Project Clay

MAJOR PROJECT REPORT

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF (12pt.)

BACHELOR OF TECHNOLOGY
(Computer Science and Engineering)

Submitted By:

Pankaj Ramola (2104151)
Nikhil Bidlan (2104146)

Submitted To.:

Prof. (Golden Deep Kaur)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GURU NANAK DEV ENGINEERING COLLEGE
LUDHIANA, 141006 (14pt.)

May, 2025

ACKNOWLEDGEMENT

We are highly grateful to the Dr.Sehijpal Singh , Principal, Guru Nanak Dev Engineering College (GNDEC), Ludhiana, for providing this opportunity to carry out the major project work at Project Clay .

The constant guidance and encouragement received from Dr. Kiran Jyoti H.O.D. CSE Department, GNDEC Ludhiana has been of great help in carrying out the project work and is acknowledged with reverential thanks.

We would like to express a deep sense of gratitude and thanks profusely to Project Guide *Pf. Goldendeep Kaur without his/her wise counsel and able guidance, it would have been impossible* complete the project in this manner.

We express gratitude to other faculty members of computer science and engineering department of GNDEC for their intellectual support throughout the course of this work.

Finally, we are indebted to all whosoever have contributed in this report work.

Pankaj Ramola

Nikhil Bidlan

LIST OF FIGURES (Sample)

LIST OF TABLES (Sample)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

“ **Project Clay** ” is a web based online project. The main aim of the project is to provide a centralized point from where every Student in the India can view and get mentorship of their Deals. It is an online Counseling Based project that serves the functionality of one to one connect. The system allows only registered users login and new User are allowed to register on the application. This is proposed to be a web application. This project gives user's the opportunity to the get Session and mentorship provided by all the College mentors enables them to get their guidence at home. This project allows mentors to direct communicate with users.

CHAPTER 1

Gathering and Analyzing Information

1.1 INTRODUCTION

Here, a web based interaction system of Project Clay is developed. This site manages all the Mentors related information, this system also maintain records of all Users of College who register on the website. In this system administrator and Mentors can view their deals and details related to their Mentors. Customers can also chat with the administrator of Mentors for any query about their menu content. Administrator can add, edit, view, update, and delete any information on the website. Customers can view the information authorized by the administrator.

This is a web application allows you to access the whole information about all Mentors of College. The collection of the resources and information at one place in an organized way is made accessible to all the registered Customers and Mentors of College.

Project Caly is a collection of information of all Mentors of College are stored in one place for current and later use anytime and anywhere. The purpose of this project is to collect, display and store all the deals and details of guidance of all Mentors of College at one website point. To provide web-based communication there is also online web based chat box for necessity, to ask about any deal and detail about guidance. Manual searching for the Mentors and reasonable prices is most time consuming and difficult as compared to Manual Suggestion Based System. This system shows the comparison of various existing guidance items and their prices. In this project, I will introduce some new ideas for improving the existing systems.

1.2 PURPOSE

This project aims at creating an Project Caly for online guidance ordering from all over the College. This allows registered users of the system to easily log in and can easily visit all restaurant's menu online that are available on the site and choose the menu available for the order. Saving time, money & easily order by sitting at home are major goals of this project. Customers can register themselves and view all the allowed details of the Mentors of all College. The Main objective of Project Caly is to gather all the Mentors details to one same place and suggest. In this way, every customer will be

able to order any kind of guidance from any Restaurant from this App/Website. Below are the some objectives we will cover in this project.

- We can save detail of all Mentors in College.
- All Deals and Guidance Staff Record.
- All Customer Details.
- System for any person

1.3 SCOPE

The Project Caly Cell that is to be developed provide the mentors as well customers of University of Education deals and Guidance Stuff and notes information and also their next and previous order details. We will firstly cover all mentors of College and after further implementation and improvements, it will be expand to Punjab region. The E-Menu is supposed to have the following features:

- The project provides the customers and mentors with online deals and Guidance Stuff viewing and downloading.
- The project provides login facility to the customers.
- The system provides the customers suggestions about mentors and guidance items according to their budget.
- The customers can chat with the administrator for any query about mentors details and deals .

The features that are described in this document Project Caly will cover all the Mentors in College City. After this, we will expand it to Punjab Region for other further implementation after improvement.

Document are used in the future phases of the project development cycle. The features described here meet the needs of all the customers. The success criteria for the system is based in the level up to which the features described in this document are implemented in the system.

1.4 DEFINITIONS, ACRONYMS AND ABBREVIATIONS

The definitions, acronyms and abbreviation used in project are:

- PHP : Hypertext Preprocessor
- UML: Unified Modeling Language
- HTML : Hypertext Markup Language
- CSS : Cascading Style Sheet
- SDLC : Software Development Life Cycle
- OS : Operating System
- XML : Extensible Markup Language
- DFD : Data Flow Diagram
- MySQLi : My Structured Query Language (Improved)
- AUP : Agile Unified Process
- RUP : Rational Unified Process
- TDD : Test Driven Development
- RIPP : Rapid Iterative Production Prototyping
- RAD : Rapid Application Development
- ERD : Entity Relationship Diagram

CHAPTER 2

Software Requirement Specification

2.1 FUNCTIONAL REQUIREMENTS

2.1.1 Network Login:

Since the system needs to handle a lot of confidential customer information, a network login function is essential for ensuring security. Users who do not have the correct access rights will be prevented from connecting to the database. There are two groups of users of the system with different access rights:

- Admin - acts as the system administrator and can perform all functions.

Customer - can view and order deals and guidance Items.

2.1.2 Customer Information Management

This function allows the personal information of customer, such as their name, phone no., address, etc., to be managed. Using this function a customer can:

- View or order the deals and guidance items
- Can Check his/her order status

2.1.3 Admin Information Management

Project Caly app needs to change the records of items/ deals, such as the particulars, their prices, the type or Image, etc. Using this function and admin can:

- Insert a new record for a deal/Guidance Item;
- Make changes to the record of a Guidance Items;
- Search for a particular item or a group of items in a record.

2.2 NON-FUNCTIONAL REQUIREMENTS

2.2.1 Reliability

Project Caly reliability requirements deals with failures to provide online guidance order services. As this project is based totally on PHP so it provides the advanced functionality of better Security and Usability.

2.2.2 Supportability

Project Caly is designed & developed in HTML5, CSS3, Bootstrap4, PHP7.0, and JavaScript, it is supported by every device including PCs, Laptops, Tablet, Smart phones, and any handheld devices connected to the internet. It supports all the browsers versions.

2.2.3 System Requirements

Now, this method is intended in such the way that it takes fewer resources to figure out work correctly. It's its type of minimum needs that we'd like to require care of :

- System neesd 2GB of RAM to run all the operations smoothly.
- It wants a minimum of 1.3GHz of Processor for all processes to run smoothly.
- The system must be controlled by approved person as wrong hands can delete deals and Guidance Stuff, notes, results or leak them.
- Rest is all up to the customer's usage of project.
- The system is made correctly without any bug or error. All the testing is done as per the requirements. So everything now depends on the usage of this project.

2.2.4 Usability

- The system shall allow the users to access the Project Caly from the Internet using HTML or it's derivative technologies. The Project Caly uses a web browser as an interface.
- No specific training is required as every person knows how to use Browser now adays.
- The Project Caly is user-friendly, self-explanatory, and responsive.

2.2.5 Efficiency

Project Caly is very efficient and compatible to every hardware on which it will run. Whenever admin or customer search any query it will respond in no time(very fast).

2.2.6 Portability

Project Caly is adaptable to every environment of other Mentors of College. It is designed and developed in such as way that it can be ported to any hardware of Mentors.

2.3 DOMAIN REQUIREMENTS

The Domain Requirements of Project Caly are below and all these domain requirements are capable to fullfill the requirements.

- XAMP/Local Server
- HTML5
- CSS3
- PHP7
- Javascript
- MySQLi

2.4 USE CASES AND USAGE SCENARIOS

2.4.1 Use Case Diagrams

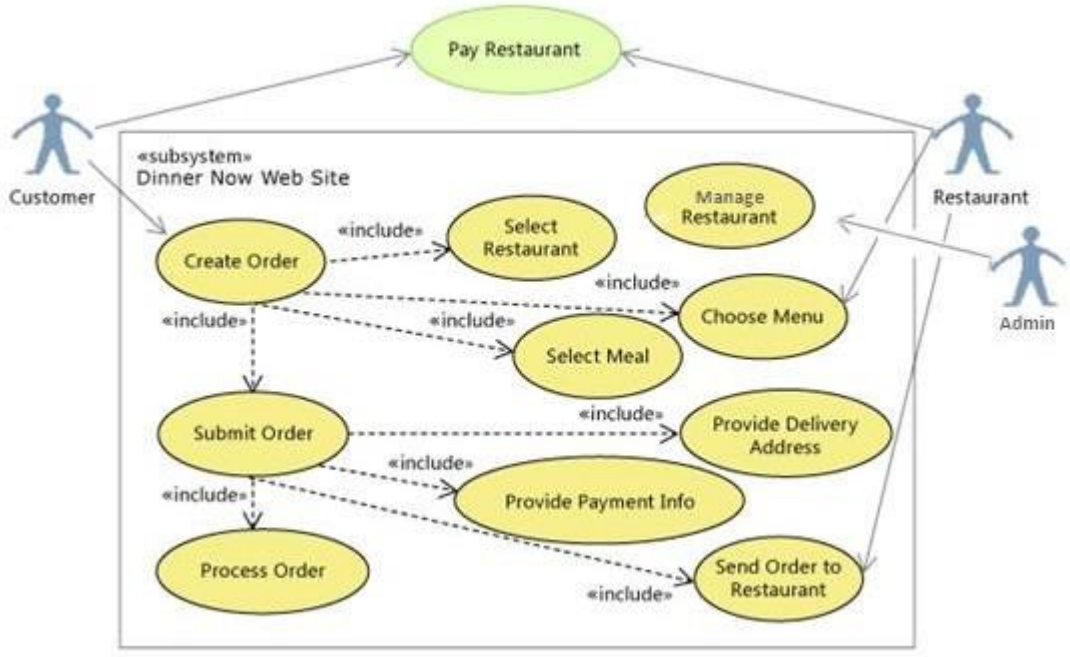


Figure 2.4.1.1: Use Case Diagram

In Use Case Diagram, a customer can view and order deals. Admin can access every panel including Customer Dashboard. Customer can check order status. He can also view and order deals and guidance items.

1. Use case name: Place Order
Description: Customer places an order from the available choices.
Primary Actor: Customer

Pre-conditions: System is connected to a power source, display is turned on and system is configured to accept the inputs.
Flow of Events: 1. User selects his language preference for the session. 2. User selects from the menu. 3. User confirms the order **Alternative Flow of Events:** 1. User accidentally presses a wrong button and after realizing it he hits the backspace button. 2. User enters a wrong order and wants to go back to the main menu.

Post Condition: Order has been made that goes to the system for processing.	**Assumption:** User is familiar with how to enter values through mouse and has a general idea

Table 2.4.1.1: This Table shows the process of Order Placement

1. Use case name: customer, On Delievery, cash collector.
Description: The user is asked for the mode of payment. The payment is accepted or is collected by cash collector. And the customer is given a token with their order number.
Primary Actor: Customer
Pre-conditions: The order has been confirmed and the total bill has been displayed on the screen to the customer. Costumer decides to go ahead with the order.
Flow of Events: • User enters the mode of payment. (online/on Delievery) • User makes the payment in cash • Cash collector collects the money and gives back the change if required.

<table>
<tr><td colspan="2">

- User receives a token number and final bill.

Alternative Flow of Events:

1. User accidentally presses a wrong button and after realizing it he hits the backspace button.
2. User enters a wrong order and wants to go back to the main menu.

</td></tr>
<tr><td>Post Condition: Customer waits for the order to be processed.</td><td>Assumption: User is familiar with how the system works and what is expected out of system.</td></tr>
</table>

Table 2.4.1.2: This Table shows the process of Delivery

<table>
<tr><td colspan="2">1. Use case name: Monitor Guidance Items</td></tr>
<tr><td colspan="2">Description: The user is asked for the mode of payment. The payment is accepted or is collected by cash collector. And the customer is given a token with their order number.</td></tr>
<tr><td colspan="2">Primary Actor Guidance preparation person, Store Manager Description: This use case triggers when an item goes out of stock.</td></tr>
<tr><td colspan="2">Pre-conditions: None</td></tr>
<tr><td colspan="2">

- Flow of Events:
  1. Guidance preparation person/Store manager notices an item out of stock
  2. Updates the menu accordingly.

</td></tr>
<tr><td>Post Condition: A new and updated menu list will be displayed.</td><td>Assumption: The Store manager is given the rights and privileges to enter the system and make the required changes.</td></tr>
</table>

Table 2.4.1.3: This Table shows the Monitor Guidance Items

1. Use case name: Read Order.	
Description: Internal order system reads the order once the customer confirms his order and then he communicates the order to the Restaurant person.	
Primary Actor Guidance preparation person, Internal Order system.	
Pre-conditions: User confirms the order.	
• **Flow of Events:** 1. Internal order system reads the order 2. Communicates the order to the Restaurant	
Post Condition: A new and updated menu list will be displayed.	**Assumption:** The Store manager is given the rights and privileges to enter the system and make the required changes.

Table 2.4.1.3: This Table shows the Read Order.

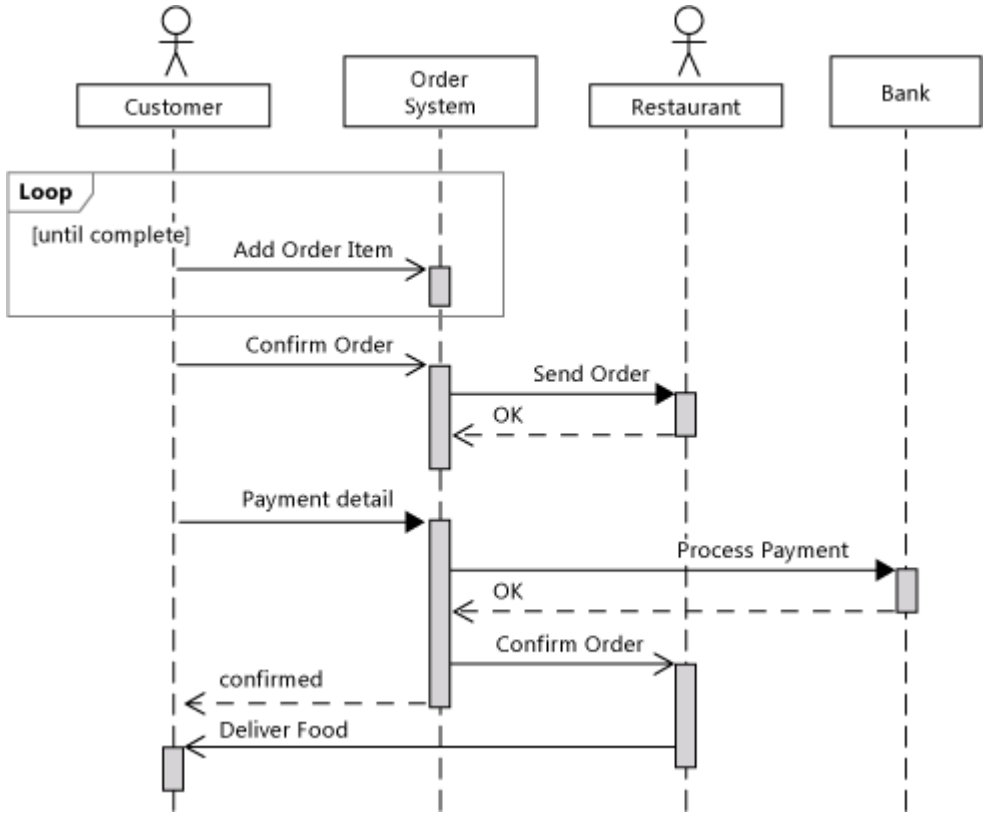


Figure 2.4.1.2: Sequence Diagram

In Sequence Diagram, we show the activities of the customer and it's steps how he check and access the website. Customer access website and website is fetched by web server which can be fetched by Database. Admin controls the Database and Web Server.

