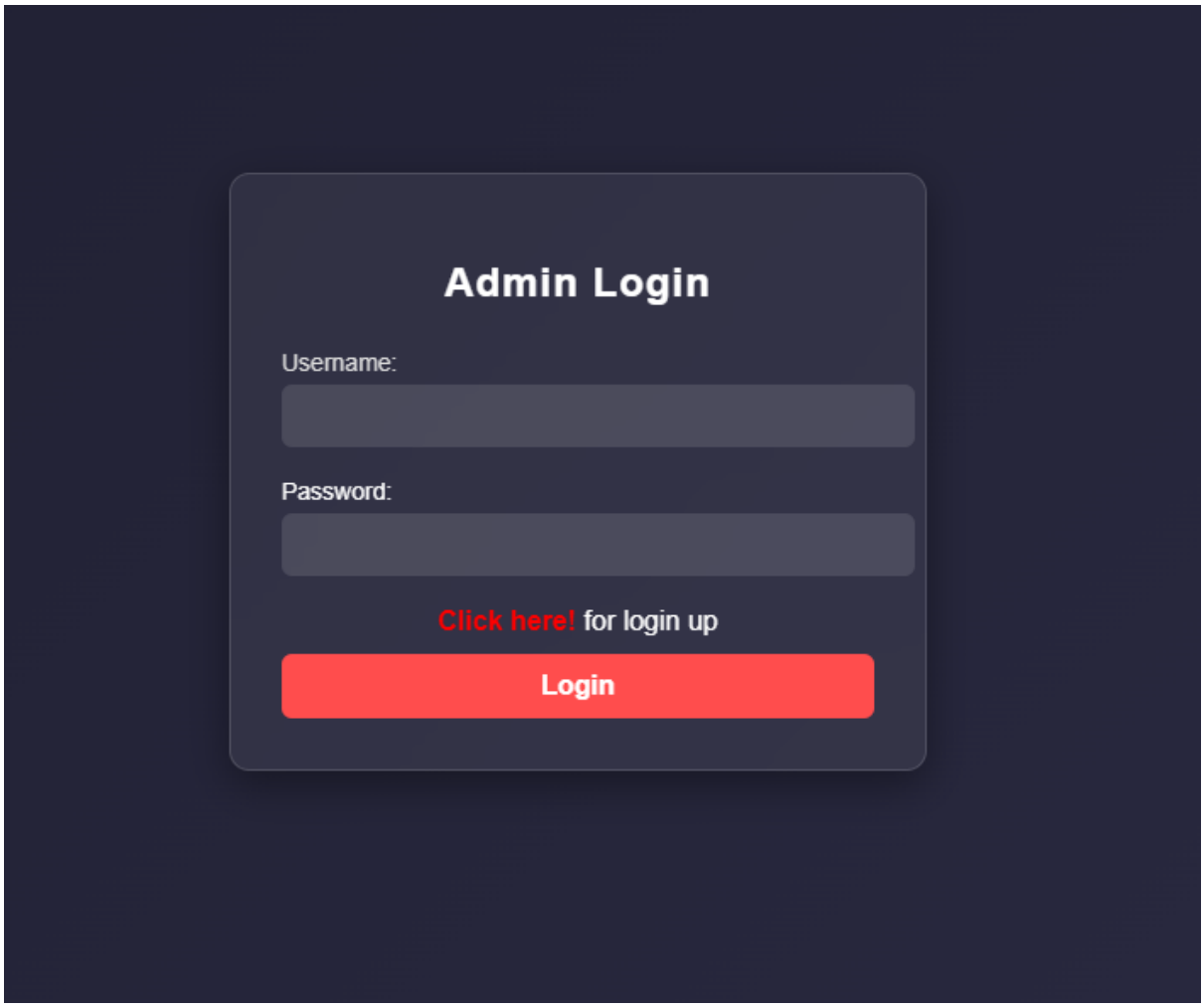
```
<?php
$conn = new mysqli("localhost", "root", "ChetanRootUser-PWD_Store-
7729", "watch");

if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}

$from = $_GET['from_date'] ?? '';
$to   = $_GET['to_date'] ?? '';

$total_orders = 0;
$total_revenue = 0;
$total_sold = 0;
$list_result = null;

if ($from && $to) {

   // Orders Count
   $order_q = "SELECT COUNT(*) AS total_orders
            FROM orders
            WHERE DATE(created_at) BETWEEN '$from' AND '$to'";
```

```
    $total_orders = $conn->query($order_q)->fetch_assoc()['total_orders'];

    // Revenue
    $rev_q = "SELECT SUM(total_amount) AS total_revenue
              FROM orders
              WHERE DATE(created_at) BETWEEN '$from' AND '$to'";
    $total_revenue = $conn->query($rev_q)->fetch_assoc()['total_revenue'] ?? 0;

    // Products Sold
    $sold_q = "SELECT SUM(oi.quantity) AS total_sold
               FROM order_items oi
               JOIN orders o ON oi.order_id = o.id
               WHERE DATE(o.created_at) BETWEEN '$from' AND '$to'";
    $total_sold = $conn->query($sold_q)->fetch_assoc()['total_sold'] ?? 0;

    // Orders List
    $list_q = "SELECT o.id, u.name, o.total_amount, o.payment_method, o.created_at
               FROM orders o
               JOIN user u ON o.user_id = u.id
               WHERE DATE(o.created_at) BETWEEN '$from' AND '$to'
               ORDER BY o.created_at DESC";
    $list_result = $conn->query($list_q);
}
?>

<!DOCTYPE html>
<html>
<head>
    <title>Admin Sales Report</title>
    <style>
        body { font-family: Arial; background:#f4f6f9; padding:20px }
        h1 { text-align:center }
        form { text-align:center; margin-bottom:20px }
