**Project Report**

**On**

**E-COMMERCE WEBSITE**

**CSC Computer Education**

**FULL STACK PYTHON**

**GUIDED BY: - MR. Mr. R DEIVASIGAMANI, MSc.,**

**PREPARED BY:- P BALAGANESH**

****

**YEAR-2023-2024**

**SUBMITTED TO**

**CSC COMPUTER EDUCATION**

**Branch : PERUNDURAI**

**64 First Floor, Amman Complex, Opp New Bus Stand,**

**Perundurai. Erode(district). 638052.**

**Email:** [**cscperundurai@gmail.com**](mailto:cscperundurai@gmail.com)**,** [**perunduraicsc@gmail.com**](mailto:perunduraicsc@gmail.com)

**Phone: 7200001338,9965553027**

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Chapter** | **Title** | **Page No.** |
| **1** | **Project Overview** | 1 |
|  | Project Title | 1 |
| Project Description |
| Project Objective | 2 |
| **2** | **Requirements** | 3 |
|  | Functional Requirements | 3 |
| Non-Functional Requirements |
| Technical Requirements |
| **3** | **System Architecture** | 5 |
|  | High-Level Architecture | 4 |
| Components |
| **4** | **Design** | 6 |
|  | Database Design |  |
| Application Flow |
| **5** | **Implementation** | 7 |
|  | Project Setup |  |
| Code Structure |
| **6** | **Testing** | 8 |
| **7** | **References** | 10 |

**1. Project Overview**

**1.1. Project Title**

**Title:** E-Commerce Website

**1.2. Project Description**

**Description: Welcome to Art Haven**

Explore a curated collection of exquisite drawings at Art Haven, your premier online destination for art enthusiasts and collectors. Whether you’re looking for contemporary pieces, classic sketches, or unique illustrations, our store offers a diverse range of artworks to suit every taste and style.

**Why Choose Us?**

**Exclusive Collection**: Discover a wide variety of drawings from talented artists around the world.

* **Authenticity Guaranteed**: Each piece is verified for authenticity, ensuring you receive genuine artwork.
* **Secure Transactions**: Shop with confidence using our secure payment methods and data protection measures.
* **Fast Delivery**: Enjoy prompt and reliable shipping to get your artwork delivered safely to your doorstep.
* **Customer Support**: Our dedicated team is here to assist you with any questions or concerns.

**How It Works**

1. **Browse**: Navigate through our user-friendly website to explore our extensive collection of drawings.
2. **Select**: Choose your favorite pieces in cart and add them to your cart.
3. **Purchase**: Proceed to checkout, provide your shipping details, and select your preferred payment method.
4. **Receive**: Sit back and relax while we carefully package and ship your artwork to you.

Join our community of art lovers and start building your collection with Art Haven. Happy shopping!

**1.3. Objectives**

* To develop a user-friendly web application using Flask.
* The customer can login in web page.
* The customer data is store in database and before login run the python file of user.
* To buy the products click cart icon in the navigation bar.
* Select the products to buy in cart.
* After selecting the products in the cart, click pay now button.
* Fill the details in payment form to buy the products.
* After completing the payment form, click order now button to place the order.
* The customer order is confirmed.
* The customer will receive the product within 2 or 3 days.
* The customer can contact us by filling the contact us details in web page.

**1.4. Target Audience**

**Audience:** Who loves and interested in the art work. They will buy our products. Everyone can buy our products, our products are made for every customers to satisfy them.

**2. Requirements**

**2.1. Functional Requirements**

* User registration and login functionality.
* Shopping cart, Checkout and Order.
* Customer support, return and exchanges.
* Integration with payments and shipping charge is free.

**2.2. Non-Functional Requirements**

* Performance: The application should handle up to 1000 simultaneous users.
* Security: User data must be protected, and password hashing should be implemented.
* Usability: The application should be intuitive and user-friendly.