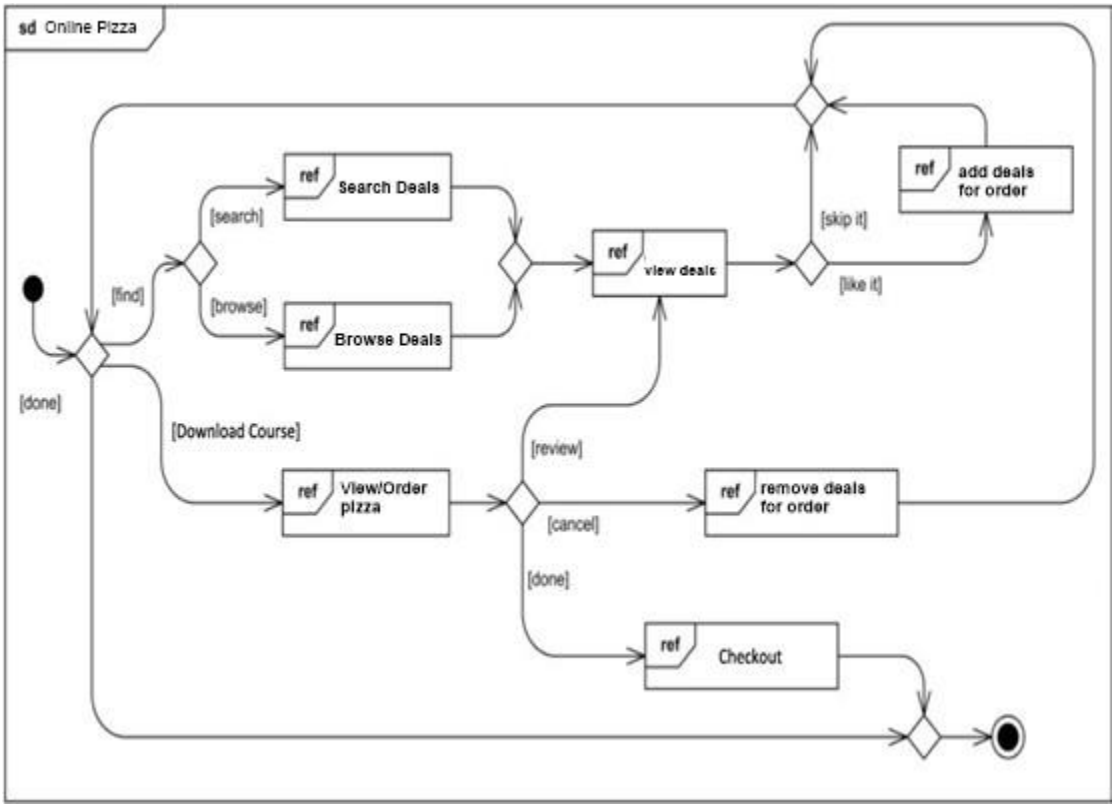


Figure 2.4.1.3: Interaction Overview Diagram

In Interaction Overview Diagram, we show the customer open the website search deals and order it. He can also contact with the admin for any error or query. He can search, browse, view or order the guidance items and deals and can check out.

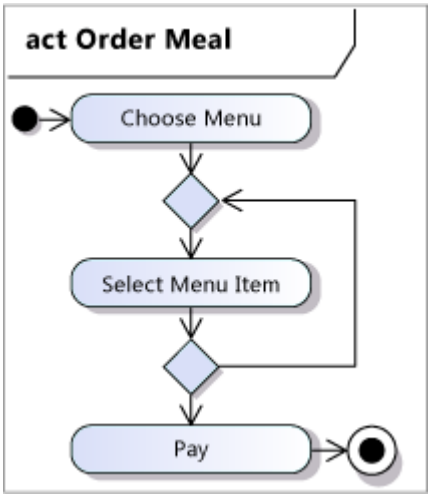


Figure 2.4.1.4: Activity Diagram

In this Activity Diagram, we show the flow of admins how they can login and access the website. admins can view and update information there and new admin/ restaurant can register. Admin/ restaurant can update their profile and can left the website.

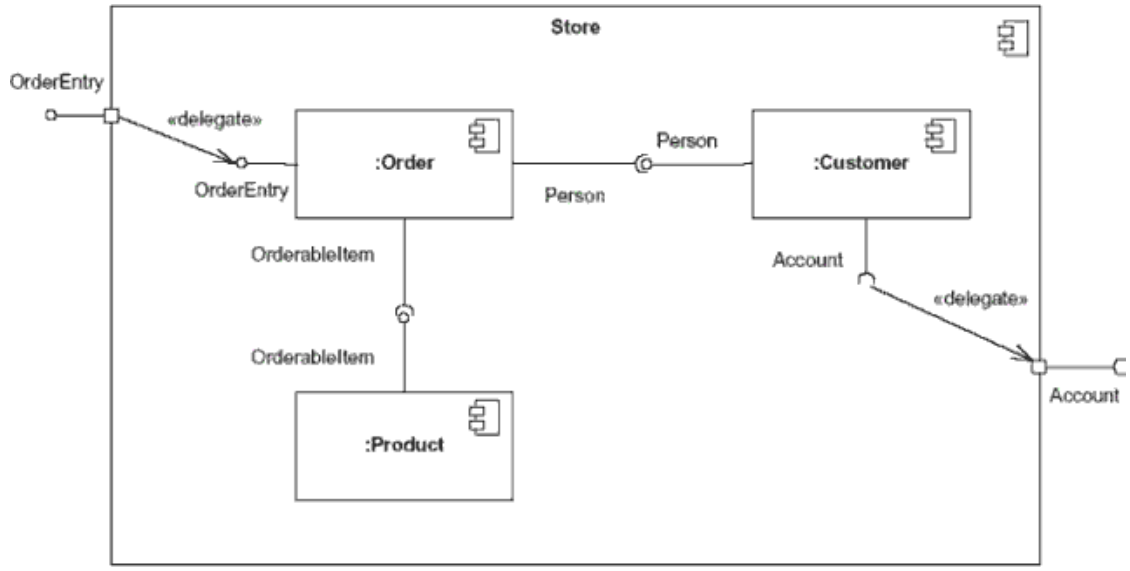


Figure 2.4.1.5: Component Diagram

In the Component Diagram, we show the Internal structure of the Project Caly. Management System in which customer can search, view, and order the deals or guidance items. User can search the deals from the Database we created which contains all type of guidance stuff.

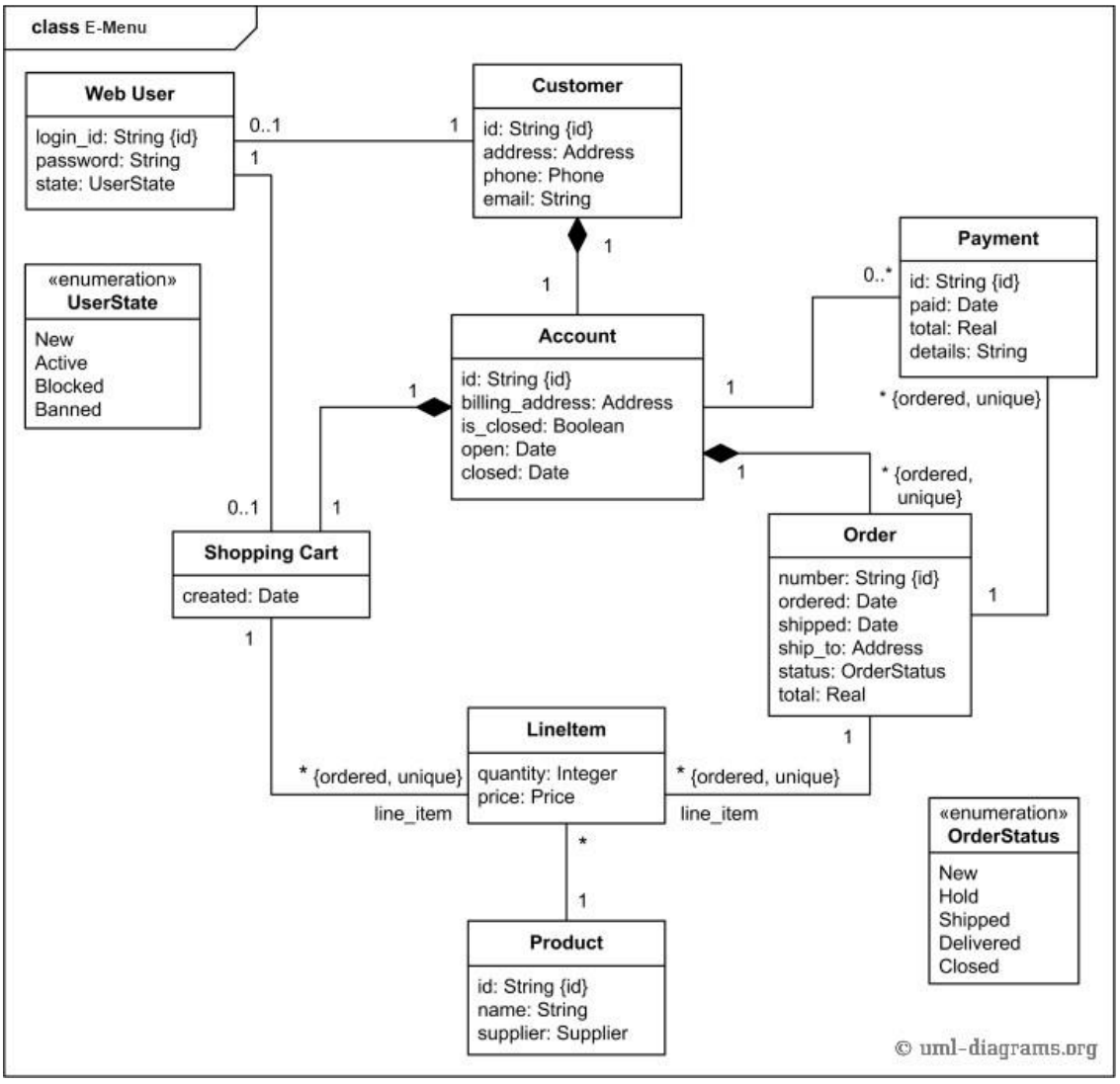


Figure 2.4.1.6: Class Diagram

In the Class Diagram, we show the activities of admin and customer. How admin can login and what he can do on the website like approve order, add items, modify items, add items, and add deals.

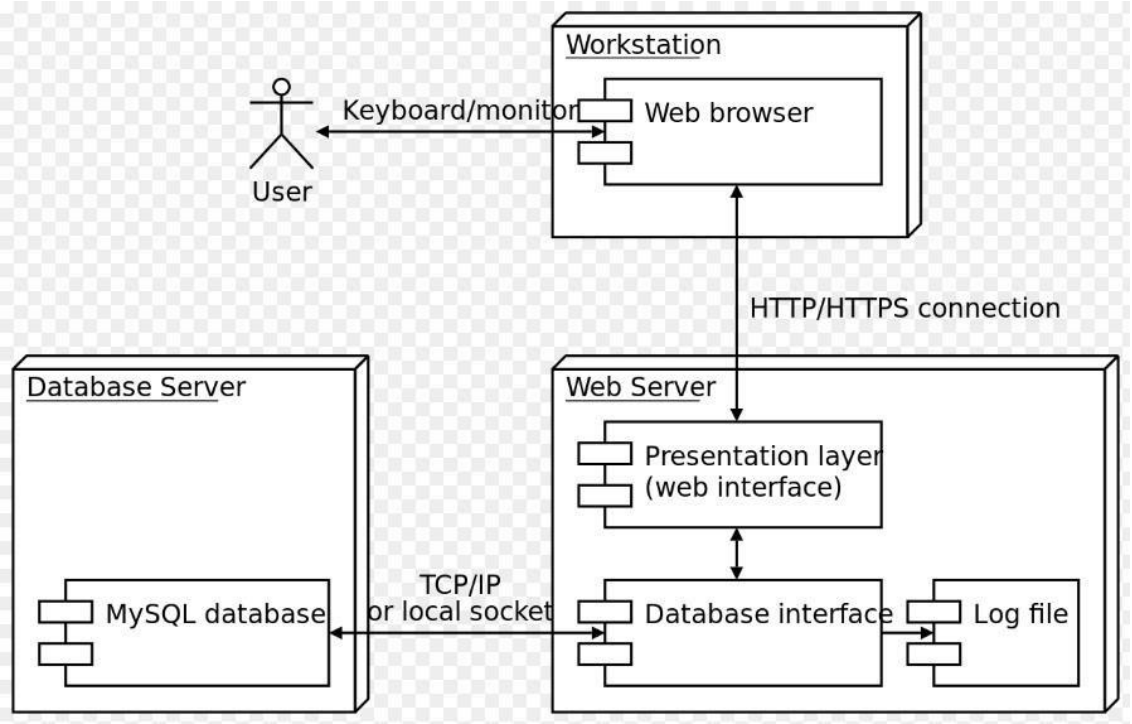


Figure 2.4.1.7: Deployment Diagram

In the Deployment Diagram, user visit the Project Caly to check the deals and order it from the database. The history of deals and it's details are saved in the database server.

CHAPTER 3

Planning the Project

3.1 INTRODUCTION

The Planning Phase is the time when the project team transform the initial vision/scope from the Envisioning Phase into practical plans on how to achieve it. The main purpose of the Planning Phase is to define the solution in detail along with the approved project plan and schedule. It states that how and when the project's objective to be achieved. A project planning is a formal, approved document used to guide both project execution and project control.

3.2 METHODOLOGY

A software development methodology or methodology is a framework that is used to structure, plan and control the process of developing a system. It provides a project team with the game plan for implementing the project. A good methodology should be flexible and adapt to the needs of the project organization. It defining the overall goals of the project.

3.2.1 Available Methodologies

Here we have some Methodologies for developing a system.

Rapid Application Development Model

The term was roused by James Martin, who worked with associates to build up another technique called Rapid Iterative Production Prototyping (RIPP). In 1991, this approach turned into the start of the book Rapid Application Development. Quick application advancement is still being used today and a few organizations offer items that give a few or the greater part of the apparatuses for RAD programming improvement. (The idea can be connected to equipment improvement too.) These items incorporate necessities gathering instruments, prototyping devices, PC helped programming building devices, dialect advancement situations, groupware for correspondence among improvement individuals, and testing devices.

3.2.2 CHOSEN METHODOLOGIES

We have used **Agile Unified Process Methodology** to develop Project Caly System.

3.3 REASONS FOR CHOSEN METHODOLOGY

Agile development, in its simplest form, offers a lightweight framework for helping teams, given a constantly evolving functional and technical landscape, maintain a focus on the rapid delivery of business value (i.e., bang for the buck). As a result of this focus, the benefits of agile software development are that organizations are capable of significantly reducing the overall risk associated with software development.

Agile Method have some of these Advantages:

- Stakeholder Engagement
- Transparency
- Early and Predictable Delivery
- Predictable Costs and Schedule
- Allows for Change
- Focuses on Business Value
- Focuses on Users
- Improves Quality

3.4 WORK PLAN

Here is some work plan for unique web portal in which we defined how we plan to work:

SR No	***Activities***	***Days***	***Reason of Durations***
1	Project/Product Feasibility Report	2	Analysis and finding feasibilities which are exists in our Product/project need little bit more concentration by we can judge that what kind of benefits can be obtained to a specific user.
2	Project/Product Scope	1	In scope we concentrate on the limit of area at which our Project/Product facilitating and defined.

3	Project/Product Costing	3	We need the all counts of internal external files. Identifying the files also take more concentration. Calculation of F_i by rating, Cost / FP, Total Estimated Effort, and Total Project Cost involve deep study of project.
4	CPM - Critical Path Method	3	It is a large procedure in which first identifying project activities after that define all activities early start, early finish, late start, late finish and calculation of total slack time and Free slack time. And then finding the critical path by analysis.
5	Gantt Chart	1	We have just entered data in MS Visio application software, because all work for Gantt chart already specify.
6	Introduction to Team member and Tools and Technology	1	We mention just introduction with specific skill set also define tools and technology which we are going to use.
7	Vision document	2	There is need only some more things and description of previous identified parameters of products for generating the Vision.
8	Risk List	1	There is need of finding the uncertainties which can be come in our product/project and they definitely lead to loss also define the approaches to resolve them.
9	System specification and external entities	1	Need analysis according to system specification**.**

10	Use case descriptions	8	First identify the use case name and Description of every single use case according to OOAD.
11	Use Case Diagram	5	Make high level use case diagram and analysis level use case diagram in which we define inclusion, extension and generalization relation ships.
12	Design Class Diagram	4	Already identify Classes in domain model .in DCD first identify attributes and function of every class also define relationships among or between classes.
13	Data Model	1	Make ERD using Microsoft Visio.
14	Interface Creation	15	Interfaces are easy to use, user can easily interact with our system, and it is really a user friendly system.
15	Back-end coding	36	System is capable of performing its required functionality on demand and without failure system can accomplish its given task accurately and completely according to standard time and cost.
16	System Testing	9	This is the phase which aware us the work need some changes or it meet user requirements or not, If not then what changes are required.

Table 3.1 Work Plan

3.5 PROJECT STRUCTURE

- Prepare for the activity of site mapping
- Brainstorm the types of content
- Define primary navigation
- Flesh out second & third level structure & content

- Don’t forget about utility pages
- Create notes and high-level specifications for each page
- Designate the type of design template
- Iterate. Iterate. Iterate.

3.5.1 Team Structure

A good web development project nearly always has the following necessary roles:

- Content Creator – helps generate new content for the website.
- Project architect – develops the overall concept of the project (not the design)
- Project manager – manages the project and keeps everything on track
- Site designer – designs the look and functionality of the site
- Back-end developer – creates the framework the site interface will be placed on
- Front-end developer – creates and implements interface components
- Site tester – ideally does everything possible to try and break the site until it can be broken no more.

