### TABLE OF CONTENTS

[TABLE OF CONTENTS 1](#_Toc133967868)

[1. SYNOPSIS 1](#_Toc133967869)

[1.1. Title of the Project 2](#_Toc133967870)

[1.2. Abstract 2](#_Toc133967871)

[1.3. Objective of the Project 2](#_Toc133967872)

[1.4. Project Category 3](#_Toc133967873)

[1.5. Language(s) to be used 3](#_Toc133967874)

[1.5.1. Frontend 3](#_Toc133967875)

[1.5.2. Backend 3](#_Toc133967876)

[1.5.3. Database : 3](#_Toc133967877)

[1.6. Structure of the proposed project 3](#_Toc133967878)

[1.7. Module Description 3](#_Toc133967879)

[1.7.1. User Management Module: 3](#_Toc133967880)

[1.7.2. Restaurant Discovery Module: 4](#_Toc133967881)

[1.7.3. User Review Module: 4](#_Toc133967882)

[1.7.4. Favorites Management Module: 4](#_Toc133967883)

[1.7.5. Drops Module: 4](#_Toc133967884)

[1.7.6. Price Filtering Module: 4](#_Toc133967885)

[1.7.7. Admin Panel Module: 4](#_Toc133967886)

[1.8. Future scope of the Project 5](#_Toc133967887)

[1.9. Bibliography 5](#_Toc133967888)

[2.1. Introduction 7](#_Toc133967889)

[2.1.1. Purpose 7](#_Toc133967890)

[2.1.2. Intended Audience and Reading Suggestions 7](#_Toc133967891)

[2.1.3. Product Scope 8](#_Toc133967892)

[2.1.4. References 8](#_Toc133967893)

[2.2. Overall Description 9](#_Toc133967894)

[2.2.1. Product Perspective 9](#_Toc133967895)

[2.2.2. Product Functions 9](#_Toc133967896)

[2.2.3. User Classes and Characteristics 10](#_Toc133967897)

[2.2.4. Operating Environment 10](#_Toc133967898)

[2.2.5. Assumptions and Dependencies 11](#_Toc133967899)

[2.3. External Interface Requirements 11](#_Toc133967900)

[2.3.1. User Interfaces 11](#_Toc133967901)

[2.3.2. Hardware Interfaces 11](#_Toc133967902)

[2.3.3. Software Interfaces 12](#_Toc133967903)

[2.3.4. Communications Interfaces 12](#_Toc133967904)

[2.4. System Features 13](#_Toc133967905)

[2.4.1. User Registration and Authentication 13](#_Toc133967906)

[2.4.2. Geolocation: 13](#_Toc133967907)

[2.4.3. Restaurant information 14](#_Toc133967908)

[2.4.4. Reviews: 14](#_Toc133967909)

[2.4.5. Favorites Management: 14](#_Toc133967910)

[2.4.6. Drops (Food Vlog Upload): 15](#_Toc133967911)

[2.4.7. Price Filtering: 15](#_Toc133967912)

[2.4.8. Restaurant Management (Admin Panel): 15](#_Toc133967913)

[2.5. Other Nonfunctional Requirements 16](#_Toc133967914)

[2.5.1. Performance Requirements 16](#_Toc133967915)

[2.5.2. Security Requirements 16](#_Toc133967916)

[2.5.3. Software Quality Attributes 16](#_Toc133967917)

[2.6.1. LEVEL 0 DFD OR (CONTEXT FLOW DIAGRAM) 18](#_Toc133967918)

[2.6.2. LEVEL 1 DFD (ADMIN) 18](#_Toc133967919)

[2.6.3. LEVEL 1 DFD (USER) 19](#_Toc133967920)

[**3.1** TABLE DESCRIPTION 8](#_Toc133967921)

[**3.2** NORMALIZATION 9](#_Toc133967922)

[. 2](#_Toc133967923)

[4.1 TESTING AND VALIDATION 2](#_Toc133967924)

[**4.1** INTRODUCTION 3](#_Toc133967925)

[**4.2** TESTING OBJECTIVES 3](#_Toc133967926)

[**4.3** TESTING METHODS 3](#_Toc133967927)

[**4.4** TESTING STEPS 5](#_Toc133967928)

### 

### SYNOPSIS

### Title of the Project

### 

Eatables

### Abstract

EATABLES is a location-based food discovery platform that allows users to explore and find new dining options. Users can access information about nearby restaurants, including contact details, location, and reviews from other users. They can leave their own reviews, add food items to favorites, and upload food vlog videos through the "drops" feature. The platform offers filtered search results based on price and allows users to submit new restaurant additions for review by the admin. EATABLES aims to provide a comprehensive and engaging experience for users to discover, share, and review food options in their area.

### Objective of the Project

The objective of the EATABLES project is to provide users with a convenient and comprehensive platform to discover new food options based on their location. The project aims to fulfill the following objectives:

* Location-based Food Discovery: Enable users to find nearby restaurants, view their information, and read reviews from other users, helping them make informed dining choices.
* User Reviews and Engagement: Allow users to leave their own reviews, fostering a community-driven approach and facilitating knowledge sharing about dining experiences.
* Favorites List: Provide users with the ability to create a personalized list of favorite food items or restaurants for easy reference and future visits.
* Drops Feature: Allow users to upload and share food vlog videos, enhancing engagement and providing a platform for users to discover and appreciate food-related content
* Price Filtering: Offer users the option to filter search results based on price, ensuring that they can find dining options that align with their budget or desired price range.
* Restaurant Submission: Enable users to suggest new restaurants to be added to the platform, ensuring the continuous growth and relevance of the database.

Overall, the objective of the EATABLES project is to enhance the dining experience for users by providing them with a user-friendly platform that leverages location data, user reviews, engagement features, and personalized preferences to facilitate food discovery and decision-making.