**2.3. Technical Requirements**

* Programming Language: Python as a can simply be referred to as a multi-functionality having capabilities to operate in multiple ways. Surprisingly a developer can do wonders and can easily develop software by implementing easy methods. Being an interpreted language (smoothness in dev. process), this platform is absolutely free and open for all, it also offers platform independency, meaning their (Python’s) code can run on any platform without making any changes (such as Linux, macOS, etc.)
* Framework: Flask is a lightweight backend framework with minimal dependencies. Flask is easy to learn because its simple and intuitive API makes it easy to learn and use for beginners. Flask is a flexible Framework because it allows you to customize and extend the framework to suit your needs easily. Flask can be used with any database like:- SQL and NoSQL and with any Frontend Technology such as React or Angular. Flask is great for small to medium projects that do not require the complexity of a large framework.
* Database: A database is an organized collection of data stored in a computer system and usually controlled by a database management system (DBMS). The data in common databases is modeled in tables, making querying and processing efficient. Structured query language (SQL) is commonly used for data querying and writing. The Database is an essential part of our life. We encounter several activities that involve our interaction with databases, for example in the bank, in the railway station, in school, in a grocery store, etc. These are the instances where we need to store a large amount of data in one place and fetch these data easily.

**Frontend:**

* + **HTML: Hypertext markup language (HTML)** is a Hypertext markup **language, the standard markup language for documents designed to displayed** and viewed on the online during a browser also helps to create the structure of the web page. because it is a markup language, it consists of many tags. There are tags to display text, tables, ordered lists and unordered lists, etc. There are two main sections on the HTML page: head and body section. The data that describes the page also termed as metadata is inside the head section while the body section includes all the tags that are necessary to represent the visible content of the web page HTML is a platform-independent language so that can be made in use in any platform like Windows, Linux, Macintosh, etc.
  + **CSS:** **Cascading Style Sheets (CSS)** is defined as a method sheet language that provides web designers control over how an internet site communicates with web browsers including the formatting and display of their HTML documents.CSS or cascading sheet may be a text-based coding language that specifies the website formats and the way a site communicates with web browsers. The language allows web developers to regulate various style elements and functionalities, like layout, color, fonts, and therefore the formatting and display of HTML documents. The main goal (as a method sheet language) was to separate document content from document presentation, which incorporates style elements, like color, layout, and fonts. CSS handles the design and feel a part of an internet page. Using CSS, you will control the color of the text, the design of fonts, the spacing between paragraphs, how columns are sized and laid out, etc. CSS instructs the display of the HTML on how the web site will display at the user’s end.
  + **JavaScript:** It is an interpreted programming as well as a scripting language. Many of these are related to the way, [it](https://www.geeksforgeeks.org/introduction-to-javascript/) is often executed directly in a client’s browser commonly utilized in web development. It was originally developed by Netscape as a way to feature dynamic and interactive elements on websites. JavaScript is influenced by Java with a similar syntax to C. JavaScript follows the ECMAScript specifications which were developed by Sun Microsystems. Like server-side scripting languages such as PHP and ASP, JavaScript code is often inserted anywhere within the HTML of a webpage. The output of the server side is displayed in the HTML, but the JavaScript code remains visible in the source of the webpage. The file can be a separate “.js” file, which can be displayed in the browser.

**3. System Architecture**

**3.1. High-Level Architecture**

A diagram or description of the system architecture, including:

* **Client-side:** The user interface in the browser.
* **Server-side:** Flask application handling requests and responses.
* **Database:** Storage for user data and tasks.
* **APIs:** Any external services or APIs used.

**3.2. Components**

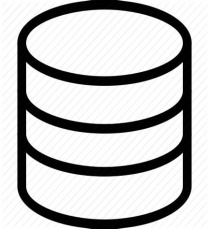
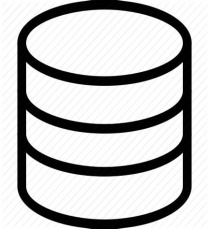
* **Frontend:** HTML/CSS/JavaScript templates for user interaction.
* **Backend:** Flask routes and handlers for business logic.
* **Database:** Schema and models for user data and tasks.

Home

User

Contact Us

Cart

** **

Payment

Order

Database

Database

**4. Design**

**4.1. Database Design**

* **Table-1: Customer Comment’s Data**
  + Username varchar(20),Email varchar(30),Phonenumber varchar(15),Comment varchar(150)
* **Table-2: Login User Data**
  + Username varchar(30),Email varchar(40),Password varchar(20),Phonenumber varchar(10),Address varchar(50),Pincode varchar(6)
* **Relationships:** How tables relate to one another (e.g., one-to-many relationship between users and tasks).
* **Schema:** Detailed schema definition for each table.

**4.2. Application Flow**

* **User Flow:** Diagram or description of how users interact with the application.
* **Request-Response Flow:** How the application processes requests and sends responses.

