### **SYSTEM ANALYSIS , SRS , DATABASE DESIGN**

### **1.SYSTEM ANALYSIS**

### **1.1 Problem Definition**

Whether it is a special curry or a quick snack recipe passed down through friends, food lovers often want a place to share and discover such gems. But finding a platform that feels welcoming and interactive is still a challenge. Most existing recipe websites are either too commercial or focused only on professional content, leaving little room for everyday users to participate. They often lack proper features for user engagement, recipe moderation, or safe sharing. What’s really needed is a simple and friendly space where anyone can upload their favorite recipes, interact with others through likes, comments, and ratings, and enjoy a secure environment where quality content is encouraged and protected.

### **1.2 Advantages of the Proposed System**

The TasteIt platform offers:

* Centralized platform for food lovers to share and discover recipes.
* Easy recipe uploading and interactive feedback system (likes, comments, ratings).
* Bookmarking and search features for personalized use.
* Admin moderation for quality and safety.
* Promotes user engagement and food culture sharing.

**1.3 Feasibility Analysis**

**a. Technical Feasibility:**  
 The development of the TasteIt platform is technically feasible due to the availability and compatibility of the technologies involved. The system will be developed using PHP for the backend. The database will be managed using MySQL. The front-end will be built using HTML, CSS, and JavaScript to ensure a responsive and user-friendly interface. These technologies are lightweight, platform-independent, and supported on all major operating systems and browsers. Development and testing will be carried out using XAMPP which provide an integrated environment with Apache, MySQL, and PHP.

**b. Operational Feasibility:**  
 Operationally, the system is highly feasible because it aligns well with the needs of both users and administrators. The design ensures ease of use for all three roles: guest users, registered users, and admins. Guest users can conveniently browse and search for recipes, while registered users have additional features such as posting recipes, editing their content, and interacting through likes, comments, and bookmarks. Admins have simple yet powerful tools to moderate content and manage user activities. The system will also support smooth navigation and quick response times, enhancing the overall user experience.

**c. Economic Feasibility:**  
 From an economic perspective, the TasteIt project is entirely feasible and cost-effective. Since this is an academic mini-project, there is no requirement for financial investment in commercial tools or infrastructure. All necessary software components, including PHP, MySQL, and XAMPP are freely available. Development and testing can be done locally using a personal computer, avoiding any hardware upgrade costs. Optional deployment for demonstration purposes can be done using free web hosting platforms by eliminating the need for paid hosting services. The advantages it provides greatly surpass the minimal effort required for setup."

### **1.4 Recommended Implementations**

The recommended implementation of the TasteIt system involves the use of a standard web development stack that is both efficient and accessible for student-level projects. The front-end will be developed using HTML, CSS, and JavaScript to create a responsive and visually appealing user interface that supports smooth interaction across various devices. For the server-side scripting, PHP is chosen due to its compatibility with web applications and ease of integration with databases. MySQL will be used to manage and store all backend data, including user information, recipes, comments, likes, and reports. Development and testing will be carried out locally using XAMPP which provide a bundled environment of Apache server, PHP, and MySQL, eliminating the need for separate server configurations. Visual Studio Code or any modern text editor can be used for writing code efficiently. The Waterfall Model is recommended as the development methodology since it allows a clear, structured approach to project stages requirement analysis, design, development, testing, and deployment making it ideal for academic timelines. The overall system design emphasizes simplicity, usability, and functionality while using cost-effective and widely available tools.

### **1.5 Techniques Used**

The development of the TasteIt system follows the Waterfall Model, which provides a step-by-step approach through requirements gathering, design, implementation, testing, and deployment. This model is suitable for academic projects with a clear scope and timeline. For security, basic techniques such as input validation and password hashing are used to protect user data and prevent vulnerabilities like SQL injection. The design of the system ensures organized code structure, with PHP handling server-side logic and MySQL managing relational data. The interface is built using standard web technologies, ensuring a responsive and user-friendly experience.

**2. SOFTWARE REQUIREMENTS SPECIFICATION**

### **2.1 Introduction**

The Software Requirements Specification (SRS) document provides a detailed and structured description of the TasteIt recipe-sharing platform. It outlines the functional and non-functional requirements of the system, ensuring that all stakeholders such as developers, testers, users, and evaluators have a clear understanding of what the system is intended to do. This document acts as a guide throughout the development process, serving as a reference for verifying and validating the system’s features and performance. By capturing the complete requirements early, the SRS reduces the chances of scope creep and development errors later in the project lifecycle.