### Project Category

Web-based Application

### Language(s) to be used

* + 1. Frontend : HTML, CSS & JavaScript
    2. Backend : PHP
    3. Database : MySQL

### Structure of the proposed project

* User Management: This component enables user authentication and registration, allowing users to create accounts and log in to access the system.
* Location-Based Restaurant Discovery: Using the device's location services, this component identifies nearby restaurants and provides users with relevant information, such as contact details and location.
* User Reviews: Users can leave reviews for restaurants they have visited, sharing their experiences and opinions to help other users make informed dining decisions.
* Favorites Management: Users can create a personalized list of favorite food items or restaurants for easy access and future reference.
* Drops: The Drops feature allows users to upload and share food vlog videos, fostering engagement and enabling users to discover and appreciate food-related content.
* Price Filtering: Users can filter search results based on their desired price range, aiding in finding dining options that match their budget.
* Restaurant Management (Admin Panel): This component grants administrative privileges and allows the admin to manage restaurants. It includes functionalities such as adding new restaurants, editing existing ones, and reviewing user requests for adding new restaurants.

### Module Description

### User Management Module:

This module handles user authentication and registration. It allows

users to create accounts, log in securely, and manage their profiles.

### Restaurant Discovery Module:

The Restaurant Discovery module utilizes the device's location services to identify nearby restaurants. It provides users with essential information such as contact details, location, and user reviews.

### User Review Module:

The User Review module enables users to leave reviews for restaurants they have visited. Users can share their dining experiences, providing valuable insights for other users.

### Favorites Management Module:

With the Favorites Management module, users can create a personalized list of their favorite food items or restaurants. This module allows users to easily access and organize their preferred options.

### Drops Module:

The Drops module is designed for users to upload and share food vlog videos. Users can showcase their culinary experiences and engage with other users' content**.**

### Price Filtering Module:

The Price Filtering module enables users to filter search results based on their desired price range. This module helps users find dining options that fit their budget.

### Admin Panel Module:

The Admin Panel module is accessible only to the admin. It allows the admin to manage restaurants, including adding new restaurants, editing existing ones, and reviewing user requests for adding new restaurants.

### Future scope of the Project

* Integration of Advanced Recommendation Systems: The platform can incorporate advanced recommendation algorithms to provide personalized food recommendations to users based on their preferences, previous reviews, and browsing history. This can greatly enhance the user experience and help users discover new food options tailored to their individual tastes.
* Social Media Integration and Influencer Collaborations: The project can explore integration with popular social media platforms, enabling users to share their dining experiences and reviews on their social media accounts directly from the EATABLES platform. Collaborations with food influencers or bloggers can also be considered to enhance the visibility and engagement of the platform.
* Enhanced User Engagement and Gamification: Implementing gamification elements such as badges, rewards, and challenges can boost user engagement and create a more interactive experience. Users can earn rewards for leaving reviews, participating in community activities, or achieving specific milestones within the platform.
* Expansion to Additional Geographical Locations: While initially focusing on a specific location, the project can expand its reach to cover more cities or even different countries, allowing users to discover food options in various locations.
* Integration with Third-Party Services: The platform can explore partnerships and integrations with third-party services such as reservation systems, food delivery aggregators, or restaurant booking platforms. This would offer users a comprehensive solution for discovering, reviewing, and accessing various restaurant services through a single platform

### Bibliography

* LECTURE NOTES ON SOFTWARE ENGINEERING

By Dr. H.S.Behera Asst. Prof K.K.Sahu Asst. Prof Gargi Bhattacharjee

* Fundamentals of Software Engineering, PHI

By Mall Rajib

1. **SOFTWARE REQUIREMENTS AND SPECIFICATION**

### Introduction

The Software Requirements Specification (SRS) document is a crucial artifact in the software development process. It serves as a comprehensive guide that

captures the functional and non-functional requirements of a software system. The SRS document outlines the purpose, scope, and architecture of the software, providing a clear understanding of what needs to be developed and how it should behave. It serves as a reference for stakeholders, including developers, designers, testers, and clients, ensuring a common understanding of the project goals and

facilitating effective communication and collaboration throughout the software

development lifecycle. The SRS document acts as a contract between the

development team and the stakeholders, guiding the development process and serving as a benchmark for validation and verification of the final product.

### Purpose

The purpose of the project is to develop a comprehensive and user-friendly food discovery platform that allows users to explore, review, and interact with restaurants in their vicinity. The project aims to simplify the process of finding new dining options, providing users with valuable information, user reviews, and the ability to save favorites. By creating a platform that enhances the dining experience and facilitates informed decision-making, the project seeks to improve user satisfaction and promote the culinary exploration of different food options.

### Intended Audience and Reading Suggestions

* Project team members: Students who are working on the project, including

developers, designers, and testers.

* Internal guide: The SRS will be reviewed and evaluated by the internal guide who is overseeing the project.
* Peers: Other students in the class may also review the SRS to provide feedback and suggestions.
* External guide: They will want to review the document to ensure that the product meets their needs and expectations.

Reading suggestions for the SRS document could include the following:

* Review the document in its entirety to get a general understanding of the project and its requirements.
* Focus on specific sections that are relevant to readers role on the project, such as the functional requirements or non-functional requirements.
* Review the document periodically throughout the project to ensure that the requirements are being met and to make any necessary updates.

### Product Scope

The platform aims to provide users with a convenient and user-friendly interface to discover nearby restaurants, view essential information such as contact details and location, and read reviews from other users. Users can also leave their own reviews, add food items or restaurants to their favorites list, and upload food vlog videos through the Drops feature. Additionally, the project includes an admin panel for restaurant management, allowing the admin to add, edit, or delete restaurants,

manage menu items, and review user requests for adding new restaurants.