        input, button { padding:8px; margin:5px }
        .card-container { display:flex; justify-content:center; gap:20px; flex-wrap:wrap }
        .card {
            background: #585866;
            padding:20px;
            width:220px;
            border-radius:10px;
            box-shadow:0 4px 8px rgba(0,0,0,0.1);
            text-align:center;
        }
    </style>
    <link rel="stylesheet" href="style.css">
```

```
</head>
<body>

  <header class="header">
    <h1>Admin Dashboard</h1>
  </header>

  <?php include("partials/nav.php"); ?>
<h1>📊 Date Wise Sales Report</h1>

<form method="GET">
   <label>From:</label>
   <input type="date" name="from_date" required value="<?= $from
?>">

   <label>To:</label>
   <input type="date" name="to_date" required value="<?= $to ?>">

   <button type="submit">Generate Report</button>
</form>

<?php if($from && $to): ?>

<div class="card-container">
   <div class="card">
      <h2>Total Orders</h2>
      <p><?= $total_orders ?></p>
   </div>

   <div class="card">
      <h2>Total Revenue</h2>
      <p>₹<?= $total_revenue ?></p>
   </div>

   <div class="card">
      <h2>Products Sold</h2>
      <p><?= $total_sold ?></p>
   </div>
</div>

<h2>Orders Between <?= $from ?> and <?= $to ?></h2>

<table class="table">
<tr>
   <th>Order ID</th>
   <th>User</th>
   <th>Amount</th>
   <th>Payment</th>
   <th>Date</th>
</tr>
```

```
<?php while($row = $list_result->fetch_assoc()): ?>
<tr>
   <td><?= $row['id'] ?></td>
   <td><?= $row['name'] ?></td>
   <td>₹<?= $row['total_amount'] ?></td>
   <td><?= $row['payment_method'] ?></td>
   <td><?= $row['created_at'] ?></td>
</tr>
<?php endwhile; ?>

</table>

<?php endif; ?>

</body>
</html>
```