3.5.2 PROJECT SCHEDULE

Scheduling is all about keeping your workflow as organized as possible. At its core, it allows you to define your project’s final deadline and determine the individual milestones that must be met in order to hit that deadline. Here we have seven steps for creating project schedule.

- Identify individual deliverables
- Define the sequence of activities
- Determine the resources required
- Identify external factors that could influence your project
- Establish key milestones and final deadlines
- Create your schedule
- Monitor and reforecast

CHAPTER 4

Designing the Project

4.1 INTRODUCTION

Any Design in an application is the base to attract any user to our website or an application. When user visit Website or an application he has no concern about the code used behind the Website or application. He not need to know which tool is used in this design of the Application. The Design of any Particular Website or an application represents its Functionality. Proper Designing or Structure of the website Provide User Support.

4.2 PURPOSE

1. The Purpose of the application is to facilitate the Customer.

2. It Provides data about all Resturants of College

3. It stores the all kind of Information about Resturants.

4. It Generates Information of the nearby all Resturants.

5. It Stores all kind of Record of Resturants.

6. Many More….

4.3 SCOPE

1. To Provide Customers the Access of ordering any kind of Guidance Online.

2. To Reduce Restaurant Choosing Time.

3. To Provide User Friendly Environment.

4. To Provide Wide Collection of Mentors Record.

5. To Add , Delete, and Manage their Accounts.

6. To Communicate with Customers and Mentors Members.

4.4 DEFINITIONS, ACRONYMS AND ABBREVIATIONS

While making the Project Caly no special acronyms and abbreviations can be used there are simple terms can be used which is given below.

- **HTTP**

Hyper Text Transfer Protocol is transaction oriented client/server protocol between Application and a web server.

- **XML**

Extensible Markup Language is a markup language that was designed to transport and store data

- **MySQLi**

Mysqli is an improved version of Structure Query Language. It is a database management system that provide a flexible and efficient database platform to rise a strong "On Command" business application.

- **Administrator**

The administrator can perform all the activities of the system such like add, remove, update different records and He or She can also create new users and also can delete any deals and Guidance Stuff and user and change its password.

- **CSS**

Cascading Style Sheet is the language for describing the presentation of Web pages, including colors, layout, and fonts. It allows one to adapt the presentation to different types of devices, such as large screens, small screens, or printers. CSS is independent of HTML and can be used with any XML-based markup language.

- **PHP**

PHP (recursive acronym for PHP: Hypertext Preprocessor) is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML.

4.5 ARCHITECTURAL REPRESENTATION (ARCHITECTURE DIAGRAM)

For system developers, they need system architecture diagrams to understand, clarify, and communicate ideas about the system structure and the user requirements that the system must support. It's a basic framework can be used at the system planning phase helping partners understand the architecture, discuss changes, and communicate intentions clearly.

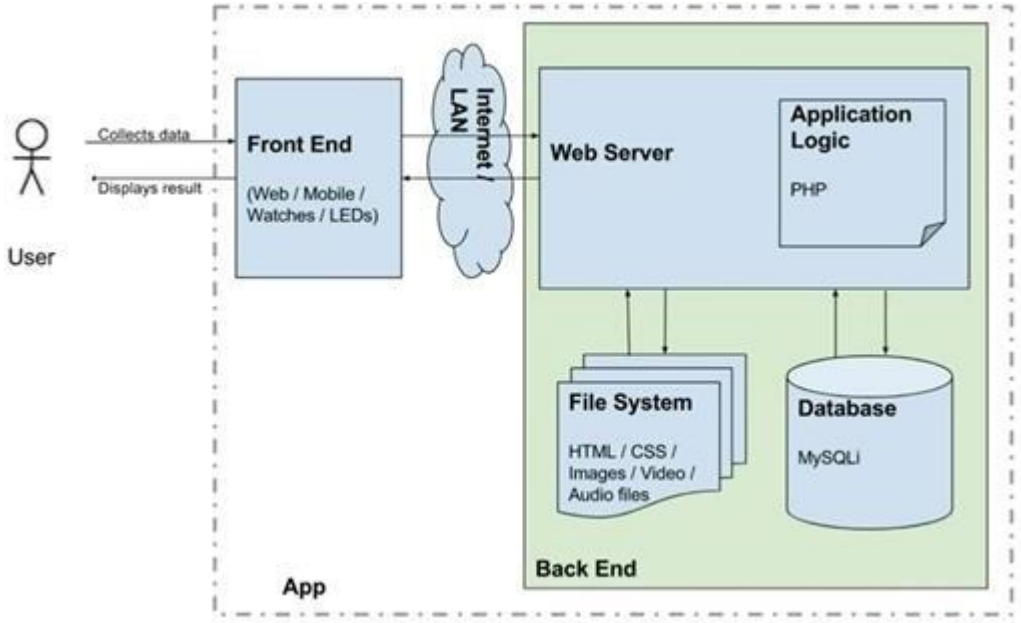


Figure 4.5. ARCHITECTURE DIAGRAM

4.6 DYNAMIC MODEL: SEQUENCE DIAGRAMS

Sequence diagrams are used to show the functionality through a use case. Any actors involved in the system are shown at the top of the diagrams. The objects that the system needs in the order to perform the use cases are shown at the top of the diagram. Each arrow represents a message pass between actors and the objects or object and object to perform the needed functionality. Following diagrams are used to describe the behavior of the system.

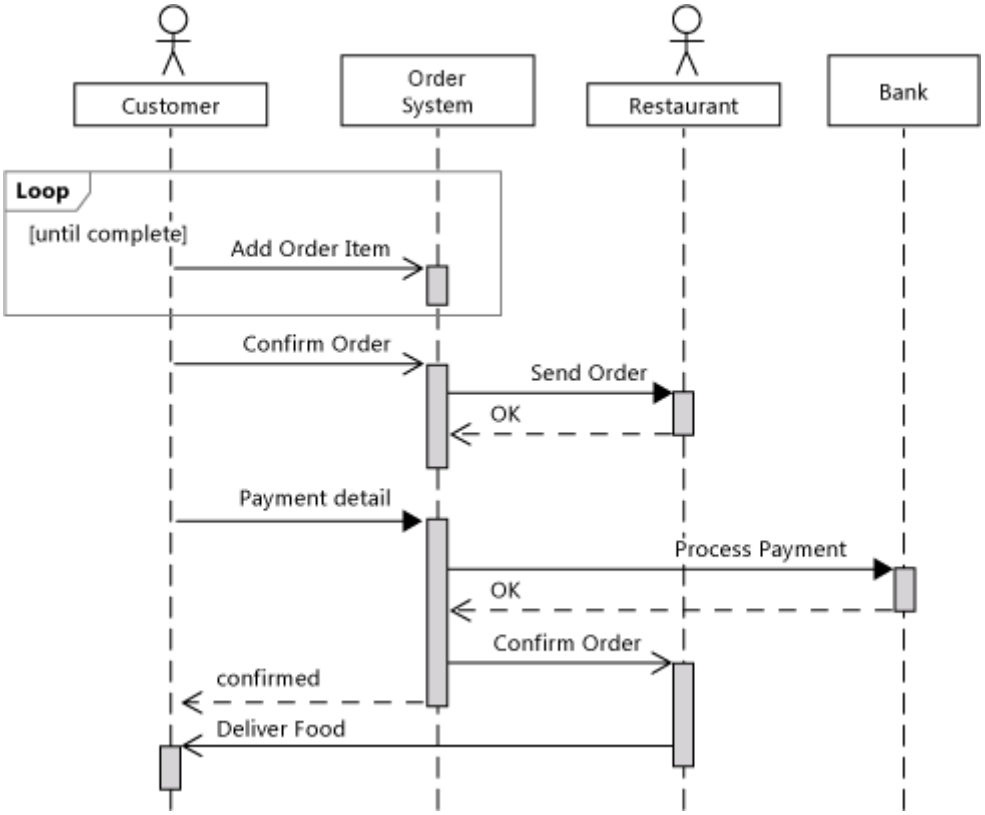


Figure 4.6. SEQUENCE DIAGRAM

4.7 OBJECT MODEL/LOGICAL MODEL: CLASS DIAGRAM

In the Class Diagram, we show the activities of admin and customer. How admin can login and what he can do on the website like approve order, add items, modify items, add items, and add deals.

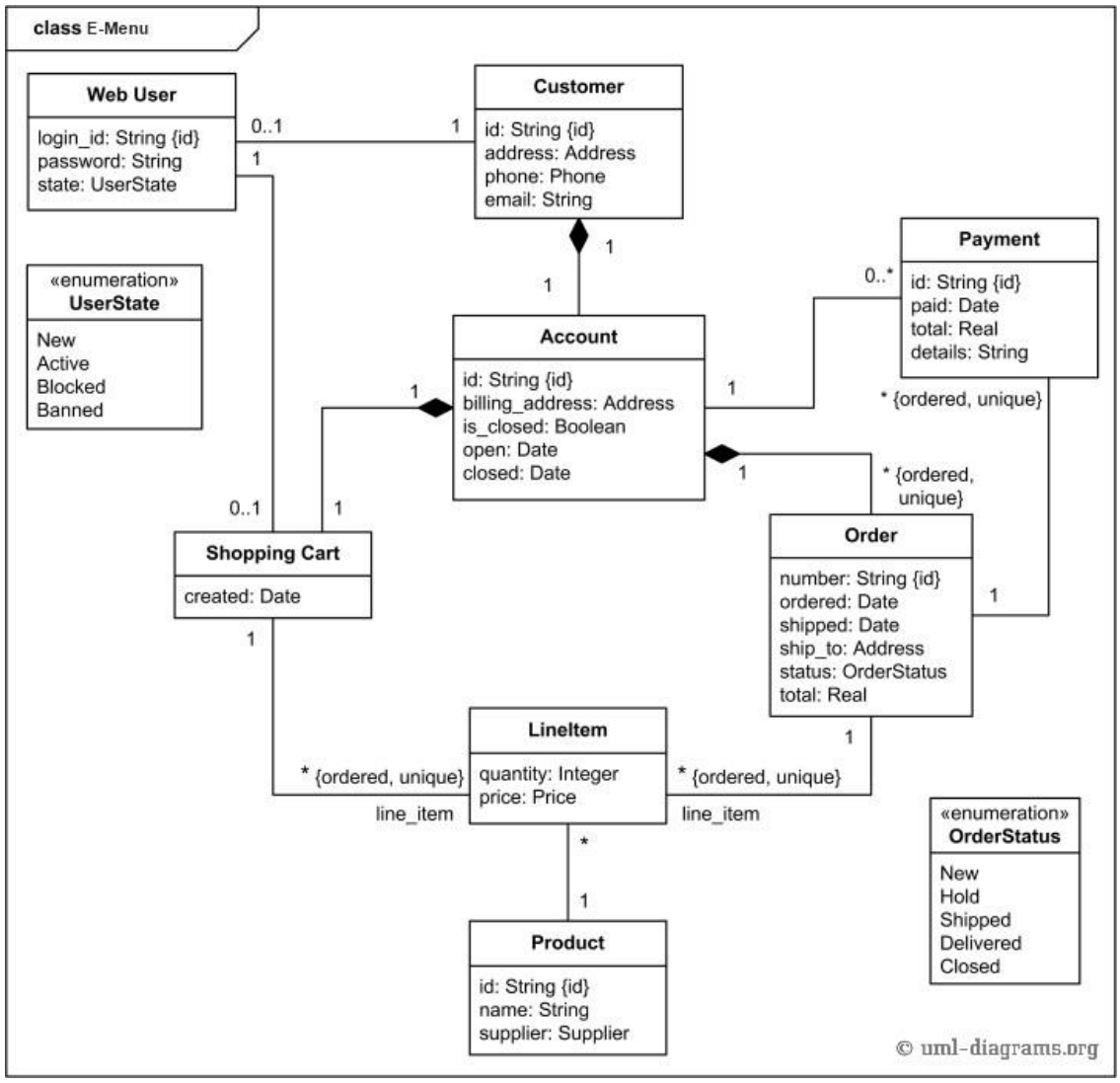


Figure 4.7. CLASS DIAGRAM

4.8 DEPLOYMENT MODEL (DEPLOYMENT DIAGRAM)

Deployment diagrams is a kind of structure diagram used in modeling the physical aspects of a system. Deployment target is usually represented by a node which is either hardware device or some software execution environment. Nodes could be connected through communication paths to create networked systems of arbitrary complexity. In the Deployment Diagram, user visit the Project Caly web app to check the Deals and order it from the database. The history of orders and it's details are saved in the database server.

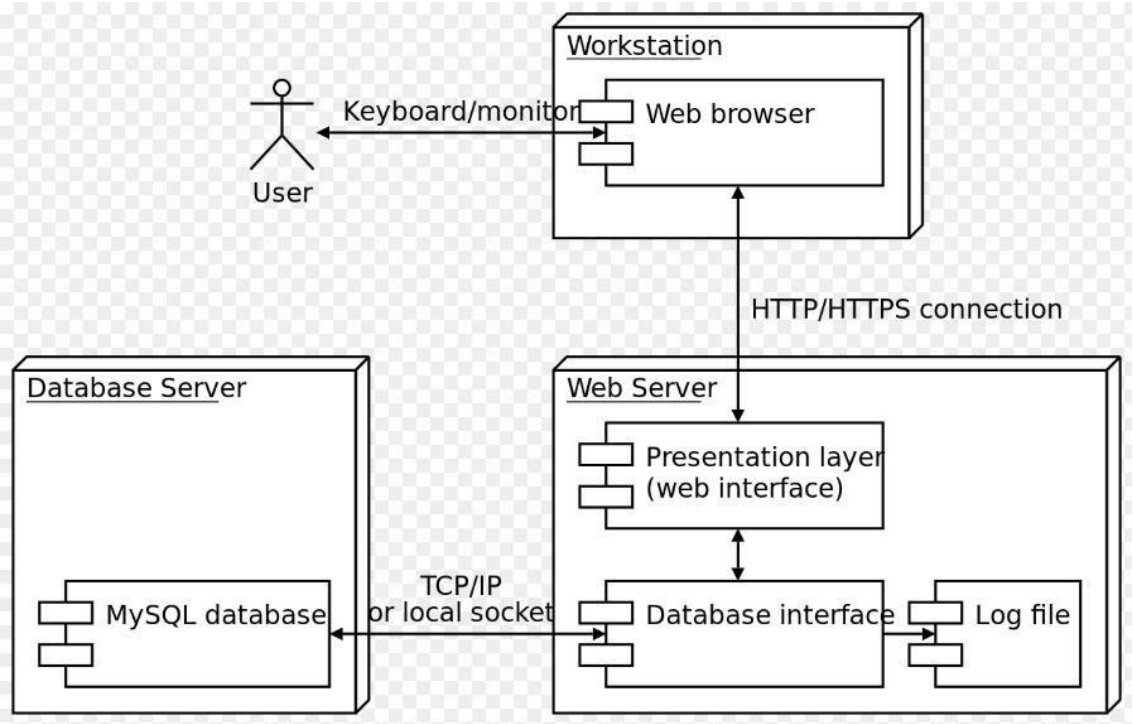


Figure 4.8 DEPLOYMENT DIAGRAM

4.9 DATABASE MODEL (ER-DIAGRAM, DFD)

Many Organizations depend on Web Applications for performing various activities and stores the collected details in the database. Testing of the database is done to ensure the correctness for the design and working of the database. This paper performs testing of database design for the application using the Entity Relationship diagram for various components used in the database, the model is generated from the relational database that represents various data, entities and the relationship among them. It generates the test data in the form of semantics of the prolog facts and the rules. The Guard queries are generated as test cases for the database. These are executed to validate that the design of the database is appropriate for the Web Application with the objects.