The project focuses on delivering the core functionalities mentioned while ensuring a seamless user experience and providing a comprehensive solution for food

discovery and engagement.

### References

* IEEE Software Requirements Specification Template:

https://web.cs.dal.ca/~hawkey/3130/srs\_template-ieee.doc

* How to Write a Software Requirements Specification (SRS Document):

https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document

* 6 Steps for Writing an SRS That Works:

https://www.uptech.team/blog/srs-document

### Overall Description

### Product Perspective

The EATABLES food discovery platform is designed to operate as a standalone system, independent of any other software or external systems. It serves as a

centralized platform for users to explore and engage with restaurants in their

vicinity. While the platform interacts with external services such as location

services for restaurant discovery, it does not rely on or integrate with any

specific external systems for its core functionality.

The platform is developed as a web-based application accessible through web browsers on various devices, including desktops, laptops, tablets, and smartphones. It does not require any specific hardware components or dependencies beyond standard web-browsing capabilities.

As an independent system, EATABLES aims to provide a comprehensive solution for users to discover, review, and interact with restaurants. It does not seek to

replace or integrate with existing restaurant management systems or third-party

services. However, future integrations with external systems such as online ordering and delivery services can be considered to enhance the user experience and expand the platform's capabilities

### Product Functions

* User Registration and Authentication: Users can create accounts and

securely log in to access the platform's

* Restaurant Discovery: The platform utilizes location services to identify nearby restaurants and provides users with essential information, such as contact details, location, and reviews from other users.
* User Reviews: Users can leave detailed reviews for restaurants they have visited, sharing their experiences and opinions to assist other users in

making informed dining decisions.

* Favorites Management: Users can create a personalized list of favorite food items or restaurants, allowing for quick and easy access to their

preferred options.

* Drops (Food Vlog Upload): Users have the ability to upload and share food vlog videos, creating an engaging platform for users to showcase their

culinary experiences and interact with others.

* Price Filtering: Users can filter search results based on their desired price range, helping them find dining options that align with their budget or

affordability preferences.

* Admin Panel: The admin panel provides administrative privileges, allowing the admin to manage restaurants, add new restaurants, edit existing ones, and review user requests for adding new restaurants.

### User Classes and Characteristics

* General Users: These are individuals who use the platform to discover and explore restaurants in their vicinity. They may have varying levels of

technological expertise and preferences when it comes to dining

experiences. General users rely on the platform to provide accurate and

relevant restaurant information, user reviews, and an intuitive interface for seamless navigation.

* Administrators: Administrators have elevated privileges and are responsible for overseeing the overall functioning of the platform. They have access to the admin panel to review user requests for adding new restaurants,

moderate content, and maintain the integrity of the platform. Administrators possess advanced system management skills and a comprehensive

understanding of the platform's functionalities.

### Operating Environment

* Hardware platform: the system should be accessible on both desktop and mobile devices, including laptops, smartphones, and tablets.
* Operating system: the system should be compatible with popular operating systems such as Windows, mac OS, iOS, and Android.
* Browser compatibility: the system should be accessible via modern web browsers, such as Google Chrome, Mozilla Firefox, Apple Safari, and

Microsoft Edge.

### Assumptions and Dependencies

* Location data accuracy: The system relies on accurate location data to

provide relevant results and recommendations. If the location data is not accurate, the system may provide incorrect results, affecting the user

experience.

* User reviews: The system is based on user reviews, which may not always be accurate or trustworthy. There is a risk that fake reviews could be posted, affecting the accuracy of the information provided by the system.
* Dependency on network connectivity: The system may rely on network connectivity to retrieve data from the server and provide results to the user. If network connectivity is poor or unavailable, the system may not work properly.

### External Interface Requirements

### User Interfaces

The interface should be easy to navigate, with clear and consistent labeling and a logical hierarchy of information.

The system should display ratings and reviews for each restaurant, with the ability for users to leave their own reviews and ratings. The review system should be easy to use and understand, with clear guidelines for leaving reviews and feedback.

The system should include maps to help users find the restaurants they are interested in. Users should be able to view maps with clear markers for each restaurant.

The system should be designed to be mobile responsive, with a layout that adapts to different screen sizes and resolutions. This will ensure that users can access your system from any device, whether they are using a desktop computer, tablet, or smartphone.

### Hardware Interfaces

Supported device types: The Eatables platform should support a range of device types, including desktop and mobile devices. This will involve optimizing the UI for different screen sizes and orientations, as well as ensuring compatibility with different operating systems (e.g., Windows, macOS, iOS, Android).

### Software Interfaces

* Databases: The Eatables platform will require a database to store and

retrieve user data, such as reviews, ratings, and restaurant information. The database could be a SQL or NoSQL database, such as MySQL, MongoDB, or Cassandra, depending on the specific requirements of the platform.

* Operating systems: The Eatables platform should be compatible with a range of operating systems, including Windows, macOS, iOS, and Android.
* Data items and messages: The data items and messages coming into the

system will primarily include user-generated data, such as reviews,

ratings,as well as location data from the user's device. Data items and

messages going out of the system will primarily include restaurant

information, recommendations, and search results.

* Services needed: The Eatables platform will require a range of services to support its functionality, such as geolocation services, data storage and

retrieval services, and user authentication services.

* Shared data: Data that will be shared across software components will

primarily include user data, such as reviews, ratings, as well as restaurant information. The data sharing mechanism will likely involve database

queries and API requests and responses.

### Communications Interfaces

* Web browser: The Eatables platform will primarily be accessed through a web browser on desktop and mobile devices. It should support modern browsers such as Google Chrome, Mozilla Firefox, and Safari.
* Electronic forms: The Eatables platform may use electronic forms for user authentication, such as sign up and login, and for collecting user-generated data, such as reviews and ratings.
* Communication standards: The Eatables platform may use HTTP or HTTPS for communication between the client and server. It may also use the

Geolocation API for retrieving location data from the user's device.