**5. Implementation**

**5.1. Project Setup**

* **Dependencies:** List of required libraries and tools (e.g., Flask, SQLAlchemy).
* **Installation Instructions:** Steps to set up the development environment.

**5.2. Code Structure**

**File Organization:**

**Key Modules:** Explanation of important modules and their responsibilities.

**5.3. Example Code**

Provide snippets of critical code for key functionalities, such as user authentication or task management.

**6. Testing**

**6.1. Testing Strategy**

* **Unit Testing:**

Unit testing is the process where you test the smallest functional unit of code. Software testing helps ensure code quality, and it's an integral part of software development. It's a software development best practice to write software as small, functional units then write a unit test for each code unit. You can first write unit tests as code. Then, run that test code automatically every time you make changes in the software code. This way, if a test fails, you can quickly isolate the area of the code that has the bug or error. Unit testing enforces modular thinking paradigms and improves test coverage and quality. Automated unit testing helps ensure you or your developers have more time to concentrate on coding.

To create unit tests, you can follow some basic techniques to ensure coverage of all test cases.

**Logic checks**

Does the system perform the right calculations and follow the right path through the code given a correct, expected input? Are all paths through the code covered by the given inputs?

**Boundary checks**

For the given inputs, how does the system respond? How does it respond to typical inputs, edge cases, or invalid inputs?

Let’s say you expect an integer input between 3 and 7. How does the system respond when you use a 5 (typical input), a 3 (edge case), or a 9 (invalid input)?

* **End-to-End Testing:**

Horizontal E2E Testing: This checks a specific layer or component works consistently across different parts of your software.

For instance, if you have a messaging app, horizontal testing will ensure that sending messages works well on different devices, like phones, tablets, and laptops.

Vertical E2E Testing: This ensures a particular function works smoothly from the front-end interface down to the server and database.

If we stick with the messaging app example, vertical testing will verify that not only can you send messages across devices, but the whole process works — from typing a message, to the server processing it, to the recipient getting it.

**End-to-End Testing Process**

The process for an end-to-end test requires an analysis of the test to be carried out. Studying the systems and subsystems of the application will determine the design of the test. Below are the steps required to complete an end-to-end test.

1. To begin with, study the requirements of the software. Get a good understanding of the application’s functions and workflow.

2. Then, prepare the test environment in alignment with the hardware and software requirements.

3. Next, understand the system and subsystems. Highlight the responses.

4. Now, define the methods necessary for testing these responses. These include end-to-end testing tools.

5. Afterward, create the test cases using the established testing standards.

6. Execute the defined test. In addition, perform exploratory testing to cover other areas undefined in the pre-determined test.

7. Finally, save and study the test results.

**Build user functions**

Identify features of the software and all the existing subsystems. Then, take note of the input and output data, action, and the relationships between them. Determine if the user functions are independent or reusable. For example, in a scenario where you want to transfer money from your bank account to a third-party subsystem, user functions will include the following:

1. Login into the banking application system

2. Check the current balance in the account

3. Transfer money to the third-party subsystem

4. Check the current balance after the transfer

5. Logout of the system

**Build conditions**

Ensure you build conditions peculiar to every user function. These could be data conditions, timing, and sequence. For example, during login, check for conditions like:

• Invalid Username and password

• Checking Password strength

• Checking for the validity of Username and password

• Checking error messages

While checking the balance amount, the following conditions can be built:

• Check for an error message if the transfer amount is more than the current balance

• Check the current balance 24 hours after the transfer is made to another bank

Build test scenarios/cases

Build specific test cases and scenerios for pre-defined user functions. Every condition must have a particular test assigned to it. In this banking system, test scenarios include:

• Transfer money to a third-party bank within the same locality

• Make an eCommerce transaction with bank details

• Pay international banks from banking application

**7. References**

List any resources, libraries, frameworks, or tools that were used or referenced during the project.

Web references:

* https://www.behance.net/
* https://www.w3schools.com/
* Alexandrou, M., n.d. , Web Development Methodology ,mariosalexandrou.com [blog]Available at
* Amit, R., Zott, C., 2001. Value Creation in E-business, StrategicManagement Journal, 2001, Vol.22(6-7),
* Barnes, S. J., Vidgen, R.T., 2002, An integrative approach to theassessment of e-commerce quality.
* Journal of Electronic Commerce Research, Vol. 3(3),p. 114, 2002.Available at:
* <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.83.8310&rep=rep1&type=pdf>
* Bauer, M., 2005, Successful Web DevelopmentMethodologies Article.