Low Level Design

(eCommerce – Online Cake Order Website)

|  |  |
| --- | --- |
| Written By | Anshul Maurya |
| Document Version | 1.0 |
| Last Revised Date | 13 – June -2023 |

**Document Control**

### Change Record:

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| 1.0 | 13/07/2023 | Anshul Maurya | Initial document of LLD created successfully |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Date Issued** | **Version** | **Description** | **Author** |
| 13/07/2023 | 1.0 | Initial LLD version – 1.0 | Anshul Maurya |

### Approval Status:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Review**  **Date** | **Reviewed By** | **Approved By** | **Comments** |
|  |  |  |  |  |

Contents

1. [Introduction 1](#_bookmark0)
   1. [What is a Low-Level design document? 1](#_bookmark1)
   2. [Scope 1](#_bookmark2)
2. Technology Stack……………………………………………………………………………………………………………………6
3. [Architecture and Design](#_bookmark4) 7
   1. [System Architecture 7](#_bookmark5)
   2. [Database Design 7](#_bookmark6)
   3. [Frontend Design 8](#_bookmark7)
   4. [Backend Design 8](#_bookmark8)
   5. [User Authentication and Authorization 8](#_bookmark9)
4. Development Environment Setup………………………………………………………………………………………………9

5.1. Installing Node.js and NPM…………………………………………………………………………………………………9

5.2. Setting up ReactJs Project………………………………………………………………………………………………..…9

5.3 Configuring ExpressJs Application………………………………………………………………………………….……9

5.4 Integrating Firebase Cloud Storage…………………………………………………………………,,…………………9

1. [Frontend Implementaion 10](#_bookmark20)
2. Backend Implementation……………………………………………………………………………………………………….11
3. Conclusion………………………………………………………………………………………………………………………………14
4. References………………………………………………………………………………………………………………………………15

# Introduction

## What is a Low-Level design document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Food Recommendation System. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer can directly code the program from the document.

## Scope

Low-level design (LLD) is a component-level design process that follows a step-by-

step [refinement](https://en.wikipedia.org/wiki/Refinement_(computing)) process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work.

# Technology Stack

The document addresses the requirement for various technical requirements for the proper usage of various features of website including the courses and the managing of the user database and smooth process for the instructor to upload the courses. The following technologies have been used-

* + - Firebase for database
    - Express and Nodejs for the backend.
    - ReactJs, MUI for frontend.

## 

## Architecture and Design

## 3.1 System Architecture:

Client-Server Architecture: The application will follow a client-server architecture, where the frontend and backend components are decoupled. The client side will be responsible for user interaction and rendering the UI using ReactJs, while the server side will handle business logic and data processing using Nodejs and ExpressJs.

RESTful API: The communication between the frontend and backend will be facilitated through a RESTful API, which will provide endpoints for various operations such as retrieving cake information, handling user authentication, managing shopping carts, and processing orders.

## 

## 3.2 Database Design:

Firebase Cloud Storage: Firebase Cloud Storage will be used to store various media files, including images of cakes, user avatars, and other multimedia assets.

NoSQL Database: Firebase Realtime Database or Firestore (both NoSQL databases) will be used to store dynamic data such as user information, cake details, orders, and shopping cart data. NoSQL databases provide the flexibility and scalability required for web applications.

## 3.3 Frontend Design:

Component-Based Architecture: The front end will be structured using a component-based architecture in ReactJs. Each UI element (e.g., header, footer, cake listing, shopping cart) will be implemented as reusable components for better organization and maintainability.

Responsive Design: The website will be designed to be responsive, ensuring a seamless user experience on various devices, including desktops, tablets, and mobile phones.

State Management: Redux or React Context API will be used for state management to handle the application's global state, including cart items, user authentication status, and other shared data.

## 3.4 Backend Design:

Middleware: ExpressJs middleware will be utilized to handle request pre-processing, such as authentication checks, CORS handling, logging, and error handling.