### System Features

### User Registration and Authentication

Description:

This feature allows users to create an account and login securely to access the system.

Stimulus/Response Sequence:

User clicks on the "Sign Up" button on the login page

System prompts the user to enter their name, email address, and password

User fills in the details and clicks on the "Submit" button

System verifies the information and creates the user account

User enters their credentials and clicks on the "Login" button

System authenticates the user and grants access to the platform

Functional Requirements:

User account creation

Secure login and authentication process

### Geolocation:

Description:

This feature uses the device's location to provide users with a list of nearby restaurants. When a user opens the system, it should request permission to access the user's location. If the user grants permission, the system should display a list of restaurants near the user's current location.

Stimulus/Response Sequence:

User opens the app -> App requests permission to access the user's location -> User grants permission -> App displays a list of nearby restaurants.

Functional Requirements:

The system should use an accurate and reliable geolocation service to determine the user's location.

### Restaurant information

Description:

This feature displays detailed information about each restaurant, including its name, address, contact information, menu items, and reviews.

Stimulus/Response Sequence:

User selects a restaurant -> App displays detailed information about the restaurant.

Functional Requirements:

Location-based restaurant identification

Display of essential restaurant information

Filtering options based on price

### Reviews:

Description:

This feature allows users to leave reviews for each food items and read reviews posted by others.

Stimulus/Response Sequence:

User selects a restaurant -> User selects an item -->User leaves a review

--> App displays the review.

Functional Requirements:

The system should have a clear and easy-to-use rating and review system.

### Favorites Management:

Description:

This feature allows users to create a personalized list of favorite food items or restaurants and quickly access them for future reference.

Stimulus/Response Sequence:

User selects a restaurant -> User selects an item --> user add item to their favorites list

User can access their list of favorites from their account page

Functional Requirements:

Option for users to create a personalized list of favorite food items

Quick access to saved favorites for future reference

### Drops (Food Vlog Upload):

Description:

This feature allows users to upload and share their food vlog videos and interact with other users' videos.

Stimulus/Response Sequence:

User creates a food vlog video -->User uploads the video to the system -->

System displays the video on the Drops page for other users to view and interact with.

Other users can like, comment, and share the video

Functional Requirements:

Users can upload and share their food vlog videos

Like and comment functionalities on uploaded videos

### Price Filtering:

Description:

This feature allows users to filter search results based on their desired price range.

Stimulus/Response Sequence:

User selects a price range filter option -->System retrieves and displays the list of restaurants that fall within the selected price range

Functional Requirements:

Filtering search results based on desired price range

Helping users find dining options within their budget

### Restaurant Management (Admin Panel):

Description:

This feature allows the admin to manage restaurants and their information, including adding new restaurants to the platform, editing or deleting existing restaurant details, and reviewing and approving user requests for adding new restaurants.

Stimulus/Response Sequence:

Admin logs into the admin panel using their credentials-->

System authenticates the admin and grants access to the

admin functionalities-->Admin manages the restaurant

Functional Requirements:

Admin login and authentication

Adding new restaurants to the platform

Editing and updating existing restaurant details

Deleting restaurants from the platform

Reviewing and approving user requests for adding new restaurants

### Other Nonfunctional Requirements

### Performance Requirements

Speed: The app should be fast and responsive, with pages that load quickly and smoothly. Users expect web pages to load within a few seconds.

### Security Requirements

The web application should validate user input to prevent SQL injection attacks and cross-site scripting (XSS) attacks.

### Software Quality Attributes

Reliability: the app should be available and functional at all times, with minimal downtime or errors. It should also be able to recover quickly in the event of a failure or disruption.

* 1. **DATA FLOW DIAGRAM**

A Data Flow Diagram is the graphical representation of the data through an information system, modelling its process aspects. A DFD also known as bubble chart or dataflow graph are commonly used during problem analysis. DFD’S are very useful in understanding a system and can be efficiently used during analysis. A DFD will show what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored it does not show information about process timing or whether process will

operate in sequence or in parallel format, a DFD is often used as the preliminary step to create an overview of the system without going into great detail, which can later be elaborated, it can also be used for the visualization of data processing DFD shows the movement of data through different transformations or process in the system.

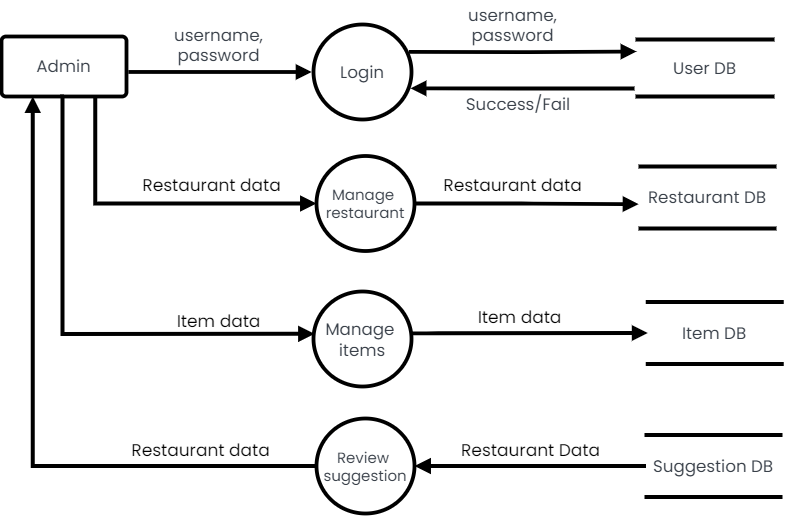
|  |  |
| --- | --- |
| **DIAGRAM** | **DESCRIPTION** |
|  | Represents source or destination of data |
|  | Represents a process that transforms incoming data into outgoing flows |
|  | Represents data flow |
|  | Represents data stores |

A Data Flow Diagram (DFD) illustrates how data is processed by a system in terms of inputs and outputs. As its name indicates its focus is on the flow of information, where it goes and how it gets stored. DFD provide critical insights into the systems and way the information passes through it. DFD helps structure every element of the system to keep them logically intact and interconnected. On the other hand, you have the customers who need to know what is going on in a digestible easy to follow manner.

