**Project Report**

**On**

**FOOD COURT**

**CSC Computer Education**

**Full stack python**

**GUIDED BY: PREPARED BY:-**

**MR. Mr. R DEIVASIGAMANI, MSc., S.S.Santhiya**

****

**YEAR-20xx-xx**

**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** | **Introduction** |  |
|  |  |  |
|  |
|  |
|  |
|  |
| **2** |  |  |
|  |  |  |
|  |
|  |
| **3** |  |  |
|  |  |  |
|  |
|  |
| **4** |  |  |
|  |  |  |
|  |
|  |
| **5** |  |  |
|  |  |  |
|  |
|  |
| **6** |  |  |
| **7** |  |  |
| **8** |  |  |

**1. Project Overview**

**1.1. Project Title**

**Title:Food court**

**1.2. Project Description**

**Description:**

Discover a world of delicious recipes and food inspiration.Our blog features mouthwatering dishes.Empowering your minds with healthy eating habits.Our food pamntry provides nutritious foods to families and individuals .

1.3. Objectives

* Provide high-quality,fresh and diverse food products .
* .
* Achieve sustainable profitability through efficient marketing,and strategic partnerships.
* Contribute positively to the local community.

**1.4. Target Audience**

* **Health-consious individuals:**People who prioritize healthy eating habits and seeking organic products.
* **Busy professionals:**

**2. Requirements**

**2.1. Functional Requirements**

* User registration and login functionality.
* Product catalog,search and filtering,product comparison.
* Shopping cart,checkout process
* Customer support,returns and exchanges.
* Integration with payment gateway,integration with shipping carriers.
* Marketing and promotions ,analytics and reporting.

**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: <Detailed Description about your Programming Language>
* Framework: Flask/ <web Framework Description>
* Database: <database Description>
* Frontend: HTML, CSS, JavaScript

**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.

**4. Design**

**4.1. Database Design**

* **Tables:** Users, Tasks, etc.
* **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:** Description of the file structure and 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:** Tests for individual components or functions.
* **Integration Testing:** Tests for interactions between components.
* **End-to-End Testing:** Tests for complete workflows from start to finish.

**6.2. Tools**

* **Testing Frameworks:** pytest, unittest, etc.
* **Mocking:** Use of mock objects or services for testing.

**7. References**

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