### **2.2 Purpose**

The purpose of the TasteIt system is to provide an interactive, user-driven web platform where individuals can share, discover, and engage with a wide variety of recipes. The system will bridge the gap between home cooks and food enthusiasts by offering a dedicated space to upload recipes, rate and comment on others’ creations, and manage their own recipe collections. It also introduces administrative controls to ensure content quality and safety. This system is intended to promote food culture, encourage culinary creativity, and simplify the way recipes are organized and accessed in a digital environment.

### **2.3 Scope**

TasteIt is a web-based application designed to facilitate the sharing and discovery of recipes through a structured and moderated environment. The system will support three roles: guest users, registered users, and admin. Guest users can view and search for recipes, while registered users can add, edit, delete, rate, comment, like, and bookmark recipes. Admins will have the authority to manage users, approve or delete recipes, and respond to reported content.

### **2.4 Technical Overview**

* **Frontend Technologies**: HTML, CSS, JavaScript
* **Backend Language**: PHP (for server-side logic)
* **Database**: MySQL (for storing recipes, users, interactions)
* **Development Environment**: XAMPP (includes Apache, PHP, MySQL)
* **Architecture**: Modular structure based on user roles (guest, registered, admin)
* **Browser Compatibility**: Chrome, Firefox,Edge

### **2.5 Stated Requirements**

#### **A . Functional Requirements**

* User registration and login system
* View, search, and filter recipes
* Registered users can:
  + Upload, edit, and delete their own recipes
  + Like, comment on, and bookmark recipes
  + Report inappropriate content
* Admin users can:
  + Approve or delete any recipe
  + Block/delete users
  + Act on reported content
  + View total users and active contributors

#### **B. Non-Functional Requirements**

* **Usability**: Clean and intuitive user interface
* **Performance**: Fast page load and smooth interactions
* **Security**: Password protection, input validation, role-based access
* **Compatibility**: Works across major browsers and devices
* **Reliability**: Stable system that functions without errors

### **2.6 External Interface Requirements**

* **User Interfaces**:
  + Registration and login forms
  + Recipe submission/edit forms
  + Recipe detail pages with like, comment, and bookmark buttons
  + Admin dashboard for managing users and content
* **Database Interfaces**:
  + PHP-MySQL interaction using SQL queries
  + CRUD operations (Create, Read, Update, Delete) for recipes, users, comments
* **System Constraints**:
  + Must run on local server (XAMPP) during development
  + Should be accessible on standard web browsers (Chrome, Firefox, etc.)
  + Responsive design to support both mobile and desktop views

## **3. DATABASE DESIGN**

### **3.1 ER Diagram -Entities and Attributes**

#### **User**

Represents all users of the system including guests, registered users, and admins.

* **user\_id** (Primary Key) – Unique identifier for each user
* **name** – Full name of the user
* **email** – Email address (used for login)
* **password** – Hashed password
* **role** – Defines user role:
  + guest *(not stored in DB, only browsing)*
  + registered *(can post, comment, like, etc.)*
  + admin *(can moderate and manage content/users)*
* **profile\_image** – Optional user profile picture
* **created\_at** – Account creation date

#### **Recipe**

Stores all user-submitted recipes.

* **recipe\_id** (Primary Key) – Unique recipe identifier
* **user\_id** (Foreign Key) – References User.user\_id
* **title** – Name of the recipe
* **ingredients** – Text list of ingredients
* **instructions** – Preparation steps
* **category** – Type (e.g., Dessert, Breakfast, Vegetarian)
* **image** – Optional recipe photo
* **created\_at** – Date of upload
* **approved** – Boolean (Yes/No) – Whether admin has approved the recipe

#### **Comment**

Stores user comments on recipes.

* **comment\_id** (Primary Key) – Unique comment identifier
* **recipe\_id** (Foreign Key) – References Recipe.recipe\_id
* **user\_id** (Foreign Key) – References User.user\_id
* **comment\_text** – The actual comment content
* **comment\_date** – Date and time of comment

#### **Like**

Tracks recipe likes from registered users.

* **user\_id** (Foreign Key) – References User.user\_id
* **recipe\_id** (Foreign Key) – References Recipe.recipe\_id
* **PRIMARY KEY** – Composite key of user\_id and recipe\_id

#### **Bookmark**

Tracks bookmarked recipes.