API Routing: ExpressJs will define various API routes with which the frontend can interact. These routes will be responsible for querying the database and returning data in response to front-end requests.

Server-Side Validation: Input data will be validated on the server side to prevent malicious or incorrect data from affecting the application's integrity.

Security Considerations: Authentication mechanisms like JWT (JSON Web Tokens) will be implemented for securing API endpoints and user sessions. Proper security measures will be taken to prevent common web vulnerabilities like Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF).

## 3.5 User Authentication and Authorization:

User Registration and Login: Users will be able to register and log in to the website using their email and password credentials or through social media authentication.

Role-Based Access Control: Different user roles (e.g., guest, registered user, admin) will have varying levels of access to certain features and functionalities.

Authentication Flow: The frontend and backend will communicate to verify user credentials during login, and upon successful authentication, the user will receive a token for subsequent authenticated requests.

## Development Environment Setup:

## 4.1 Installing Node.js and NPM:

Node.js Installation: Download and install the latest stable version of Node.js from the official website (https://nodejs.org/). Node.js includes both Node.js runtime and NPM (Node Package Manager) by default.

Verify Installation: After installation, open the command prompt (or terminal) and run the “node -v” and “npm -v” commands to ensure Node.js and NPM are installed correctly and displaying the installed versions.

## 4.2 Setting up ReactJs Project:

Create React App: Use the Create React App command-line tool to scaffold a new React project. Open the terminal and run “yarn create-react-app online-cake-order-website” to create a new project named "online-cake-order-website."

Project Structure: Understand the project structure created by Create React App, which includes various folders like src (for source code), public (for static files), and node modules (for installed dependencies).

## 4.3 Configuring ExpressJs Application:

Initialize Node.js Project: Create a new folder for the backend, navigate to it in the terminal, and run npm init to initialize a new Node.js project. Follow the prompts to set up package.json with the project's metadata and dependencies.

Install Express: Run npm install express to install Express.js, a minimalistic Node.js web application framework, which will handle the backend server functionality.

## 4.4 Integrating Firebase Cloud Storage:

Create Firebase Project: Go to the Firebase Console (https://console.firebase.google.com/), create a new project, and configure it with a suitable name.

Set Up Firebase SDK: In the ReactJs project, install Firebase SDK by running npm install firebase. Import and initialize Firebase in the React app by adding the Firebase configuration details obtained from the Firebase Console.

Firebase Storage Configuration: Use the Firebase SDK to configure Firebase Cloud Storage in the React app and set up rules for secure access to the storage buckets.

Firebase Admin SDK (Backend): In the ExpressJs project, install the Firebase Admin SDK by running npm install firebase-admin. Use the Admin SDK to authenticate the backend and interact with Firebase Cloud Storage, e.g., for uploading cake images.

## Frontend Implementation:

Navbar Component: Design and implement a reusable Navbar component that includes navigation links to different sections of the website, such as Home, Cake Listing, Cart, Login/Register, etc.

Footer Component: Create a Footer component to display copyright information and other relevant links.

Home Banner: Design an attractive home banner that welcomes users and highlights special offers or promotions.

Featured Cakes: Display a selection of featured cakes on the homepage, showcasing images, names, and brief descriptions.

Call to Action: Include a call-to-action button that encourages users to explore the cake listing page or check out current offers.

Fetch Cake Data: Set up API calls to retrieve cake data from the backend and display it on the cake listing page.

Pagination: Implement pagination to display a limited number of cakes per page and allow users to navigate through multiple pages.

Cake Details Display: Create a detailed cake view page that includes a larger image, cake name, description, price, and an "Add to Cart" button.

Related Cakes: Show related cakes or suggested cakes on the cake details page to encourage users to explore more products.

Cart Icon: Display a cart icon in the Navbar to show the number of items currently in the user's shopping cart.