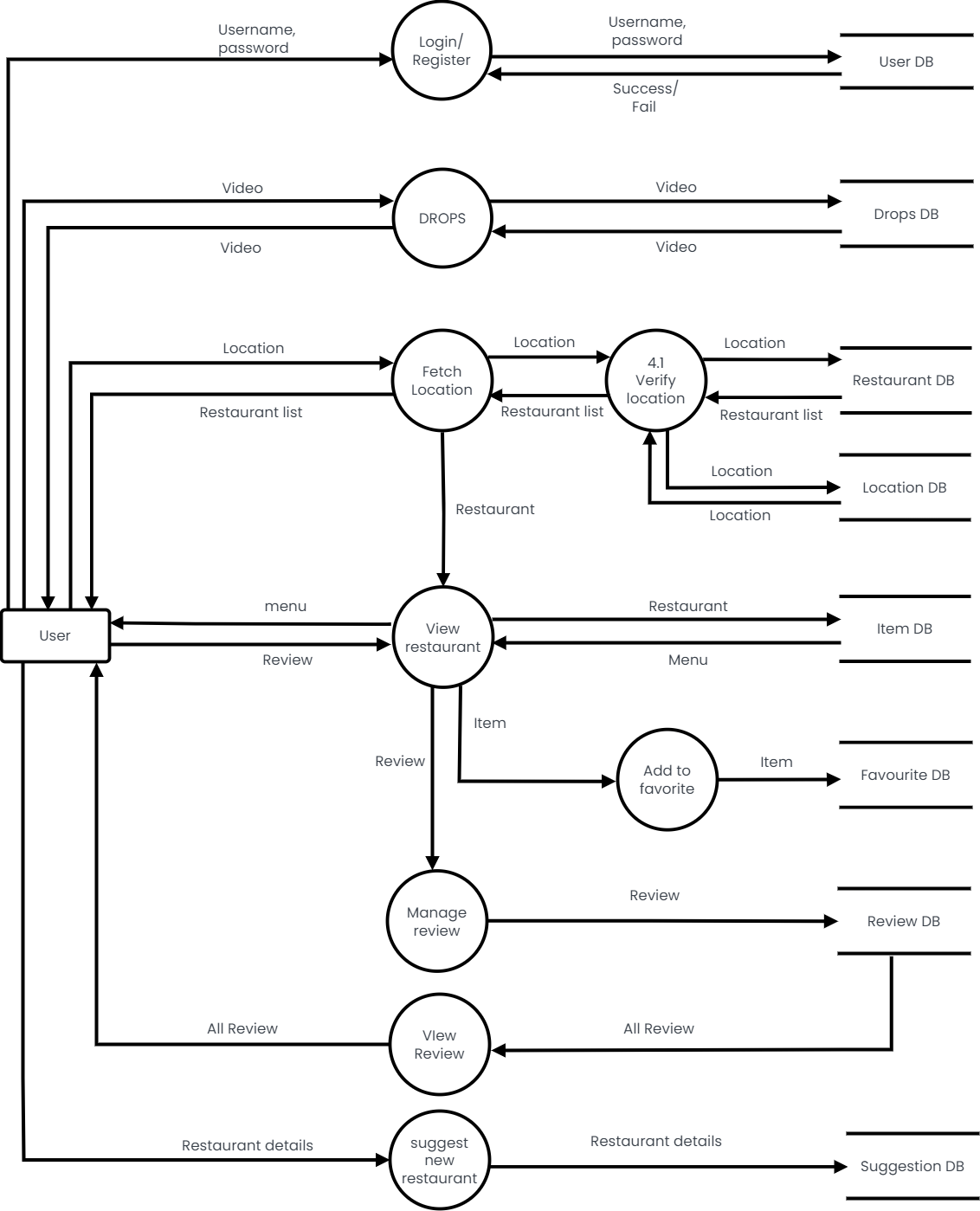
### LEVEL 0 DFD OR (CONTEXT FLOW DIAGRAM)



### LEVEL 1 DFD (ADMIN)



### LEVEL 1 DFD (USER)



System Modelling

Class diagram

* 1. **Class Diagram**

A Class diagram in the Unified Modelling Language is a type of static structure that describes the structure of a system by showing the system’s classes, their attributes, operations and the relationships among the objects. The main purpose of class diagrams is to build a static view of an application. It is the only diagram that is widely used for construction, and it can be mapped with object-oriented languages. A class diagram is used to visualize, describe, document various different aspects of the system, and also construct executable software code.

A class notation consists of three parts:

* Class Name: The name of the class appears in the first partition.
* Class Attributes: Attributes are shown in the second partition.

The attribute type is shown after the colon. Attributes map onto member variables (data members) in code.

* Class Operations (Methods): Operations are shown in the third partition.

**Visibility**

To specify the visibility of a class member (i.e. any attribute or method), these notations must be placed before the member's name:

+ Public

- Private

# Protected

~ Package

**Class Relationships**

A class may be involved in one or more relationships with other classes.

**Inheritance (or Generalization):**

* + Represents an "is-a" relationship.
  + An abstract class name is shown in italics.
  + SubClass1 and SubClass2 are specializations of Super Class.
  + A solid line with a hollow arrowhead that point from the child to the parent class.