Admin Login Page:



Code:

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    include"../partials/conn.php";
    $user_entered = $_POST['username'];
    $pass_entered = $_POST['password'];

    // Prepare a SQL query to fetch the user with entered username
    $sql = "SELECT id, name, password FROM admin WHERE name =
'$user_entered'";
    $result = $conn->query($sql);
```

```
    if ($result->num_rows > 0) {
        // User exists in the database
        $row = $result->fetch_assoc();
        $stored_password = $row['password']; // Password stored in the database

        // Verify the entered password against the stored password
        if ($pass_entered== $stored_password) {
            // Passwords match - authentication successful
            session_start();
            $_SESSION['loggedin'] = true;
            $_SESSION['username'] = $user_entered;
            $_SESSION['c_ID'] = $row['id'];;
            header('location: dashboard.php');
        } else {
            echo "<script>alert('Pleace check your password')</script>";
        }
    } else {
        echo "User does not exist. Please register or try a different username.";
    }
}
?>

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>
        /* -------------------------
    GENERAL PAGE STYLING
-------------------------*/
body {
    margin: 0;
    padding: 0;
    font-family: Arial, sans-serif;
    background: linear-gradient(135deg, #1e1e2f, #2c2c44);
    height: 100vh;
    display: flex;
    justify-content: center;
    align-items: center;
}

/* -------------------------
    LOGIN CONTAINER
-------------------------*/
.login-container {
```

```
    width: 350px;
    padding: 30px;
    background: rgba(255, 255, 255, 0.06);
    border-radius: 12px;
    backdrop-filter: blur(10px);
    border: 1px solid rgba(255, 255, 255, 0.15);
    box-shadow: 0 8px 25px rgba(0, 0, 0, 0.4);
}

/* Card inner container */
.hmmm h2 {
    text-align: center;
    margin-bottom: 25px;
    color: #fff;
    font-weight: bold;
    letter-spacing: 1px;
}

/* ------------------------
    INPUT FIELDS
------------------------*/
.form-group {
    margin-bottom: 18px;
}

label {
    display: block;
    color: #ddd;
    margin-bottom: 5px;
    font-size: 14px;
}

input[type="text"],
input[type="password"] {
    width: 100%;
    padding: 10px 12px;
    border: none;
    border-radius: 6px;
    background: rgba(255, 255, 255, 0.12);
    color: #fff;
    outline: none;
    font-size: 15px;
    transition: 0.3s;
}

input:focus {
    background: rgba(255, 255, 255, 0.2);
    box-shadow: 0 0 8px rgba(0, 255, 150, 0.7);
}
```

```
/* ------------------------
   LOGIN BUTTON
-------------------------*/
.login-btn {
   width: 100%;
   padding: 10px;
   background: #ff4d4d;
   border: none;
   border-radius: 6px;
   color: #fff;
   font-size: 16px;
   cursor: pointer;
   font-weight: bold;
   transition: 0.3s;
}

.login-btn:hover {
   background: #ff2626;
   box-shadow: 0 0 10px rgba(255, 50, 50, 0.6);
}

/* ------------------------
   LINKS
-------------------------*/
.la {
   text-align: center;
   margin-bottom: 10px;
}

.la a {
   color: #ff4d4d;
   font-weight: bold;
   text-decoration: none;
}

.la a:hover {
   text-decoration: underline;
}

   </style>
   <title>Login Page</title>
</head>

<body>

   <div class="login-container">
      <div class="hmmm">

         <h2> Admin Login</h2>
         <form method="POST">
```

```
            <div class="form-group">
                <label for="username">Username:</label>
                <input type="text" id="username" name="username"
required>
            </div>
            <div class="form-group">
                <label for="password" style="color:
white;">Password:</label>
                <input type="password" id="password" name="password"
required>
            </div>
            <p class="la" style="color: #ffff"><a
href="../login.php"style="color: red;">Click here!</a> for login up</p>
            <button type="submit" class="login-btn">Login</button>
        </form>
    </div>
  </div>