Cart Page: Create a cart page that lists all the selected items, their quantities, and their total cost.

Add/Remove Items: Allow users to add or remove items from the cart and update the total cost accordingly.

Cart Persistence: Implement cart persistence using the browser's local storage or cookies, so the cart items are retained across sessions.

Select Payment Gateway: Choose a suitable payment gateway (e.g., Stripe, PayPal) and create an account to obtain API keys for integration.

Checkout Process: Design a secure and user-friendly checkout process that collects shipping and payment details.

Payment Handling: Set up API calls to the backend to process the payment and handle transaction success or failure.

## Backend Implementation:

Create Routes: Organize the backend routes to handle different functionalities like cake retrieval, user authentication, cart management, order processing, etc.

Cake Listing API: Set up an API endpoint to fetch a list of cakes with details like name, image URL, price, and description.

Cake Details API: Create an API endpoint to retrieve detailed information about a specific cake based on its ID.

Upload Cake Images: Implement an API endpoint to allow uploading cake images to Firebase Cloud Storage and return the image URL after a successful upload.

Fetch Cake Images: Create an API endpoint to fetch cake images from Firebase Cloud Storage and return the image URL to be used in frontend rendering.

User Registration API: Set up an API endpoint to handle user registration by receiving user details, creating a user account, and securely storing user information in the database.

User Login API: Implement an API endpoint to handle user login by validating credentials and generating an authentication token (JWT) for the user.

User Authentication Middleware: Create middleware to authenticate user requests using JWT to secure the API endpoints.

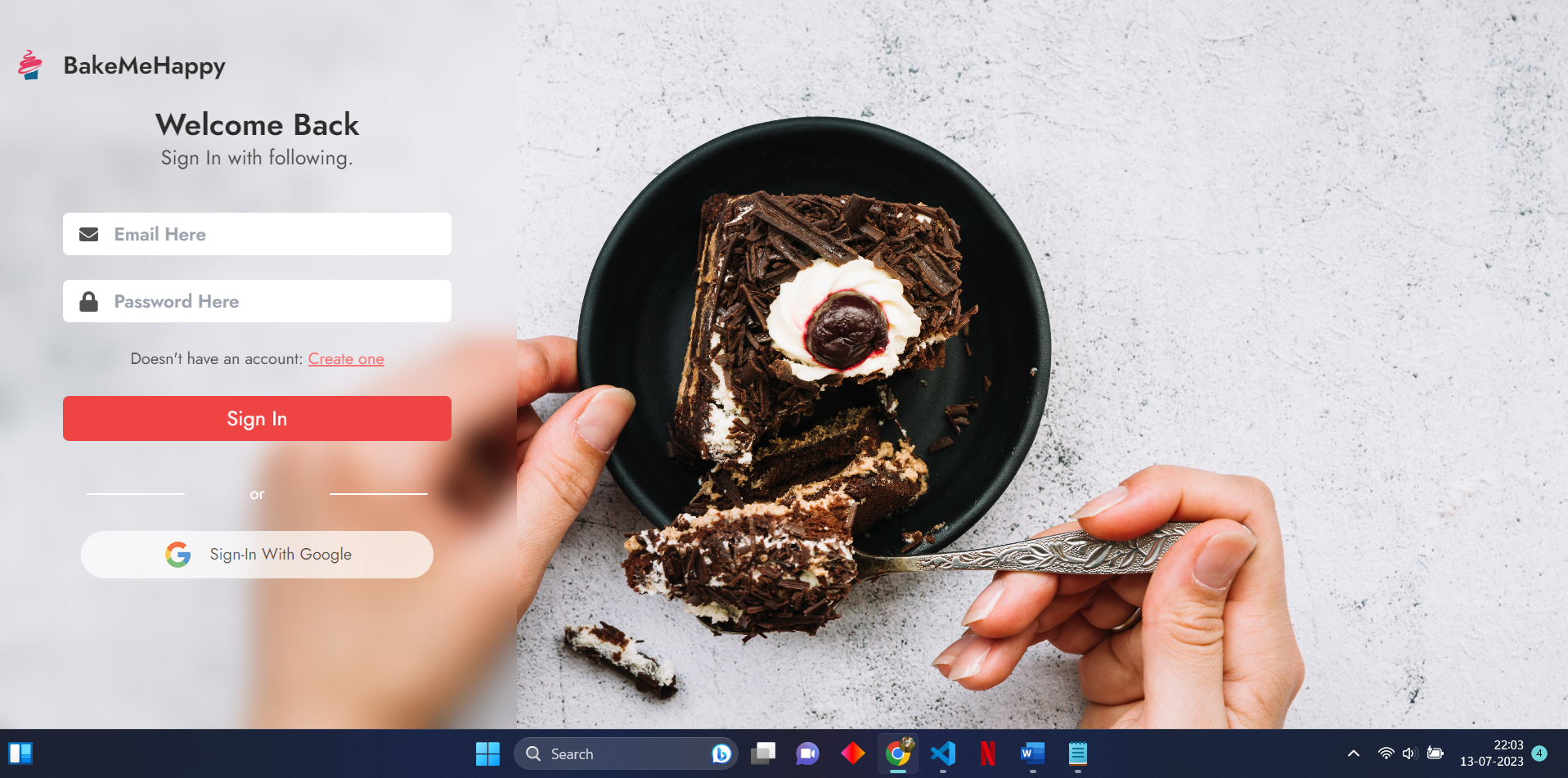
Place Order API: Design an API endpoint that allows authenticated users to place orders, capturing order details like cake items, quantities, and total cost.