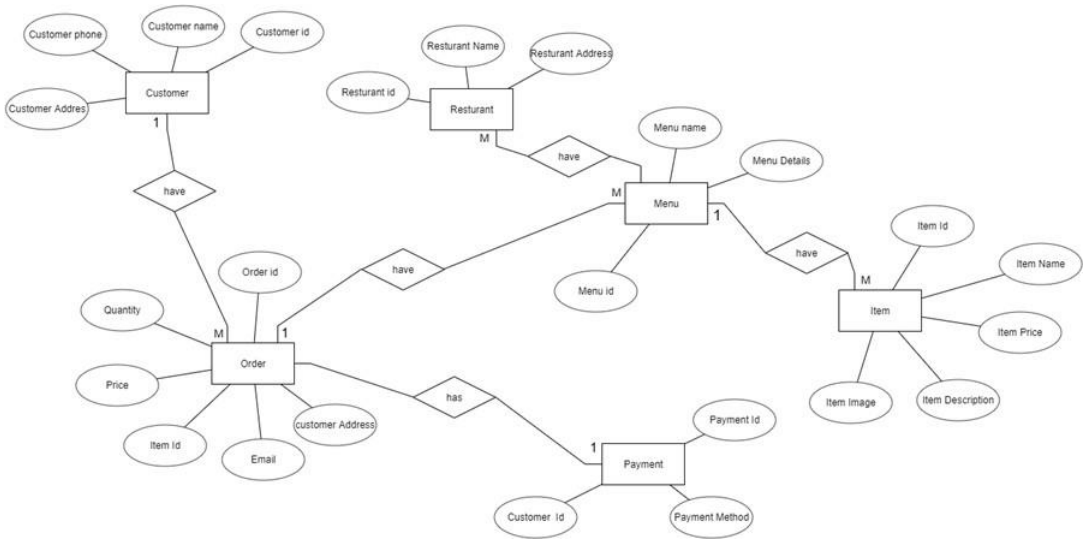


Figure 4.9: Database Model (ER-Diagram)

4.10 GRAPHICAL USER INTERFACES

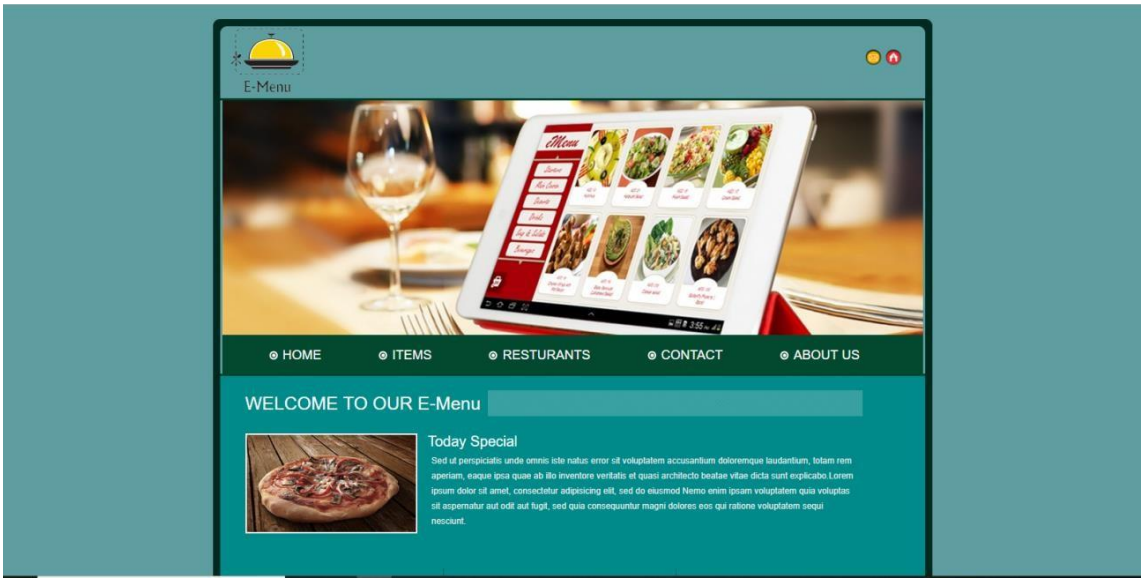


Figure 4.10.1: HOME PAGE

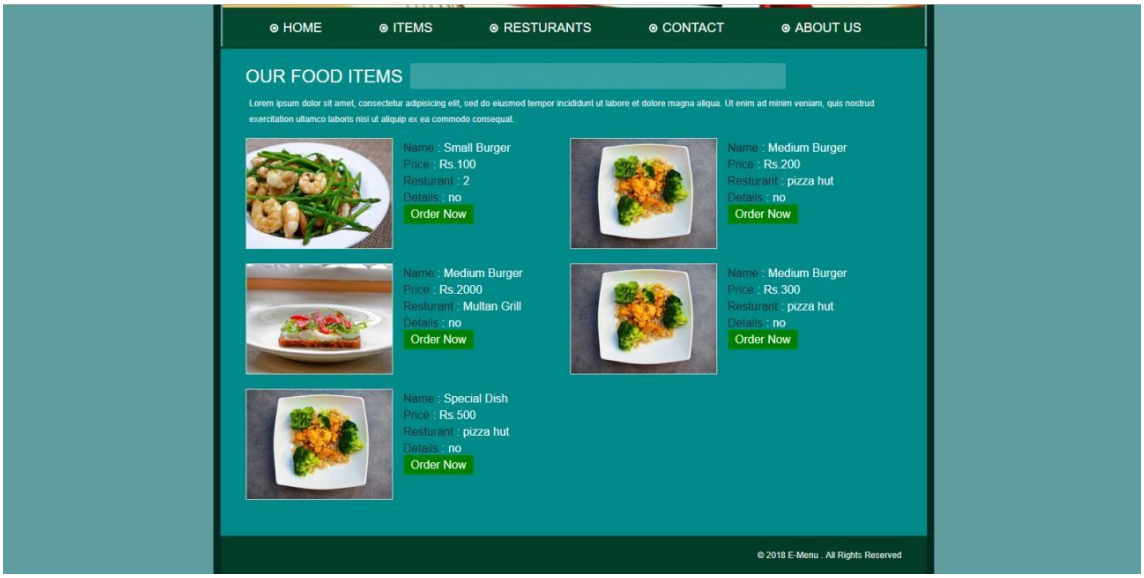


Figure 4.10.2: GUIDANCE ITEM PAGE

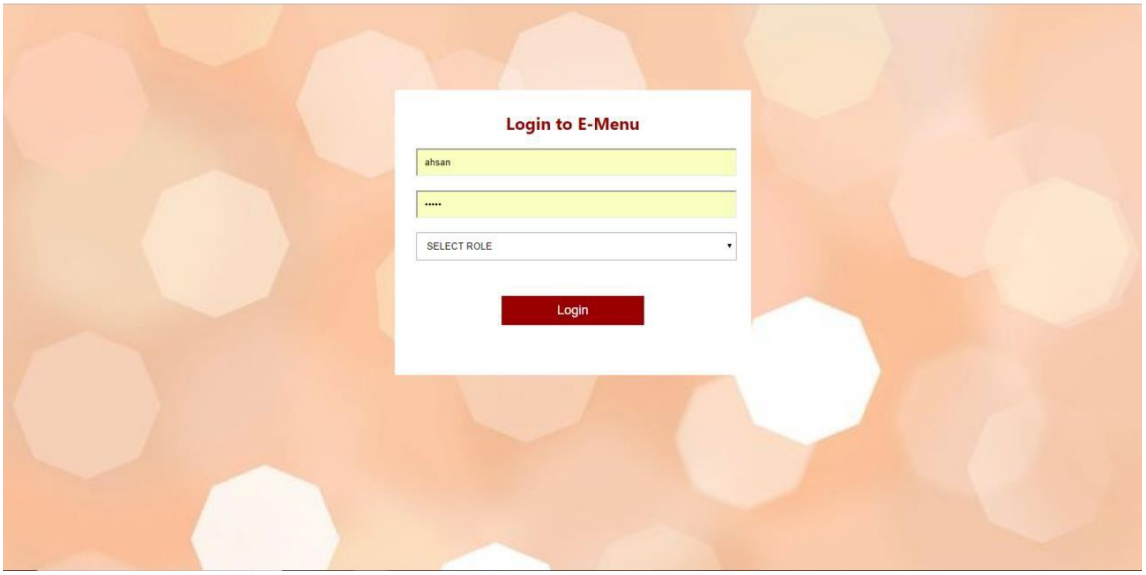


Figure 4.10.3: ADMIN LOGIN PAGE

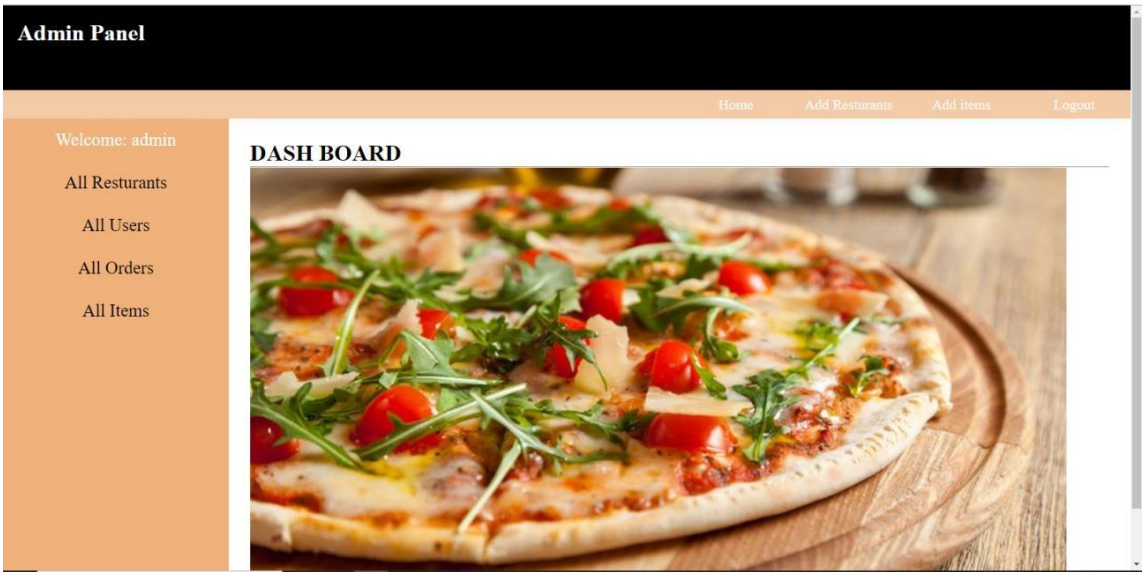


Figure 4.10.4: ADMIN PANEL

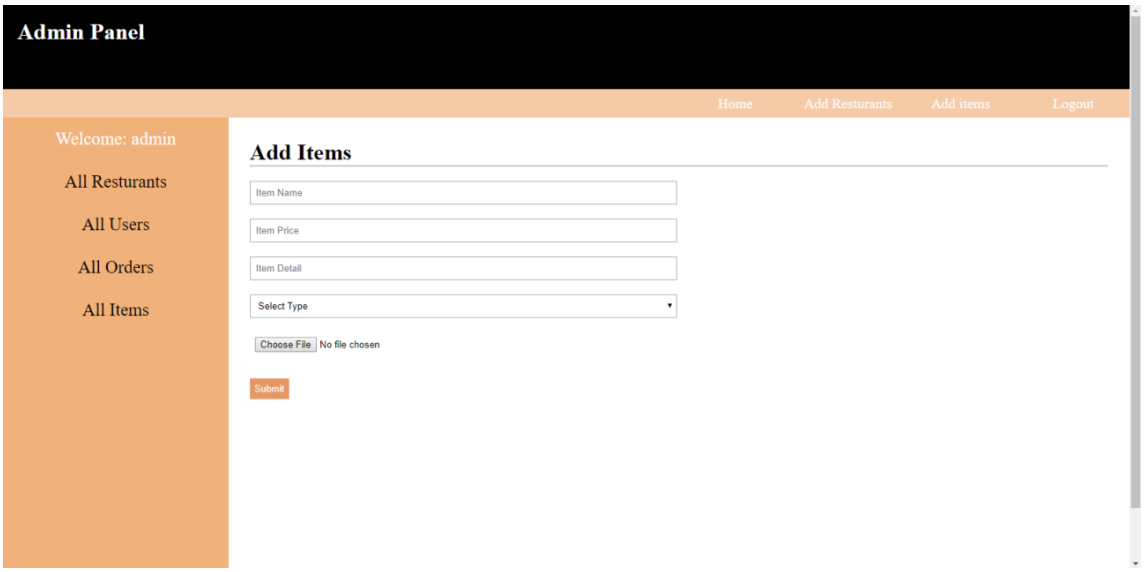


Figure 4.10.5: ADD GUIDANCE ITEM PAGE

CHAPTER 5

Development & Implementation

5.1 DEVELOPMENT PLAN (ARCHITECTURE DIAGRAM)

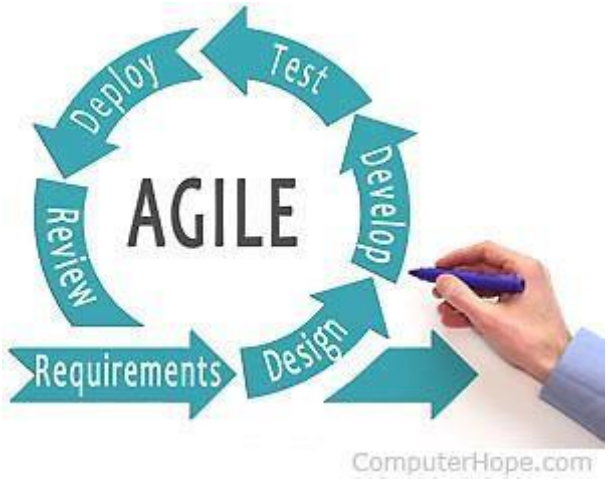


Figure 5.1 Agile Development

Here we have a step-by-step plan for web app Development Process:

- **Creating a roadmap**

You have to understand the direction of the project and establish the goals and purposes of the web application.

- **Define the target audience**

Prepare the analytics report with the following information: type of audience, age, gender, education; web access capabilities of the audience; the level of security and quantitative audience stats.

- **Create a detailed functional specifications document**

It is used to eliminate any sort of confusion in the future, a functional specifications document lists all the technical specifications and functionalities of a web application which is to be developed.

- **Deciding on outsourcing**

Web application development is often cost-effective and faster when outsourced.

- **Selecting Technology**

On this stage you need to define the platform, technology, environment, structure, and framework. Don't forget about the project timeline which has to be decided on the same stage (as far as it largely depends on technology).

- **Designing Layout and Interface**

Here a visual guide or a simple UI sketch has to be created. Once the interface and interaction models are approved, the design is implemented.

- **Web app development**

First, do the application's architecture and framework, design its database structure. Then you have to develop or customize the module, classes, and libraries, and implement all the functionalities mentioned in the specifications.

- **Testing**

That includes QA testing and bug fixing. A web application must be put through the paces, and all suitable testing techniques must be employed, including load testing, stress testing, performance testing, usability testing etc.

ARCHITECTURE DIAGRAM

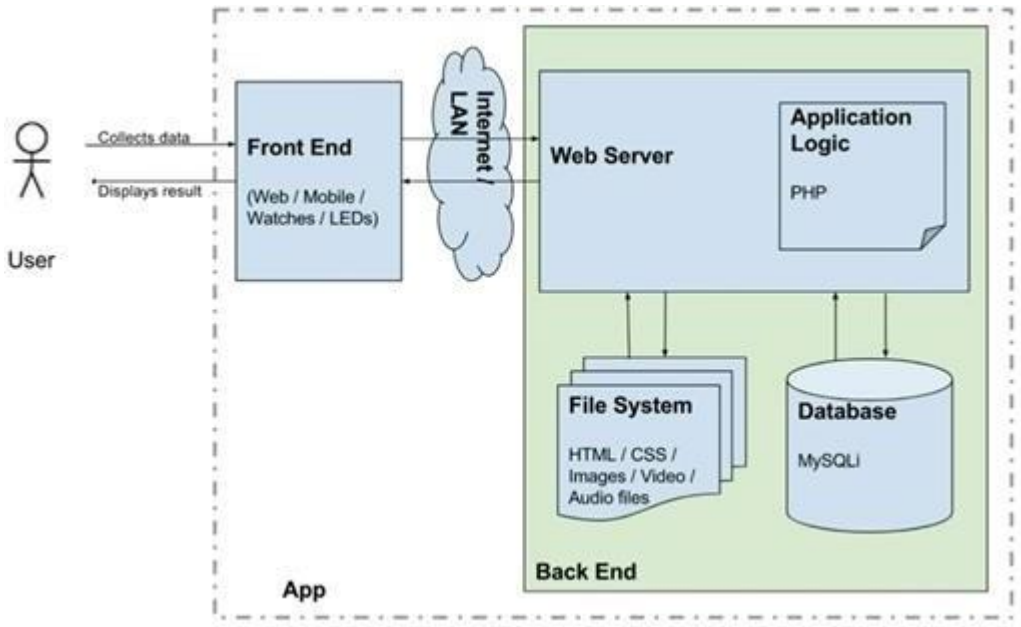


Figure 5.1.1 ARCHITECTURE DIAGRAM

CHAPTER 6

Testing

6.1 INTRODUCTION

Web Testing in simple terms is checking your web application for potential bugs before its made live or before code is moved into the production environment. During this stage issues such as that of web application security, the functioning of the site, its access to handicapped as well as regular users and its ability to handle traffic is checked. Web application testing, a software testing technique exclusively adopted to test the applications that are hosted on web in which thTest planning, the most important activity to ensure that there is initially a list of tasks and milestones in a baseline plan to track the progress of the project. It also defines the size of the test effort. It is the main document often called as master test plan or a project test plan and usually developed during the early phase of the project. Here we have some plan activities.