**Aggregation:**

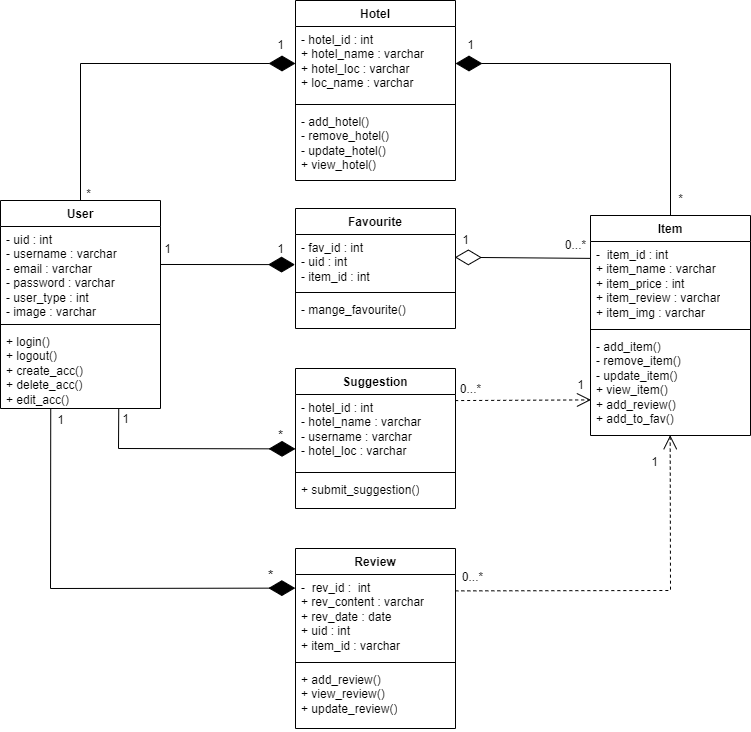
* + A special type of association. It represents a "part of" relationship.
  + Class2 is part of Class1.
  + Many instances (denoted by the \*) of Class2 can be associated with Class1.
  + Objects of Class1 and Class2 have separate lifetimes.
  + A solid line with an unfilled diamond at the association end connected to the class of composite.

**Composition:**

* + A special type of aggregation where parts are destroyed when the whole is destroyed.
  + Objects of Class2 live and die with Class1.
  + Class2 cannot stand by itself.
  + A solid line with a filled diamond at the association connected to the class of composite

**Dependency:**

* + Exists between two classes if the changes to the definition of one may cause changes to the other (but not the other way around).
  + Class1 depends on Class2
  + A dashed line with an open arrow



(Fig 3.1)

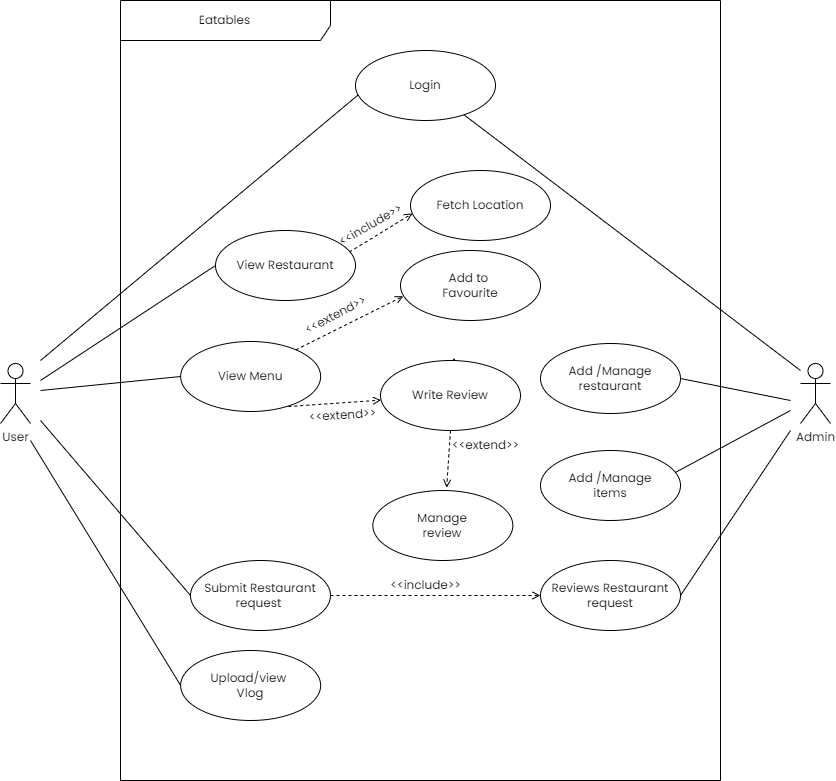
* 1. **USECASE**

A use case diagram is a graphic depiction of the interactions among the elements of a system. It is a methodology used in system analysis to identify, clarify and organize system requirements. A use case diagram does not show the detail of the use case, it only summarizes some of the relationships between use case and the actors. In particular, the diagram does not show the order in which steps are performed to achieve the goals of each use case. It deals only with the functional requirement of the system. Use case diagrams are usually referred to as behaviour diagrams, which is used to describe set of actions that systems should or can perform in collaboration with one or more external users of the system ie Actors. Each use case should provide one or more observable and valuable result to the actor.

The purpose of use case diagram is to capture core functionalities of a system and visualize the interactions of various things called as actors with the use case. A use case consists of use cases, persons, or various things that are invoking the features called as actors and the elements that are responsible for implementing the use cases. Use case diagrams capture the dynamic behaviour of live system. Use case diagrams are responsible for visualizing the external things that interact with part of the system.