Payment Processing API: Set up an API endpoint that receives payment details from the front end, processes the payment through the chosen payment gateway (e.g., Stripe), and handles payment success or failure.

Order History API: Create an API endpoint to fetch order history for authenticated users, enabling them to view their previous orders.

CORS Configuration: Set up Cross-Origin Resource Sharing (CORS) headers to restrict frontend access and prevent unauthorized cross-origin requests.

1. Conclusion:



A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Screens screenshot of a computer

Description automatically generated

1. References:

* <https://legacy.reactjs.org/docs/getting-started.html>
* <https://www.udemy.com/course/the-complete-react-fullstack-course/?utm_source=adwords&utm_medium=udemyads&utm_campaign=WebDevelopment_v.PROF_la.EN_cc.INDIA_ti.8322_Exp&utm_content=deal4584&utm_term=_._ag_82381207658_._ad_533094292062_._kw__._de_c_._dm__._pl__._ti_dsa-774930029169_._li_9299424_._pd__._&matchtype=&gclid=CjwKCAjw-b-kBhB-EiwA4fvKrAqIZwyvo5cC6QeNwL9EIfpLp_4p9fmmPr4TL1kIIoLOXAXIP3uGwhoCcDEQAvD_BwE>
* <https://firebase.google.com/docs?gad=1&gclid=CjwKCAjwwb6lBhBJEiwAbuVUSnU5Hyudl7gdj4PnTLp5a75H6QtM3rfQwrlhIWJ5VReNGCfSqsJ2MRoCdAQQAvD_BwE&gclsrc=aw.ds>
* <https://nodejs.org/en/docs>
* <https://redux.js.org/>
* <https://www.framer.com/motion/animate-function/?utm_source=google&utm_medium=adwords&utm_campaign=TW-WW-All-GS-UA-Traffic-20190326-Brand.Bmm_&gad=1&gclid=CjwKCAjwwb6lBhBJEiwAbuVUSpxSTy8P3fgtCtXOtUA49v-Rk5Ui8MPOjqs68LjdoDAypI6OM1L81xoCd3UQAvD_BwE>
* <https://mui.com/material-ui/getting-started/>