- To determine the scope and the risks that need to be tested and that are NOT to be tested.
- Documenting Test Strategy.
- Making sure that the testing activities have been included.
- Deciding Entry and Exit criteria.
- Evaluating the test estimate.
- Planning when and how to test and deciding how the test results will be evaluated, and defining test exit criterion.
- The Test artefacts delivered as part of test execution.
- Defining the management information, including the metrics required and defect resolution and risk issues.
- Ensuring that the test documentation generates repeatable test assets.

6.2.1 UNIT TESTING

Unit testing, a testing technique using which individual modules are tested to determine if there are any issues by the developer himself. It is concerned with functional correctness of the standalone modules. The main aim is to isolate each unit of the

system to identify, analyze and fix the defects. There are many advantages of Unit Testing.

- Reduces Defects in the newly developed features or reduces bugs when changing the existing functionality.
- Reduces Cost of Testing as defects are captured in very early phase.
- Improves design and allows better refactoring of code.
- Unit Tests, when integrated with build gives the quality of the build as well.

6.2.2 SYSTEM TESTING

System Testing (ST) is a black box testing technique performed to evaluate the complete system the system's compliance against specified requirements. In System testing, the functionalities of the system are tested from an end-to-end perspective.
System Testing is usually carried out by a team that is independent of the development team in order to measure the quality of the system unbiased. It includes both functional and Non-Functional testing.

6.2.3 INTEGRATING TESTING

Data integrity corresponds to the quality of data in the databases and to the level by which users examine data quality, integrity and reliability. Data integrity testing verifies that the data in the database is accurate and functions as expected within a given application. Characteristics of Integrating Testing and given below

- Checking whether or NOT a blank value or default value can be retrieved from the database.
- Validating each value if it is successfully saved to the database.
- Ensuring the data compatibility against old hardware or old versions of operating systems.
- Verifying the data in data tables can be modified and deleted
- Running data tests for all data files, including clip art, tutorials, templates, etc.

6.2.4 USER ACCEPTANCE TESTING

User acceptance testing, a testing methodology where the clients/end users involved in testing the product to validate the product against their requirements. It is performed at client location at developer's site. For industry such as medicine or aviation industry, contract and regulatory compliance testing and operational acceptance testing is also carried out as part of user acceptance testing. UAT is context dependent and the UAT plans are prepared based on the requirements and NOT mandatory to execute all kinds of user acceptance tests and even coordinated and contributed by testing team.

6.3 TEST CASES

A test case is a document, which has a set of test data, preconditions, expected results and post conditions, developed for a particular test scenario in order to verify compliance against a specific requirement. Test Case acts as the starting point for the test execution, and after applying a set of input values, the application has a definitive outcome and leaves the system at some end point or also known as execution post condition.

6.4 RESULTS

Result reporting is a mechanism with which the state of the product is presented to the customer from various angles. Format of the Report varies from time to time as mentioned below:

- Stage of testing in the Development Model.
- Targeted Audience.
- Testing technique adopted.
- Type of testing involved like Functional, Performance/Load/Stress, Disaster recovery, etc.

Test planning, the most important activity to ensure that there is initially a list of tasks and milestones in a baseline plan to track the progress of the project. It also defines the size of the test effort. It is the main document often called as master test plan or a project test plan and usually developed during the early phase of the project. Here we have some plan activities.

- To determine the scope and the risks that need to be tested and that are NOT to be tested.

- Documenting Test Strategy.
- Making sure that the testing activities have been included.
- Deciding Entry and Exit criteria.
- Evaluating the test estimate.
- Planning when and how to test and deciding how the test results will be evaluated, and defining test exit criterion.
- The Test artefacts delivered as part of test execution.
- Defining the management information, including the metrics required and defect resolution and risk issues.
- Ensuring that the test documentation generates repeatable test assets.

6.2.1 UNIT TESTING

Unit testing, a testing technique using which individual modules are tested to determine if there are any issues by the developer himself. It is concerned with functional correctness of the standalone modules. The main aim is to isolate each unit of the system to identify, analyze and fix the defects. There are many advantages of Unit Testing.

- Reduces Defects in the newly developed features or reduces bugs when changing the existing functionality.
- Reduces Cost of Testing as defects are captured in very early phase.
- Improves design and allows better refactoring of code.
- Unit Tests, when integrated with build gives the quality of the build as well.

6.2.2 SYSTEM TESTING

System Testing (ST) is a black box testing technique performed to evaluate the complete system the system's compliance against specified requirements. In System testing, the functionalities of the system are tested from an end-to-end perspective.

System Testing is usually carried out by a team that is independent of the development

team in order to measure the quality of the system unbiased. It includes both functional and Non-Functional testing.

6.2.3 INTEGRATING TESTING

Data integrity corresponds to the quality of data in the databases and to the level by which users examine data quality, integrity and reliability. Data integrity testing verifies that the data in the database is accurate and functions as expected within a given application. Characteristics of Integrating Testing and given below

- Checking whether or NOT a blank value or default value can be retrieved from the database.
- Validating each value if it is successfully saved to the database.
- Ensuring the data compatibility against old hardware or old versions of operating systems.
- Verifying the data in data tables can be modified and deleted
- Running data tests for all data files, including clip art, tutorials, templates, etc.

6.2.4 USER ACCEPTANCE TESTING

User acceptance testing, a testing methodology where the clients/end users involved in testing the product to validate the product against their requirements. It is performed at client location at developer's site. For industry such as medicine or aviation industry, contract and regulatory compliance testing and operational acceptance testing is also carried out as part of user acceptance testing. UAT is context dependent and the UAT plans are prepared based on the requirements and NOT mandatory to execute all kinds of user acceptance tests and even coordinated and contributed by testing team.

6.3 TEST CASES

A test case is a document, which has a set of test data, preconditions, expected results and post conditions, developed for a particular test scenario in order to verify compliance against a specific requirement. Test Case acts as the starting point for the test execution, and after applying a set of input values, the application has a definitive outcome and leaves the system at some end point or also known as execution post condition.

6.4 RESULTS

Result reporting is a mechanism with which the state of the product is presented to the customer from various angles. Format of the Report varies from time to time as mentioned below:

- Stage of testing in the Development Model.
- Targeted Audience.
- Testing technique adopted.
- Type of testing involved like Functional, Performance/Load/Stress, Disaster recovery, etc.

Test Case Title: Log-in

Test case 1: Log-in	**Priority (H, L):** High
Test Objective: For Log-in	
Test Description: "User enters the required fields, presses Log-in button", It must contact with the database, and database updates and sends result to the user.	
Requirements Verified: Yes	
Test Environment: Application must be in running state, Database Should contain appropriate table and connection must be established between application and database.	
Test Setup/Pre-Conditions: Application should be in running state. All the mandatory fields must be entered.	
Actions	**Expected Results**
The admin will Log-in to access application.	"Log-in successfully", Displays Main Menu.
Pass: Yes **Conditions pass: Yes** **Fail**: No	
Problems / Issues: NIL	
Notes: Successfully Executed	

Table 6.1

Test Case Title: Customer Registration

Test case 2: User Registration	**Priority (H, L):** High
Test Objective: For Verifying User Registration	
Test Description: "User selects the required options."	
Requirements Verified: Yes	
Test Environment: Application must be in running state, Database Should contain appropriate table and connection must be established between application and database.	
Test Setup/Pre-Conditions: Application should be in running state. All mandatory fields must be entered.	
Actions	**Expected Results**
The user will register to access application.	"Registered Successfully" Display Menu.
Pass: Yes **Conditions pass: Yes** **Fail**: No	
Problems / Issues: NIL	
Notes: Successfully Executed	

Table 6.2

Test Case Title: View Details

Test case 3: Viewing details	**Priority (H, L):** High
Test Objective: For Viewing Details	
Test Description: "User selects the required option from the Menu; Application connects with the database, database updates and sends result to the user.	
Requirements Verified: Yes	
Test Environment: Application must be in running state, Database Should contain appropriate table and connection must be established between application and database.	
Test Setup/Pre-Conditions: Application should be in running state. The user must have to select one of the options.	
Actions	**Expected Results**
The user will Log-in to access application.	Displays Details.
Pass: Yes **Conditions pass: Yes** **Fail**: No	
Problems / Issues: NIL	
Notes: Successfully Executed	

Table 6.3

Test Case Title: View/Order Guidance stuff

Test case 3: Viewing/Ordering Guidance stuff	**Priority (H, L):** High
Test Objective: For Viewing and Ordering Guidance stuff	
Test Description: "User selects the required option from the Menu; Application connects with the database, database updates and sends result to the user.	
Requirements Verified: Yes	
Test Environment: Application must be in running state, Database Should contain appropriate table and connection must be established between application and database.	
Test Setup/Pre-Conditions: Application should be in running state. The user must have to select one of the options.	
Actions	**Expected Results**
The user will Log-in to access application.	Displays Guidance stuff to View Or Order.
Pass: Yes **Conditions pass: Yes** **Fail**: No	
Problems / Issues: NIL	
Notes: Successfully Executed	

Table 6.4

6.4 REPORT

Reporting test execution results is very important part of testing, whenever test execution cycle is complete, tester should make a complete test results report which includes the Test Pass/Fail status of the test cycle.

If manual testing is done then the test pass/fail result should be captured in an excel sheet and if automation testing is done using automation tool then the HTML or XML reports should be provided to stakeholders as test deliverable.

CHAPTER 7

Conclusion & Future Work

7.1 INTRODUCTION

People's life is getting busier with every passing moment; especially in the metropolitan & surrounding areas. To keep up with the hectic schedule, they need quick, reliable, & anytime-anywhere assistance for various day-to-day tasks.Online guidance ordering & delivery marketplace has been a promising business idea from the start. And it is evident from the success of the first generation of online guidance ordering startups like GuidancePanda, GrubHub, Eat24, and others. However, amid the torrent of new online business ideas, which are sprouting all over the web, the sector hasn't received the attention it deserves from aspiring entrepreneurs. As a result, many parts of it still remain unexplored.

7.2 FUTURE WORK

This system is a bunch of benefits from various point of views. As this online application enables the end users to register to the system online, select the guidance items of their choice from the menu list, and order guidance online. It is developed to help mentors to simplify their daily operational and managerial task as well as improve the dining experience of customers. And also helps restaurant develop healthy customer relationships by providing good services. The system enables staff to let update and make changes to their guidance and beverage list information based on the orders placed and the orders completed.

Worldwide, the market for guidance delivery stands at €83 billion, or 1 percent of the total guidance market and 4 percent of guidance sold through mentors and fast-guidance chains. It has already matured in most countries, with an overall annual growth rate estimated at just

3.5 percent for the next five years. This shows that it is the need of Modern Era.

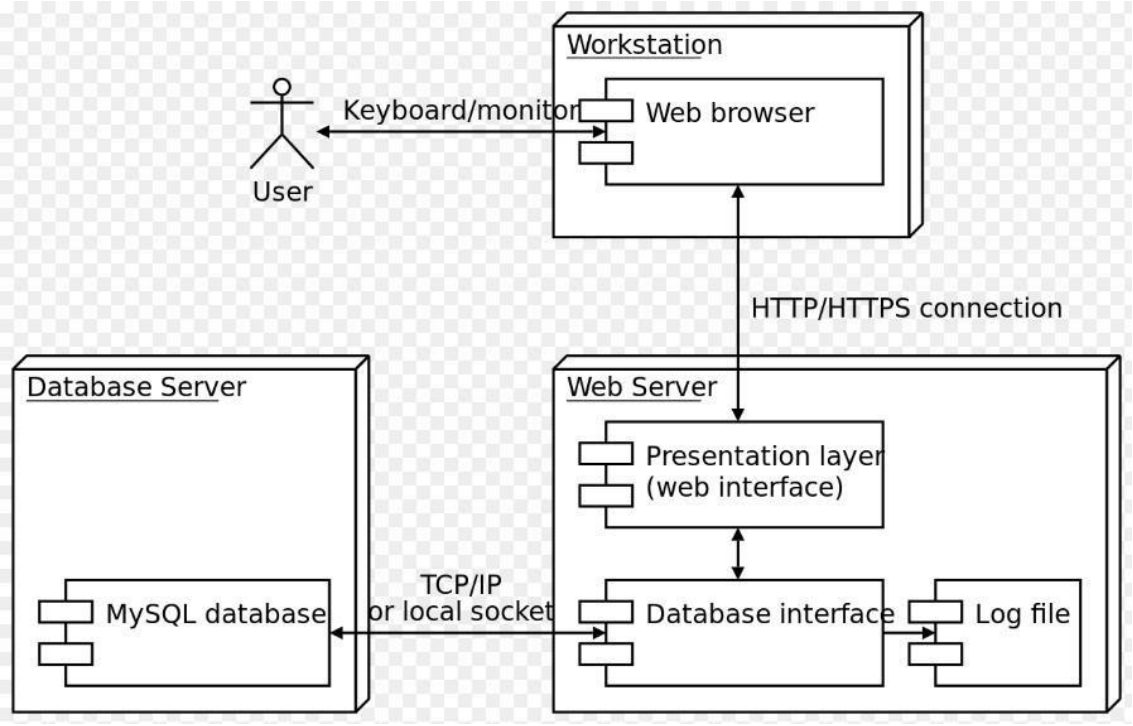
CHAPTER 8

Deployment

8.1 DEPLOYMENT PLAN (DEPLOYMENT DIAGRAM)

A Deployment Design characterizes the succession of activities or steps that must be conveyed to convey changes into an objective framework condition. The individual tasks inside a sending design can be executed physically or consequently. Organization designs are normally all around characterized and affirmed preceding the sending date. In circumstances where there is a high potential danger of disappointment in the objective framework condition, arrangement designs may practiced to guarantee there are no issues amid real organization. Organized repeatable arrangements are likewise prime contender for robotization which drives quality and proficiency. As it is a website so its deployment is not so difficult, the client who is using this just need a standard browser to run it once it can uploaded on web server online by purchasing domain and then it can accessed online. If you want to run locally you have to install WAMP/XAMP first it can show the web you have to use simply a browser to run it locally.

In the Deployment Diagram, user visit the Project Caly to check the deals and order it from the database. The history of deals and it's details are saved in the database server.



8.1 DEPLOYMENT PLAN (DEPLOYMENT DIAGRAM)

IMPLEMENTATION

The implementation stage of software development is the process of converting a system specification into an executable system. It almost always involves processes of software design and programming. In other words, it is a process of converting system requirements into program codes.

REFERENCES

Guidance stuff Reference:

[1] Steven Morris, Carlos Coronel, Peter Rob, Database Systems: Design, Implementation, and Management.

[2] Craig Larman, An Introduction to Object-Oriented Analysis and Design and Iterative Development 3rd Edition.

[3] Pressmen, Somerville, Software Engineering: Design, Implementation, and Management.

[4] Roger S. Pressman, David Lowe, Web Engineering: A Practitioner's Approach.