A use case diagram should be simple as possible. A use case diagram should be complete. A use case diagram should represent all interactions with use case. If there are too many use cases or actors, only the essential use cases should be represented. A use case diagram should describe at least a single module of a system. If the use case diagram is large then it should be generalized. The name of the actor and use case should be meaningful and relevant to the system. Interaction of an actor with the use case must be defined clearly and in an understandable way. Annotation must be used where ever they are required. If a use case or an actor has multiple relationship, then only significant interactions must be display

|  |  |
| --- | --- |
| **DIAGRAM** | **DESCRIPTION** |
| Use Case | A use case represents a user goal that can be achieved by accessing the system or software application. In Visual Paradigm, you can make use of the sub-diagram feature to describe the interaction between user and system within a use case by creating a sub-sequence diagram under a use case. |
| Association | Actor and use case can be associated to indicate that the actor participates in that use case. Therefore, an association correspond to a sequence of actions between the actor and use case in achieving the use case. |
| Actor | Actors are the entities that interact with a system. Although in most cases, actors are used to represent the users of system, actors can actually be anything that needs to exchange information with the system. So, an actor may be people, computer hardware, other systems, etc. |
| Include | An include relationship specifies how the behaviour for the inclusion use case is inserted into the behaviour defined for the base use case. |
| Extend | An extend relationship specifies how the behaviour of the extension use case can be inserted into the behaviour defined for the base use case. |



(Fig 3.2)

* 1. **SEQUENCE DIAGRAM**

UML sequence diagram model is the flow of logic within the system in a visual manner, enabling both to document and validate the logic and are commonly used for both analysis and design purposes. It is the most popular UML artifact for dynamic modelling which focus on identifying the behaviour within the system.

Sequence diagram are typically associated with use case realizations in the logical view of the system under development. Sequence diagrams are sometime called as event diagrams and event scenarios.

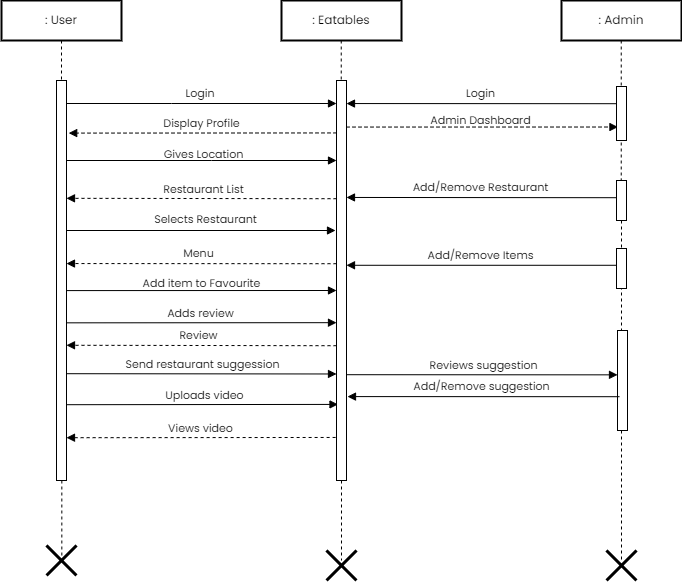
A sequence diagram shows, as parallel vertical lines different processes or the objects that live simultaneously and as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

A sequence diagram shows object interactions arranged in time sequence, it depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.



|  |  |
| --- | --- |
| **DIAGRAM** | **DESCRIPTION** |
| Object symbol | Represents a class or object in UML. The object symbol demonstrates how an object will behave in the context of the system. Class attributes should not be listed in this shape. |
| Activation box | Represents the time needed for an object to complete a task. The longer the task will take, the longer the activation box becomes. |
| Actor symbol | Shows entities that interact with or are external to the system. |
| Lifeline symbol | Represents the passage of time as it extends downward. This dashed vertical line shows the sequential events that occur to an object during the charted process. Lifelines may begin with a labelled rectangle shape or an actor symbol. |

|  |  |
| --- | --- |
| Synchronous message symbol | Represented by a solid line with a solid arrowhead. This symbol is used when a sender must wait for a response to a message before it continues. The diagram should show both  the call and the reply. |
| Asynchronous message symbol | Represented by a solid line with a lined arrowhead. Asynchronous messages don't require a response before the sender continues. Only the call should be included in the diagram. |
| Reply message symbol | Represented by a dashed line with a lined arrowhead, these messages are replies to calls. |
| Delete  message  symbol | Represented by a solid line with a solid arrowhead, followed by an X. This message destroys an object. |



DATABASE MODELING

ER DIAGRAM

An Entity Relationship Diagram (ERD) is a visual representation of different entities within a system and how they relate to each other. Here are the geometric shapes and their meaning in an E-R Diagram.

* Rectangle: Represents Entity sets.
* Ellipses: Attributes
* Diamonds: Relationship Set
* Lines: They link attributes to Entity Sets and Entity sets to Relationship Set
* Double Ellipses: Multivalued Attributes
* Dashed Ellipses: Derived Attributes
* Double Rectangles: Weak Entity Sets
* Double Lines: Total participation of an entity in a relationship set

**ER diagram has three main components:**

1. **Entity:** An entity is an object or component of data. An entity is represented as rectangle in an ER diagram.
2. **Weak Entity:** An entity that cannot be uniquely identified by its own attributes and relies on the relationship with other entity is called weak entity. The weak entity is represented by a double rectangle.
3. **Attribute:** An attribute describes the property of an entity. An attribute is represented as Oval in an ER diagram. There are four types of attributes:
4. Key attribute
5. Composite attribute
6. Multivalued attribute
7. Derived attribute
8. **Key attribute:** A key attribute can uniquely identify an entity from an entity set. Key

attribute is represented by oval same as other attributes however the text of key attribute is underlined.