</body>

</html>
```

LIMITATIONS

1. Lack of Multi-Language Support: The **Shraddha Watch Store** website currently operates solely in English, which limits its usability for non-English-speaking customers. In a diverse market where regional languages play a significant role in consumer behavior, this restriction reduces the platform's reach. Users who are more comfortable browsing and shopping in their native languages may find it challenging to navigate product specifications, read return policies, or complete the checkout process, potentially leading to a loss of sales from this demographic.

2. Absence of Real-Time Customer Support (Live Chat): The platform does not currently offer a direct chat or instant messaging feature for real-time communication between customers and the store administration. This limitation impedes immediate interaction, meaning customers cannot instantly resolve queries regarding watch features, warranty details, or stock availability. The reliance on email or standard contact forms may cause delays in response times, which can negatively impact customer satisfaction and trust during the pre-purchase phase.

3. Basic Product Recommendation Engine: The current product recommendation engine functions in a basic capacity, suggesting watches primarily based on simple category matching (e.g., "Show other Analog Watches") rather than sophisticated user behavior analysis. It lacks advanced algorithms like collaborative filtering (e.g., "Customers who bought this also bought..."). As a result, the recommendations may not be highly personalized to the user's specific style preferences or budget, leading to lower cross-selling opportunities and a less engaging shopping experience.

FUTURE SCOPE OF SHRADDHA WATCH STORE

1. AI-Driven Product Recommendations and Styling: Shraddha Watch Store can integrate artificial intelligence (AI) and machine learning algorithms to enhance product discovery based on user browsing history, purchase patterns, and style preferences. AI-driven recommendations can suggest watches that match a user's specific taste (e.g., "If you like this diver watch, you might also like..."), improving conversion rates and customer satisfaction.

2. Augmented Reality (AR) Try-On Feature: Implementing Augmented Reality (AR) technology would allow customers to virtually "try on" watches using their smartphone camera. This feature helps users visualize how a watch looks on their wrist regarding size and style, significantly reducing return rates and boosting buyer confidence.

3. Advanced Search and Voice Commerce: Implementing advanced search functionalities, including visual search (uploading an image to find similar watches) and voice-enabled search commands (e.g., "Show me gold analog watches under ₹5000"), will enhance user convenience and accessibility, catering to a tech-savvy customer base.

4. Expansion into Global Markets and Multi-Currency Support: The platform can extend its services beyond domestic shipping to international markets. This expansion would involve integrating multi-currency support and international payment gateways, allowing watch enthusiasts worldwide to purchase from **Shraddha Watch Store**.

5. AI-Powered Chatbots and Virtual Shopping Assistants: Implementing AI-driven chatbots can enhance user engagement by providing 24/7 assistance for customers. Virtual shopping assistants can answer queries about watch specifications, warranty details, and order status in real-time, simulating the experience of interacting with a knowledgeable salesperson in a physical store.

CONCLUSION

The **Shraddha Watch Store** project successfully addresses the key objectives of providing a comprehensive and user-friendly platform for watch enthusiasts and online shoppers. By offering a diverse range of watch collections, personalized product recommendations, and the ability to save favorite items to a wishlist, the platform enhances the digital shopping experience. For administrators, the platform simplifies the process of inventory management and order processing, making the retail operation more efficient.

The scope of the project includes creating a scalable system that can handle the growing demands of an online business, with a focus on secure user authentication, advanced product search functionalities, and a responsive design optimized for mobile devices. The **Model-View-Controller (MVC)** architecture ensures that the system is modular and maintainable, while the **MySQL** database allows for efficient management of product catalogs and user profiles. The project also adheres to essential security principles to protect sensitive customer data and prevent unauthorized access.

In conclusion, **Shraddha Watch Store** provides an efficient, scalable, and secure solution for connecting customers with high-quality timepieces. It meets the platform's goals by ensuring ease of use, visual appeal, and streamlined workflows for both buyers and store administrators. With future enhancements such as AI integration and global shipping capabilities, **Shraddha Watch Store** is poised to become a trusted name in the competitive e-commerce landscape.