Websites:

[1] **www.Stackoverflow**

[2] **www.php.com**

[3] **www.w3schools.com**

[4] **www.tutorialspoint.com/index.htm**

[5] **www.w3resource.com**

[6] **www.codeproject.com**

```
    col1, col2 = st.columns(2)
    monthActivitySeries, monthActivity = helper.monthActivity(selectedUser, dataFrame)
    monthActivity = monthActivity.sort_values('message')
    month = monthActivity['monthName']
    messages = monthActivity['message']

    with col2:
        fig, ax = plt.subplots(figsize=(8, 6))
        ax.pie(messages, labels=month, autopct='%1.1f%%', colors=plt.cm.Dark2.colors)
        ax.axis('equal')
        plt.style.use('dark_background')
        st.pyplot(fig)


    with col1:
        fig, ax = plt.subplots(figsize=(8, 6))
        ax.bar(month, messages)
        ax.set_xlabel('Month of the Year', color="yellow")
        ax.set_ylabel('Number of Messages', color='yellow')
        plt.style.use('dark_background')
        st.pyplot(fig)

    #hourly activity
    st.title("Hour Activity")
    h1, h2 = helper.hourActivity(selectedUser, dataFrame)

    fig, ax = plt.subplots(figsize=(12, 3))
    h1.unstack('dayName').plot(ax=ax)
    ax.set_xlabel('Hour of the Day', color='yellow')
    ax.set_ylabel('Number of Messages', color='yellow')
    ax.set_title('Messages Sent by Hour of the Day', color='white')
    plt.style.use('dark_background')
    st.pyplot(fig)

    #----
    st.header("Day-wise Activity    ")
    tabs = st.multiselect("Select day(s) to display",['Monday', 'Tuesday', 'Wednesday', 'Thursday',
'Friday', 'Saturday', 'Sunday'])
    #tab1, tab2, tab3, tab4, tab5, tab6, tab7 = st.tabs(['Monday', 'Tuesday', 'Wednesday', 'Thursday',
'Friday', 'Saturday', 'Sunday'])
    days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']

    for day in tabs:
        day_data = h2[h2['dayName'] == day]
        plot_placeholder = st.empty()
        with plot_placeholder:
            fig, axs = plt.subplots(figsize=(12, 3))
            axs.plot(day_data['hour'], day_data['message'])
            axs.set_title(day)
            axs.set_xlabel('Hour of the Day', color='yellow')
            axs.set_ylabel('Number of Messages', color='yellow')
            axs.set_xticks(range(0, 24, 2))
            axs.grid(True, alpha=0.3)
            plt.tight_layout()
            st.pyplot(fig)

    #period activity
    st.header("Activity by Time Period")
```

```
activity = helper.activity(selectedUser, dataFrame)
activity = activity.sort_values('message')
period = activity['period']
messages = activity['message']
fig, ax = plt.subplots(figsize=(16, 3))
ax.bar(period, messages)
ax.set_xlabel('Period', color="yellow")
ax.set_ylabel('Number of Messages', color='yellow')
ax.set_title('Activity Chart')
plt.style.use('dark_background')
st.pyplot(fig)



# finding busiest users in the group
if selectedUser == 'Overall':
    st.header("Top Chatters��    ")
    topChatter, topChatterPercent = helper.mostBusy(dataFrame)
    col1, col2 = st.columns(2)

    with col1:
        plt.style.use('dark_background')
        name = topChatter.index
        name = [emoji.emojize(n) for n in name]
        count = topChatter.values
        fig, ax = plt.subplots()
        plt.xlabel('Name').set_color('yellow')
        plt.ylabel('Messages Sent').set_color('yellow')
        ax.bar(name, count, width=0.8)
        plt.xticks(rotation='vertical')
        ax.tick_params(axis='both', which='major', labelsize=8)

        st.pyplot(fig)

    with col2:
        st.dataframe(topChatterPercent)




# most common words

mostCommon = helper.mostCommon(selectedUser, dataFrame)
if (mostCommon.shape[0] != 0):
    st.header("Top Words Used��")

    col1, col2 = st.columns(2)
    with col1:
        fig, ax = plt.subplots()
        plt.ylabel('Message').set_color('yellow')
        plt.xlabel('Frequency').set_color('yellow')
        ax.barh(mostCommon['Message'], mostCommon['Frequency'])
        plt.xticks(rotation="vertical")
        st.pyplot(fig)

    with col2:
        st.dataframe(mostCommon)
```

```
        # emoji analysis
        emoji_df = helper.mostEmoji(selectedUser, dataFrame)
        if (emoji_df.shape[0] != 0):
            st.title("Emoji Analysis��")

            col1, col2 = st.columns(2)

            with col1:
                st.dataframe(emoji_df)
            with col2:
                fig, ax = plt.subplots()
                color = ['#FFC107', '#2196F3', '#4CAF50', '#F44336', '#9C27B0']

                ax.pie(emoji_df['Count'].head(), labels=emoji_df['Emoji'].head(
                ), autopct="%0.2f", colors=color)
                ax.set_title("Emoji Distribution", color='yellow')
                fig.set_facecolor('#121212')
                st.pyplot(fig)

        #message extractor
        st.title("Messages Extractor��")
        inputDate = st.text_input("Enter date in format : 19-08-2003")
        messageExtract = helper.messageExtractor(selectedUser, dataFrame, inputDate)
        if st.button("Extract"):
            if messageExtract.shape[0]>0:
                st.dataframe(messageExtract, width=1400)
            else:
                st.write("No conversation(s) on", inputDate)

        #reply time analysis
        st.header("Reply Time Analysis    ")
        timeDifference, timeSelected = helper.replyTime(selectedUser, dataFrame)
        if (selectedUser!='Overall'):
            st.write("Average Reply Time by", selectedUser, "is", timeSelected)
        else:
            col1, col2 = st.columns(2)
            with col1:
                fig, ax = plt.subplots(figsize=(8, 6))
                ax.bar(timeDifference['user'], timeDifference['replyTime'].dt.seconds)
                ax.set_xlabel('Participant', color='yellow')
                ax.set_ylabel('Average Reply Time (Seconds)', color='yellow')
                ax.set_title('')
                st.pyplot(plt)
            with col2:
                fig, ax = plt.subplots(figsize=(8, 6))
                ax.pie(timeDifference['replyTime'], labels=timeDifference['user'], autopct='%1.1f%%',
colors=plt.cm.Dark2.colors)
                ax.axis('equal')
                plt.style.use('dark_background')
                ax.set_title('')
                st.pyplot(fig)
```

helper.py

```
from urlextract import URLExtract
import pandas as pd
from collections import Counter
import emoji
import re
import regex
import streamlit as st
from datetime import datetime


def fetchStats(selectedUser, dataFrame):
    if selectedUser != "Overall":
        dataFrame = dataFrame[dataFrame['user'] == selectedUser]
    totalMessages = dataFrame.shape[0]

    word = []
    for message in dataFrame['message']:
        if isinstance(message, str):
            word.extend(message.split())
    totalWords = len(word)

    totalMedia = dataFrame[dataFrame['message']
                           == '<Media omitted>\n'].shape[0]

    extractor = URLExtract()
    urls = extractor.find_urls(" ".join(word))
    totalURL = len(urls)

    return totalMessages, totalWords, totalMedia, totalURL


def mostBusy(x):
    topChatter = x['user'].value_counts().head()
    topChatterPercent = round((x['user'].value_counts(
    )/x.shape[0])*100, 2).reset_index().rename(columns={'index': "Name", 'user': 'Percentage'})

    return topChatter, topChatterPercent


def mostCommon(selectedUser, x):
    if selectedUser != "Overall":
        x = x[x['user'] == selectedUser]
    # remove stopwords and group notifications
    withoutGN = x[x['user'] != 'default']
    withoutGNMedia = withoutGN[withoutGN["message"] != '<Media omitted>\n']

    stopWords = open("stopwords-hinglish.txt", "r").read()

    words = []

    for message in withoutGNMedia['message']:
        if isinstance(message, str):
            for word in message.lower().split():
                if word not in stopWords:
                    words.append(word)
```

```
    mC = Counter(words).most_common(20)
    mostCommon = pd.DataFrame(mC)
    mostCommon = mostCommon.rename(columns={0: 'Message', 1: 'Frequency'})

    return mostCommon


def mostEmoji(selectedUser, x):
    if selectedUser != 'Overall':
        x = x[x['user'] == selectedUser]
    emojis = []
    for message in x['message']:
        if isinstance(message, str):
            message_emojized = emoji.emojize(message, language='alias')
            emojis.extend(
                [c for c in message_emojized if c in emoji.UNICODE_EMOJI['en']])

    emoji_counts = Counter(emojis)
    emoji_df = pd.DataFrame(list(emoji_counts.items()),
                            columns=['Emoji', 'Count'])
    emoji_df['Emoji'] = emoji_df['Emoji'].apply(
        lambda x: emoji.emojize(x, language='alias'))
    emoji_df = emoji_df.sort_values(
        'Count', ascending=False).reset_index(drop=True)

    return emoji_df


def monthlyTimeline(selectedUser, x):
    if selectedUser != "Overall":
        x = x[x['user'] == selectedUser]
    timeline = x.groupby(['year', 'monthNum', 'month']).count()[
        'message'].reset_index()

    time = []
    for i in range(timeline.shape[0]):
        time.append(timeline['month'][i] + "-" + str(timeline['year'][i]))
    timeline['time'] = time
    return timeline


def dailyTimeline(selectedUser, x):
    if selectedUser != "Overall":
        x = x[x['user'] == selectedUser]
    x['onlyDate'] = pd.to_datetime(x['date']).dt.date
    dailyTimeline = x.groupby("onlyDate").count()['message'].reset_index()
    return dailyTimeline


def weekActivity(selectedUser, x):
    if selectedUser != "Overall":
        x = x[x['user'] == selectedUser]
    weekActivity = x.groupby("dayName").count()['message'].reset_index()
    return x['dayName'].value_counts(), weekActivity


def monthActivity(selectedUser, x):
    if selectedUser != "Overall":
```

```
        x = x[x['user'] == selectedUser]
    monthActivity = x.groupby("monthName").count()['message'].reset_index()
    return x['monthName'].value_counts(), monthActivity


def hourActivity(selectedUser, x):
    if selectedUser != "Overall":
        x = x[x['user'] == selectedUser]
    return x.groupby(['dayName', 'hour'])['message'].count(), x.groupby(['dayName',
'hour'])['message'].count().reset_index()



def messageExtractor (selectedUser, x, inputDate):
    #inputDate = "20-04-2023"
    if selectedUser != "Overall":
        x = x[x['user'] == selectedUser]
    if (len(inputDate)==10):
        dd = inputDate[0:2]
        mm = inputDate[3:5]
        yyyy = inputDate[6:]
        if (dd[0]=='0'): dd = dd[1]
        if (mm[0]=='0'): mm = mm[1]
        mask = (x['day'].astype(str) == dd) & (x['monthNum'].astype(str) == mm) & (x['year'].astype(str)
== yyyy)
        messageExtract = pd.DataFrame(x[mask])[['user', 'message']]
        if (messageExtract.shape[0]>0):
            messageExtract['time'] = x['hour'].astype(str) + ':' + x['minute'].astype(str)
            messageExtract['message'] = messageExtract['message'].str.replace('\n', '')
        #st.dataframe(messageExtract)

        return messageExtract

def activity (selectedUser, x):
    if selectedUser != "Overall":
        x = x[x['user'] == selectedUser]
    activityX = x.groupby("period").count()['message'].reset_index()
    return activityX

def replyTime (selectedUser, x):
    timeSelected = pd.Timedelta(0)
    timeDifference = x.groupby('user')['replyTime'].mean().reset_index().sort_values('replyTime',
ascending=True).head(5)
    timeDifference = timeDifference[timeDifference['user'] != 'default']
    if selectedUser != "Overall":
        x = x[x['user'] == selectedUser]
        timeSelected = timeDifference[timeDifference['user'] == selectedUser]['replyTime'].iloc[0]

    return timeDifference, timeSelected
```