* **user\_id** (Foreign Key) – References User.user\_id
* **recipe\_id** (Foreign Key) – References Recipe.recipe\_id
* **PRIMARY KEY** – Composite key of user\_id and recipe\_id

#### **Report**

Tracks reports of inappropriate recipes made by users.

* **report\_id** (Primary Key) – Unique report identifier
* **user\_id** (Foreign Key) – Who reported
* **recipe\_id** (Foreign Key) – Which recipe was reported
* **reason** – Short description or reason for reporting
* **status** – e.g., "Pending", "Reviewed", "Action Taken"
* **report\_date** – Date and time of report

**Relationships**:

* User uploads Recipes (1:M)
* User comments on Recipe (1:M)
* User likes/ bookmarks Recipe (M:M via Like/Bookmark tables)
* User reports Recipe (1:M)

### **3.2 Relational Schema Table**

|  |  |
| --- | --- |
| **TableName** | **Columns & Data Types** |
| **User** | user\_id (INT PRIMARY KEY)  name ( VARCHAR(100) NOT NULL)  email (VARCHAR(100) UNIQUE NOT NULL)  password (VARCHAR(255) NOT NULL)  role (ENUM('registered', 'admin') DEFAULT 'registered' ) profile\_image (VARCHAR(255) DEFAULT NULL ) created\_at (DATETIME DEFAULT CURRENT\_TIMESTAMP) |
| **Recipe** | recipe\_id (INT PRIMARY KEY AUTO\_INCREMENT ) user\_id (INT NOT NULL, FOREIGN KEY (user\_id) REFERENCES User(user\_id) ) title ( VARCHAR(150) NOT NULL)  ingredients (TEXT NOT NULL ) instructions (TEXT NOT NULL)  category (VARCHAR(50) ) image (VARCHAR(255) DEFAULT NULL)  createdAt ( DATETIME DEFAULT CURRENT\_TIMESTAMP ) approved (BOOLEAN DEFAULT FALSE) |
| **Comment** | comment\_id (INT PRIMARY KEY AUTO\_INCREMENT)  recipe\_id (INT NOT NULL, FOREIGN KEY (recipe\_id) REFERENCES Recipe(recipe\_id) ) user\_id (INT NOT NULL, FOREIGN KEY (user\_id) REFERENCES User(user\_id) ) comment\_text ( TEXT NOT NULL)  comment\_date (DATETIME DEFAULT CURRENT\_TIMESTAMP) |
| **Like** | user\_id (INT NOT NULL, FOREIGN KEY (user\_id) REFERENCES User(user\_id) ) recipe\_id (INT NOT NULL, FOREIGN KEY (recipe\_id) REFERENCES Recipe(recipe\_id) ) PRIMARY KEY (user\_id, recipe\_id) |
| **Bookmark** | user\_id (INT NOT NULL, FOREIGN KEY (user\_id) REFERENCES User(user\_id) ) recipe\_id (INT NOT NULL, FOREIGN KEY (recipe\_id) REFERENCES Recipe(recipe\_id) ) PRIMARY KEY (user\_id, recipe\_id) |
| **Report** | report\_id (INT PRIMARY KEY AUTO\_INCREMENT)  user\_id ( INT NOT NULL, FOREIGN KEY (user\_id) REFERENCES User(user\_id) ) recipe\_id (INT NOT NULL, FOREIGN KEY (recipe\_id) REFERENCES Recipe(recipe\_id) ) reason (VARCHAR(255) NOT NULL)  status (ENUM('Pending', 'Reviewed', 'Action Taken') DEFAULT 'Pending' ) report\_date (DATETIME DEFAULT CURRENT\_TIMESTAMP) |

### **Normalization**

All tables in the **TasteIt** database are normalized to **Third Normal Form (3NF)**:

* **1NF**: Each column contains atomic values, such as a single name, email, or ingredient list (as text), with no repeating groups in any table.
* **2NF**: Every non-key attribute in each table is fully functionally dependent on the whole primary key. For example, in the Recipe table, all attributes like title, instructions, and category depend entirely on recipe\_id.
* **3NF**: There are no transitive dependencies between non-key attributes. For instance, in the User table, attributes like name, email, and role are only dependent on the primary key user\_id and not on each other.

This normalization ensures minimal redundancy, improved data integrity, and optimal query performance throughout the system.