1. **Composite attribute:** An attribute that is a combination of other attributes is known as composite attribute.
2. **Multivalued attribute**: An attribute that can hold multiple values is known as multivalued attribute. It is represented with double ovals in an ER Diagram.
3. **Derived attribute:** A derived attribute is one whose value is dynamic and derived from another attribute. It is represented by dashed oval in an ER Diagram.

**3. Relationship:** A relationship is represented by diamond shape in ER diagram, it shows the relationship among entities.

There are four types of relationships:

1. One to One
2. One to Many
3. Many to One
4. Many to Many
5. **One to One Relationship:** When a single instance of an entity is associated with a single instance of another entity then it is called one to one relationship.
6. **One to Many Relationship:** When a single instance of an entity is associated with more than one instances of another entity then it is called one to many relationship.
7. **Many to One Relationship:** When more than one instances of an entity is associated with a single instance of another entity then it is called many to one relationship.
8. **Many to Many Relationship:** When more than one instances of an entity is associated with more than one instances of another entity then it is called many to many relationship.

**Total Participation of an Entity set:** A Total participation of an entity set represents that each entity in entity set must have at least one relationship in a relationship

#### TABLE DESCRIPTION

A table is an arrangement of data in rows and columns, or possibly in a more complex structure. A table is a collection of related data held in a table format within a database. The database management system (DBMS) is the software that interacts with end users, applications, and the database itself to capture and analyse the data.

A database consists of one or more tables. Each table is made up of rows and columns. Each row in a relational table is uniquely identified by a primary key. This can be by one or more sets of column values. In most scenarios it is a single column, such as student ID.

Every relational table has one primary key. Its purpose is to uniquely identify each row in the database. No two rows can have the same primary key value. The practical result of this is that you can select every single row by just knowing its primary key.

#### NORMALIZATION

Normalization is the process of minimizing redundancy from a relation or set of relations. Redundancy in relation may cause insertion, deletion and updating anomalies. So, it helps to minimize the redundancy in relations. Normal forms are used to eliminate or reduce redundancy in database tables.

* + - Normalization is the process of organizing the data in the database.
    - Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate the undesirable characteristics like Insertion, Update and Deletion Anomalies.
    - Normalization divides the larger table into the smaller table and links them using relationship.
    - The normal form is used to reduce redundancy from the database table.

Types of normal forms:

**NORMAL FORMS**

**1NF**

**3NF**

**2NF**



1NF

A relation is in 1NF if it contains an atomic value.

**Description**

**Normal Form**

|  |  |
| --- | --- |
| 2NF | A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key. |
| 3NF | A relation will be in 3NF if it is in 2NF and no transition dependency exists. |

**First Normal Form(1NF)**

* + - A relation will be 1NF if it contains an atomic value.
    - It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.
    - First normal form disallows the multi-valued attribute, composite attribute, and their combinations.

**Second Normal Form(2NF)**

* + - In the 2NF, relational must be in 1NF.
    - In the second normal form, all non-key attributes are fully functional dependent on the primary key

**Third Normal Form(3NF)**

* + - A relation will be in 3NF if it is in 2NF and not contain any transitive partial dependency.
    - 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
    - If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.

A relation is in third normal form if it holds at least one of the following conditions for every non-trivial function dependency X → Y.

1. X is a super key.
2. Y is a prime attribute, i.e., each element of Y is part of some candidate key.

## .

## TESTING AND VALIDATION

#### INTRODUCTION

Testing is the major quality control method used during software development. It is the basic function to detect errors. During the requirement analysis and design the output of the document that is usually textual and non-executable after the coding phase the computer programs are available that can be executed for testing purpose. The goal of testing is to uncover requirement, design and coding errors in the program. Testing determines whether the system appears to be working according to the specifications. It is the phase where we try to break the system and we test the system with real scenarios at a point. The implementation is the final and important phase. It involves user-training system testing.

In order to ensure successful running of the proposed system, the user tests the system and changes are made according to their needs. The testing involves the testing of the developed system using various kinds of data, while testing errors are noted and correctness is made.

#### TESTING OBJECTIVES

The objectives of testing are:

* + - Testing is the process of executing a program with the intent of finding errors.
    - A successful test case is one that uncovers un-yet-discovered errors.

System testing is a stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as per the user need, before the live operation commences. As stated, before testing is vital to the success of a system. System testing makes a logical assumption that if all the parts of the system are correct, the goal will be successfully achieved. A series of test are performed before the system is ready for user acceptance test.

#### TESTING METHODS

System testing is the stage of implementation. This is to check whether the system works accurately and efficiently before live operation commences. Testing is vital to the success of the system. A series of test are performed for the proposed system is ready for the user acceptance testing.

#### TESTING STEPS

* + 1. **Unit testing**

Unit testing is a level of software testing where individual units/components of software is tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable parts of any software. It usually has one or few inputs and usually a single output.

* + 1. **Integration testing**

Integration testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in integration testing. It is performed to expose defects in the interfaces and in the interactions between integrated components or systems.

* + 1. **Validation**

Validation testing can be defined in many ways, but a simple definition is that the validation succeeds when the software functions in the manner that is expected by the client. After validation test has been conducted, one of the three possible conditions exists. The function or performance characteristics confirm to specifications and are accepted. The deviation from specification is uncovered and a deficiency list is created. Proposed system under consideration has been tested by using validation test and found to be working satisfactory.

* + 1. **Output testing**

After performing the validation, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in a specific format. The output format on the screen is found to be correct. The format was designed in the system design time according to the user needs. For the hard copy also, the output comes as per the specified requirements by the user. Hence output testing did not result in any correction for the system.

* + 1. **User acceptance testing**

Acceptance testing is the level of software testing where a system is tested for a acceptability. It is a formal testing with respect to other needs, requirements conducted to determine whether or not a system satisfies an acceptance criteria and to enable the users or other unauthorized to determine whether or not to accept a system.