Appendix B

Screen shots

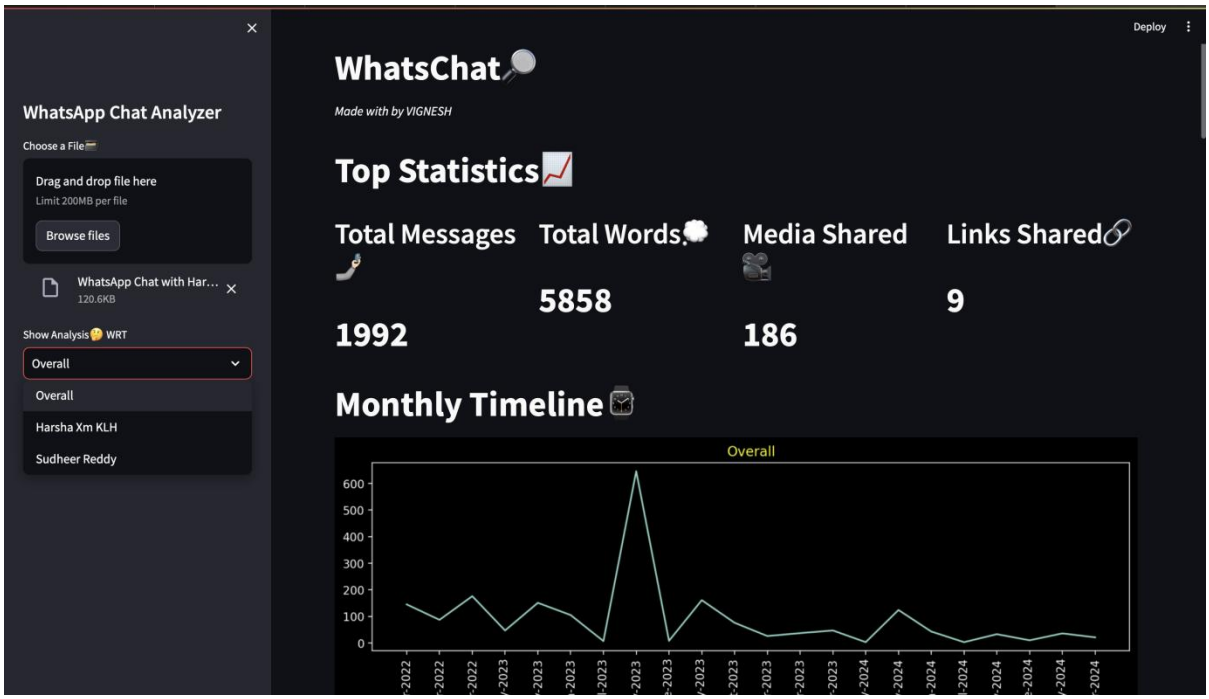


Figure 1.1

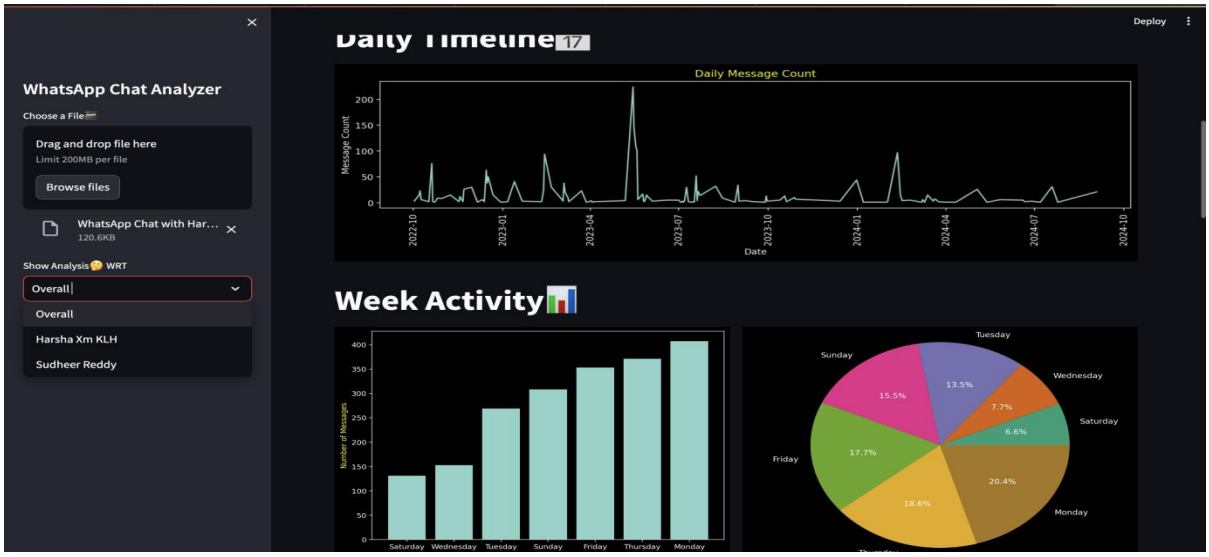


Figure 1.2

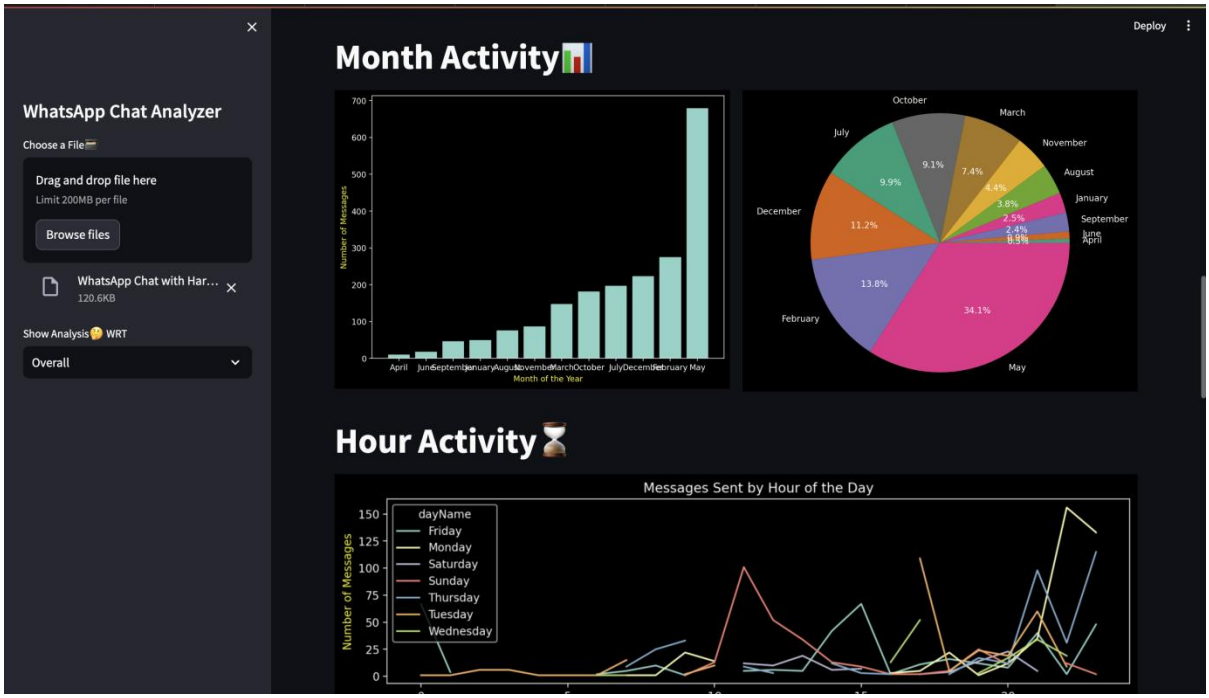


Figure 1.3

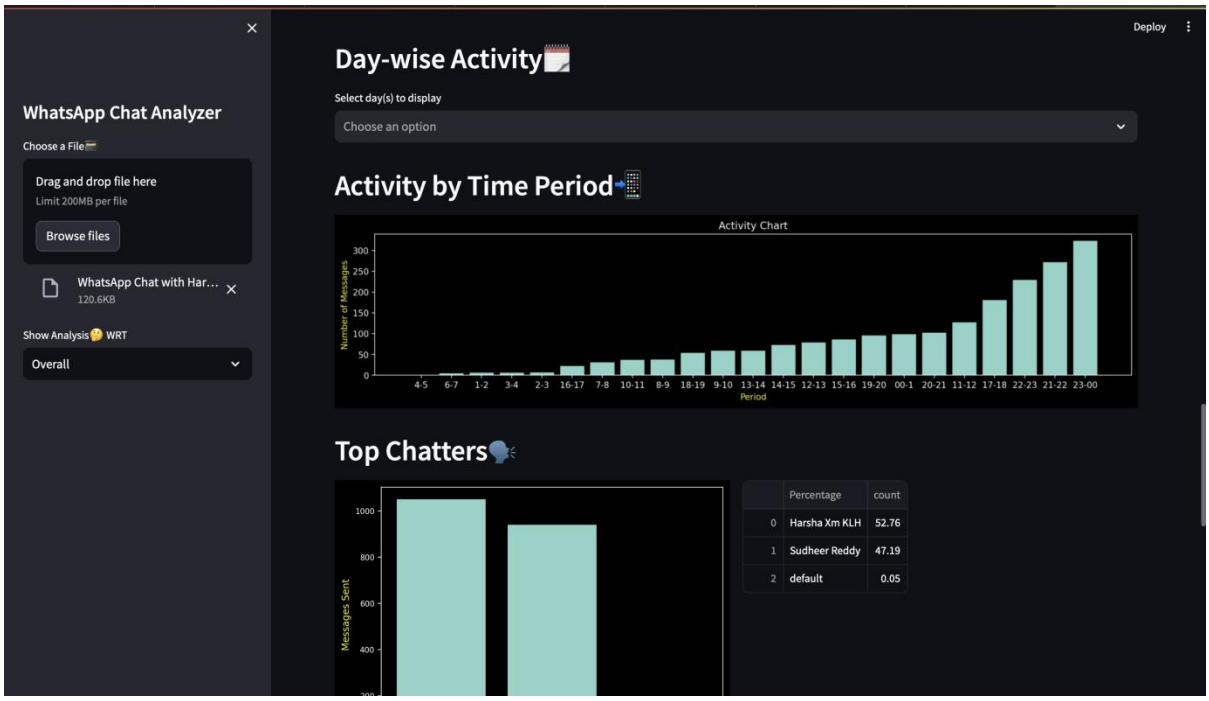


Figure 1.4

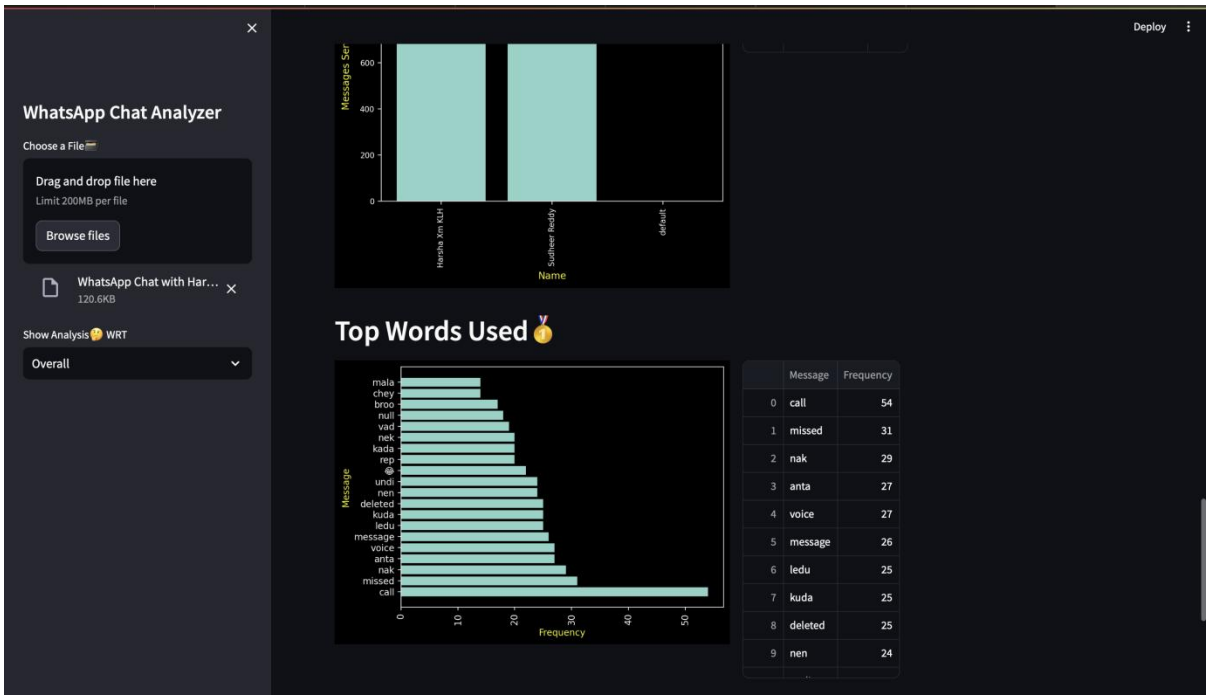

WhatsApp Chat Analyzer
Choose a File
Drag and drop file here
Limit 200MB per file
Browse files
WhatsApp Chat with Har...
120.6KB
Show Analysis WRT
Overall
Deploy
Messages Ser
600
400
200
0
Harsha Xm KLH
Sudheer Reddy
default
Name
Top Words Used
Message
mala
chey
broo
null
vad
nek
kada
rep
undi
nen
deleted
kuda
ledu
message
voice
anta
nak
missed
call
0
10
20
30
40
50
Frequency
Message
Frequency
0 call 54
1 missed 31
2 nak 29
3 anta 27
4 voice 27
5 message 26
6 ledu 25
7 kuda 25
8 deleted 25
9 nen 24

Figure 1.5

Appendix C

Data sets used in the project

1. WhatsApp Chat Export Files

- Raw chat data exported from WhatsApp in TXT or CSV format, including sender names, timestamps, and message content.

2. Sample Group Chats

- Data from group chats used to analyze participant engagement, message volume, and active hours.

3. Response Time Data

- Time differences between consecutive messages to calculate average response times for participants.

4. Sentiment Data

- Sentiment analysis of messages to determine overall conversation tone (positive, negative, neutral.

5.Public WhatsApp Chat Datasets (for Validation)

- Pre-validated, publicly available chat datasets (e.g., from Kaggle) used for testing and validating the tool's accuracy.

These datasets formed the core foundation for analyzing and generating insights in the WhatsApp Chat Analyzer project.

```
    col1, col2 = st.columns(2)
    monthActivitySeries, monthActivity = helper.monthActivity(selectedUser, dataFrame)
    monthActivity = monthActivity.sort_values('message')
    month = monthActivity['monthName']
    messages = monthActivity['message']

    with col2:
        fig, ax = plt.subplots(figsize=(8, 6))
        ax.pie(messages, labels=month, autopct='%1.1f%%', colors=plt.cm.Dark2.colors)
        ax.axis('equal')
        plt.style.use('dark_background')
        st.pyplot(fig)


    with col1:
        fig, ax = plt.subplots(figsize=(8, 6))
        ax.bar(month, messages)
        ax.set_xlabel('Month of the Year', color="yellow")
        ax.set_ylabel('Number of Messages', color='yellow')
        plt.style.use('dark_background')
        st.pyplot(fig)

    #hourly activity
    st.title("Hour Activity")
    h1, h2 = helper.hourActivity(selectedUser, dataFrame)

    fig, ax = plt.subplots(figsize=(12, 3))
    h1.unstack('dayName').plot(ax=ax)
    ax.set_xlabel('Hour of the Day', color='yellow')
    ax.set_ylabel('Number of Messages', color='yellow')
    ax.set_title('Messages Sent by Hour of the Day', color='white')
    plt.style.use('dark_background')
    st.pyplot(fig)

    #----
    st.header("Day-wise Activity    ")
    tabs = st.multiselect("Select day(s) to display",['Monday', 'Tuesday', 'Wednesday', 'Thursday',
'Friday', 'Saturday', 'Sunday'])
    #tab1, tab2, tab3, tab4, tab5, tab6, tab7 = st.tabs(['Monday', 'Tuesday', 'Wednesday', 'Thursday',
'Friday', 'Saturday', 'Sunday'])
    days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']

    for day in tabs:
        day_data = h2[h2['dayName'] == day]
        plot_placeholder = st.empty()
        with plot_placeholder:
            fig, axs = plt.subplots(figsize=(12, 3))
            axs.plot(day_data['hour'], day_data['message'])
            axs.set_title(day)
            axs.set_xlabel('Hour of the Day', color='yellow')
            axs.set_ylabel('Number of Messages', color='yellow')
            axs.set_xticks(range(0, 24, 2))
            axs.grid(True, alpha=0.3)
            plt.tight_layout()
            st.pyplot(fig)

    #period activity
    st.header("Activity by Time Period")
```

```
activity = helper.activity(selectedUser, dataFrame)
activity = activity.sort_values('message')
period = activity['period']
messages = activity['message']
fig, ax = plt.subplots(figsize=(16, 3))
ax.bar(period, messages)
ax.set_xlabel('Period', color="yellow")
ax.set_ylabel('Number of Messages', color='yellow')
ax.set_title('Activity Chart')
plt.style.use('dark_background')
st.pyplot(fig)



# finding busiest users in the group
if selectedUser == 'Overall':
    st.header("Top Chatters��    ")
    topChatter, topChatterPercent = helper.mostBusy(dataFrame)
    col1, col2 = st.columns(2)

    with col1:
        plt.style.use('dark_background')
        name = topChatter.index
        name = [emoji.emojize(n) for n in name]
        count = topChatter.values
        fig, ax = plt.subplots()
        plt.xlabel('Name').set_color('yellow')
        plt.ylabel('Messages Sent').set_color('yellow')
        ax.bar(name, count, width=0.8)
        plt.xticks(rotation='vertical')
        ax.tick_params(axis='both', which='major', labelsize=8)

        st.pyplot(fig)

    with col2:
        st.dataframe(topChatterPercent)




# most common words

mostCommon = helper.mostCommon(selectedUser, dataFrame)
if (mostCommon.shape[0] != 0):
    st.header("Top Words Used��")

    col1, col2 = st.columns(2)
    with col1:
        fig, ax = plt.subplots()
        plt.ylabel('Message').set_color('yellow')
        plt.xlabel('Frequency').set_color('yellow')
        ax.barh(mostCommon['Message'], mostCommon['Frequency'])
        plt.xticks(rotation="vertical")
        st.pyplot(fig)

    with col2:
        st.dataframe(mostCommon)
```

```
        # emoji analysis
        emoji_df = helper.mostEmoji(selectedUser, dataFrame)
        if (emoji_df.shape[0] != 0):
            st.title("Emoji Analysis��")

            col1, col2 = st.columns(2)

            with col1:
                st.dataframe(emoji_df)
            with col2:
                fig, ax = plt.subplots()
                color = ['#FFC107', '#2196F3', '#4CAF50', '#F44336', '#9C27B0']

                ax.pie(emoji_df['Count'].head(), labels=emoji_df['Emoji'].head(
                ), autopct="%0.2f", colors=color)
                ax.set_title("Emoji Distribution", color='yellow')
                fig.set_facecolor('#121212')
                st.pyplot(fig)

        #message extractor
        st.title("Messages Extractor��")
        inputDate = st.text_input("Enter date in format : 19-08-2003")
        messageExtract = helper.messageExtractor(selectedUser, dataFrame, inputDate)
        if st.button("Extract"):
            if messageExtract.shape[0]>0:
                st.dataframe(messageExtract, width=1400)
            else:
                st.write("No conversation(s) on", inputDate)

        #reply time analysis
        st.header("Reply Time Analysis    ")
        timeDifference, timeSelected = helper.replyTime(selectedUser, dataFrame)
        if (selectedUser!='Overall'):
            st.write("Average Reply Time by", selectedUser, "is", timeSelected)
        else:
            col1, col2 = st.columns(2)
            with col1:
                fig, ax = plt.subplots(figsize=(8, 6))
                ax.bar(timeDifference['user'], timeDifference['replyTime'].dt.seconds)
                ax.set_xlabel('Participant', color='yellow')
                ax.set_ylabel('Average Reply Time (Seconds)', color='yellow')
                ax.set_title('')
                st.pyplot(plt)
            with col2:
                fig, ax = plt.subplots(figsize=(8, 6))
                ax.pie(timeDifference['replyTime'], labels=timeDifference['user'], autopct='%1.1f%%',
colors=plt.cm.Dark2.colors)
                ax.axis('equal')
                plt.style.use('dark_background')
                ax.set_title('')
                st.pyplot(fig)
```

helper.py

```
from urlextract import URLExtract
import pandas as pd
from collections import Counter
import emoji
import re
import regex
import streamlit as st
from datetime import datetime


def fetchStats(selectedUser, dataFrame):
    if selectedUser != "Overall":
        dataFrame = dataFrame[dataFrame['user'] == selectedUser]
    totalMessages = dataFrame.shape[0]

    word = []
    for message in dataFrame['message']:
        if isinstance(message, str):
            word.extend(message.split())
    totalWords = len(word)

    totalMedia = dataFrame[dataFrame['message']
                           == '<Media omitted>\n'].shape[0]

    extractor = URLExtract()
    urls = extractor.find_urls(" ".join(word))
    totalURL = len(urls)

    return totalMessages, totalWords, totalMedia, totalURL


def mostBusy(x):
    topChatter = x['user'].value_counts().head()
    topChatterPercent = round((x['user'].value_counts(
    )/x.shape[0])*100, 2).reset_index().rename(columns={'index': "Name", 'user': 'Percentage'})

    return topChatter, topChatterPercent


def mostCommon(selectedUser, x):
    if selectedUser != "Overall":
        x = x[x['user'] == selectedUser]
    # remove stopwords and group notifications
    withoutGN = x[x['user'] != 'default']
    withoutGNMedia = withoutGN[withoutGN["message"] != '<Media omitted>\n']

    stopWords = open("stopwords-hinglish.txt", "r").read()

    words = []

    for message in withoutGNMedia['message']:
        if isinstance(message, str):
            for word in message.lower().split():
                if word not in stopWords:
                    words.append(word)
```

```
    mC = Counter(words).most_common(20)
    mostCommon = pd.DataFrame(mC)
    mostCommon = mostCommon.rename(columns={0: 'Message', 1: 'Frequency'})

    return mostCommon


def mostEmoji(selectedUser, x):
    if selectedUser != 'Overall':
        x = x[x['user'] == selectedUser]
    emojis = []
    for message in x['message']:
        if isinstance(message, str):
            message_emojized = emoji.emojize(message, language='alias')
            emojis.extend(
                [c for c in message_emojized if c in emoji.UNICODE_EMOJI['en']])

    emoji_counts = Counter(emojis)
    emoji_df = pd.DataFrame(list(emoji_counts.items()),
                            columns=['Emoji', 'Count'])
    emoji_df['Emoji'] = emoji_df['Emoji'].apply(
        lambda x: emoji.emojize(x, language='alias'))
    emoji_df = emoji_df.sort_values(
        'Count', ascending=False).reset_index(drop=True)

    return emoji_df


def monthlyTimeline(selectedUser, x):
    if selectedUser != "Overall":
        x = x[x['user'] == selectedUser]
    timeline = x.groupby(['year', 'monthNum', 'month']).count()[
        'message'].reset_index()

    time = []
    for i in range(timeline.shape[0]):
        time.append(timeline['month'][i] + "-" + str(timeline['year'][i]))
    timeline['time'] = time
    return timeline


def dailyTimeline(selectedUser, x):
    if selectedUser != "Overall":
        x = x[x['user'] == selectedUser]
    x['onlyDate'] = pd.to_datetime(x['date']).dt.date
    dailyTimeline = x.groupby("onlyDate").count()['message'].reset_index()
    return dailyTimeline


def weekActivity(selectedUser, x):
    if selectedUser != "Overall":
        x = x[x['user'] == selectedUser]
    weekActivity = x.groupby("dayName").count()['message'].reset_index()
    return x['dayName'].value_counts(), weekActivity


def monthActivity(selectedUser, x):
    if selectedUser != "Overall":
```

```
        x = x[x['user'] == selectedUser]
    monthActivity = x.groupby("monthName").count()['message'].reset_index()
    return x['monthName'].value_counts(), monthActivity


def hourActivity(selectedUser, x):
    if selectedUser != "Overall":
        x = x[x['user'] == selectedUser]
    return x.groupby(['dayName', 'hour'])['message'].count(), x.groupby(['dayName',
'hour'])['message'].count().reset_index()



def messageExtractor (selectedUser, x, inputDate):
    #inputDate = "20-04-2023"
    if selectedUser != "Overall":
        x = x[x['user'] == selectedUser]
    if (len(inputDate)==10):
        dd = inputDate[0:2]
        mm = inputDate[3:5]
        yyyy = inputDate[6:]
        if (dd[0]=='0'): dd = dd[1]
        if (mm[0]=='0'): mm = mm[1]
        mask = (x['day'].astype(str) == dd) & (x['monthNum'].astype(str) == mm) & (x['year'].astype(str)
== yyyy)
        messageExtract = pd.DataFrame(x[mask])[['user', 'message']]
        if (messageExtract.shape[0]>0):
            messageExtract['time'] = x['hour'].astype(str) + ':' + x['minute'].astype(str)
            messageExtract['message'] = messageExtract['message'].str.replace('\n', '')
        #st.dataframe(messageExtract)

        return messageExtract

def activity (selectedUser, x):
    if selectedUser != "Overall":
        x = x[x['user'] == selectedUser]
    activityX = x.groupby("period").count()['message'].reset_index()
    return activityX

def replyTime (selectedUser, x):
    timeSelected = pd.Timedelta(0)
    timeDifference = x.groupby('user')['replyTime'].mean().reset_index().sort_values('replyTime',
ascending=True).head(5)
    timeDifference = timeDifference[timeDifference['user'] != 'default']
    if selectedUser != "Overall":
        x = x[x['user'] == selectedUser]
        timeSelected = timeDifference[timeDifference['user'] == selectedUser]['replyTime'].iloc[0]

    return timeDifference, timeSelected
```

Appendix B

Screen shots

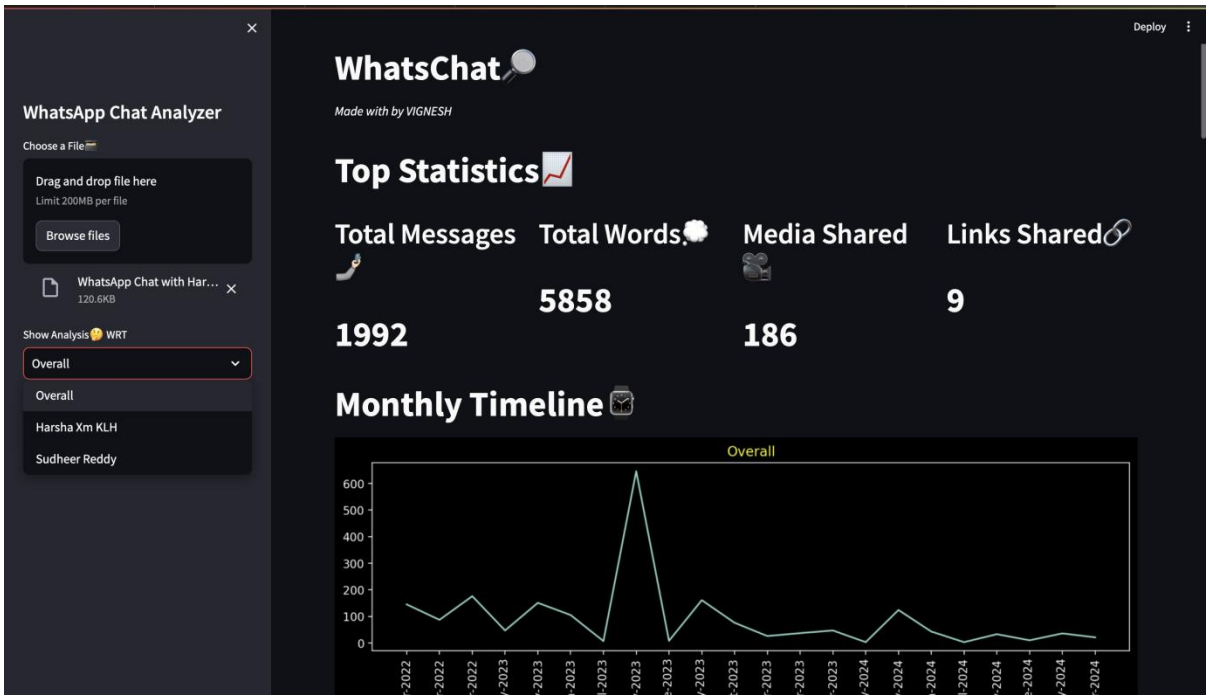


Figure 1.1

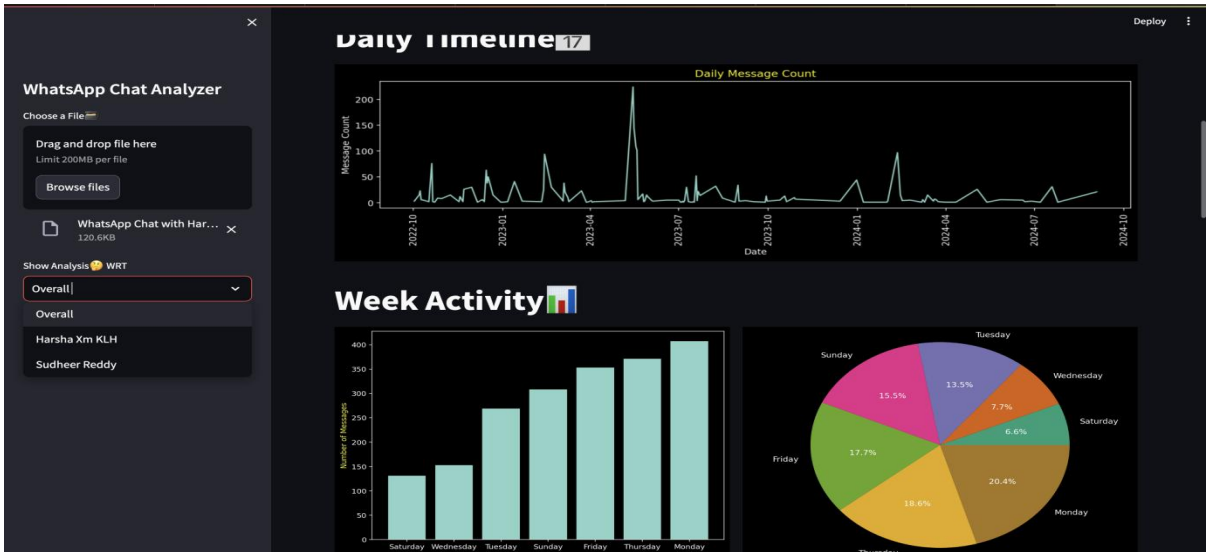


Figure 1.2

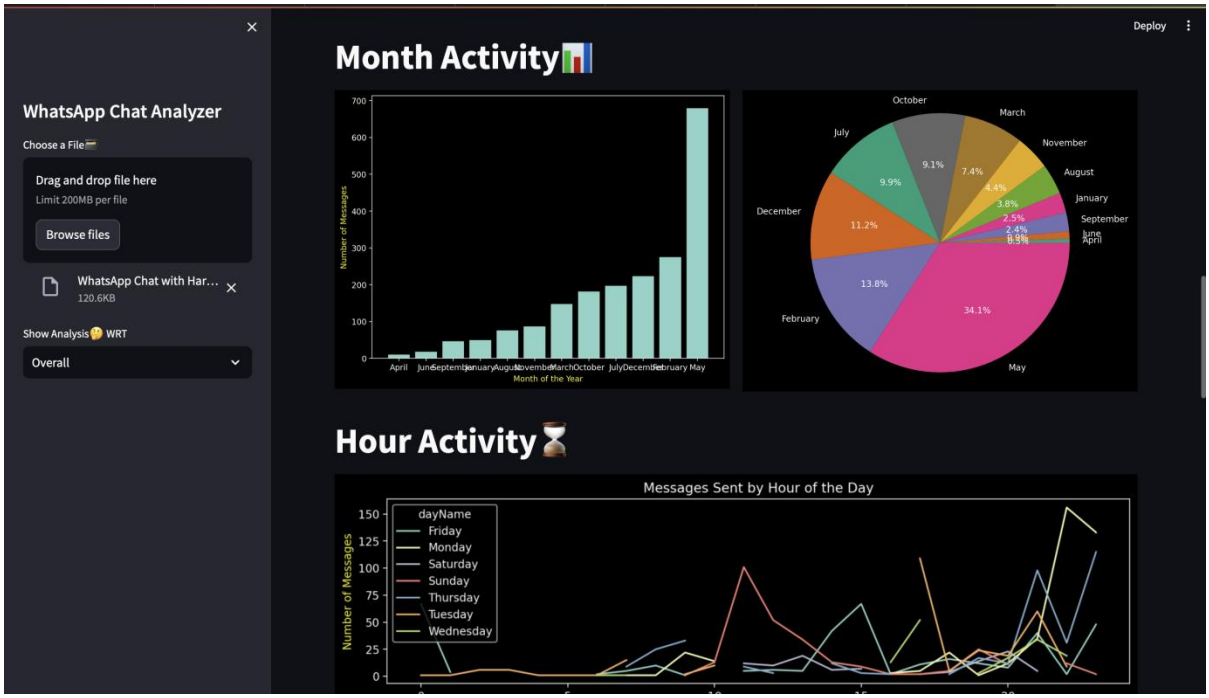


Figure 1.3

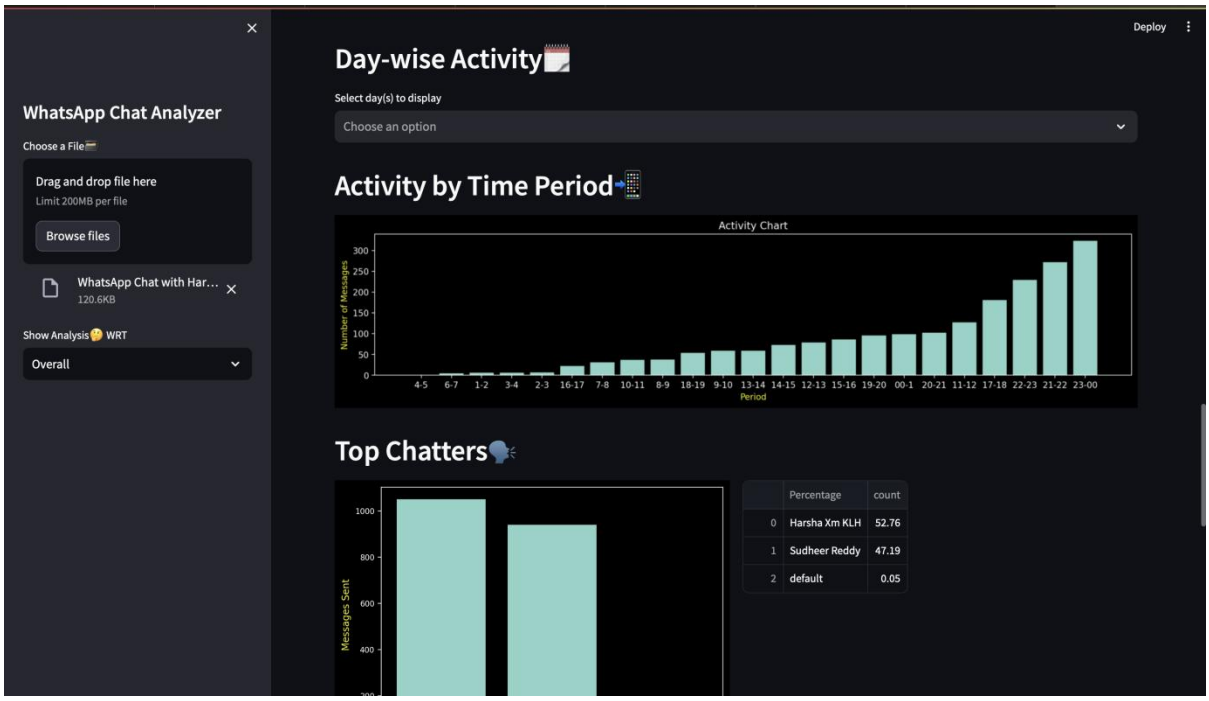


Figure 1.4

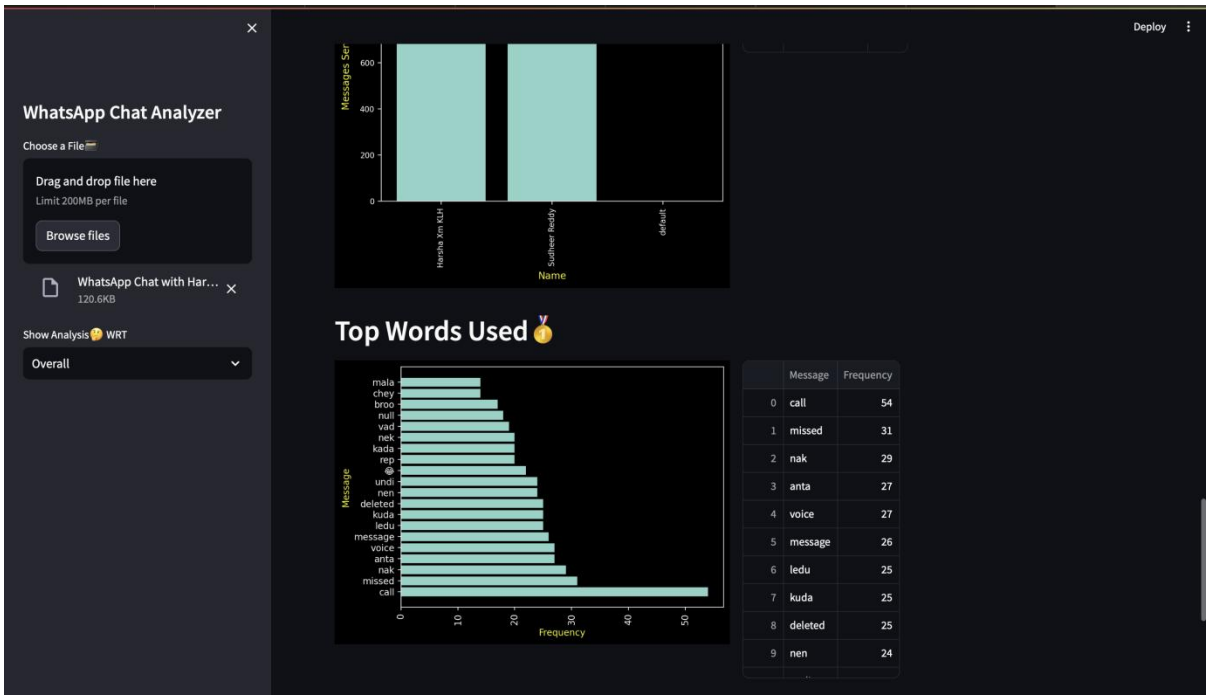

WhatsApp Chat Analyzer
Choose a File
Drag and drop file here
Limit 200MB per file
Browse files
WhatsApp Chat with Har...
120.6KB
Show Analysis WRT
Overall
Deploy
Messages Ser
600
400
200
0
Harsha Xm KLH
Sudheer Reddy
default
Name
Top Words Used
Message
mala
chey
broo
null
vad
nek
kada
rep
undi
nen
deleted
kuda
ledu
message
voice
anta
nak
missed
call
0
10
20
30
40
50
Frequency
Message
Frequency
0 call 54
1 missed 31
2 nak 29
3 anta 27
4 voice 27
5 message 26
6 ledu 25
7 kuda 25
8 deleted 25
9 nen 24

Figure 1.5

Appendix C

Data sets used in the project

1. WhatsApp Chat Export Files

- Raw chat data exported from WhatsApp in TXT or CSV format, including sender names, timestamps, and message content.

2. Sample Group Chats

- Data from group chats used to analyze participant engagement, message volume, and active hours.

3. Response Time Data

- Time differences between consecutive messages to calculate average response times for participants.

4. Sentiment Data

- Sentiment analysis of messages to determine overall conversation tone (positive, negative, neutral.

5.Public WhatsApp Chat Datasets (for Validation)

- Pre-validated, publicly available chat datasets (e.g., from Kaggle) used for testing and validating the tool's accuracy.

These datasets formed the core foundation for analyzing and generating insights in the WhatsApp Chat Analyzer project.