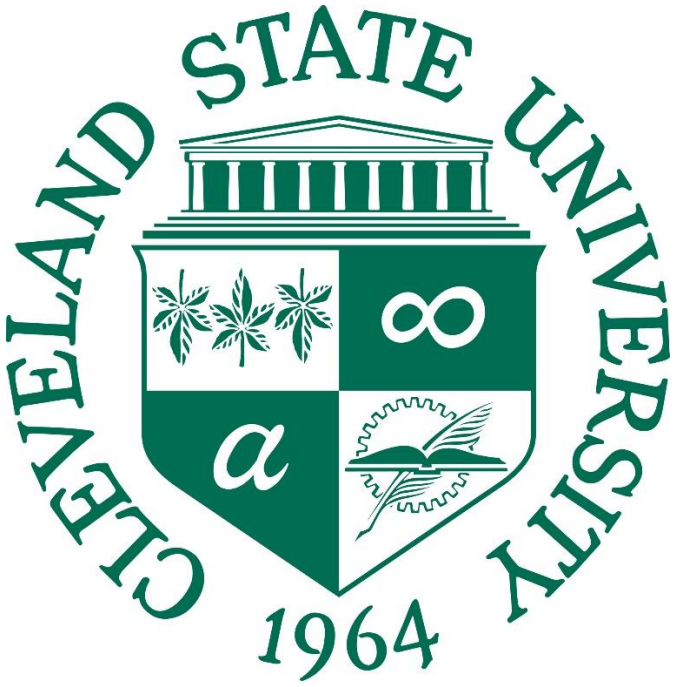
CLEVELAND STATE UNIVERSITY



CIS 694 – Object-oriented Software Engineer

SOFTWARE DESIGN SPECIFICATION

Project: Restaurant Management System

Presented By:
Group 5
Bhavana Tedlapalli (2808568)
Sravan Kumar Singupuram (2836831)
Riya Patel (2829317)
Calvin Raj Namburi (2836250)

1.0 Introduction

The goal of the design phase is to translate the requirements into conceptual solutions that could serve as a starting point for the development of software. This chapter aims to define the design requirements, data, architectural, interface, component-level design and offer design solutions. It begins by identifying the agile development design principles and incorporating them into the core of design activity. The project will then use architectural design to generate a broad vision for the evolving system. It moves on to system modeling, which involves creating design models to comprehend the limitations and characteristics of the system. Finally, effective user interface design methods are covered.

1.1 Goals and objectives

Object-oriented design strategies were employed for this development. This approach has benefits including code reuse, advantageous design, and easily maintainable facilities, classes, and objects.
Maximizing the profit is one of main objectives of any business. This can achieve by increasing efficiency and decreasing overheads without compromising customer satisfaction. Through better application of daily operations restaurant can increase the efficiency and can offer improved services to the customers. Because almost all processes are manual and time consuming, all the processes should be automated.

1.2 Statement of scope

The goal of the project was to create a computerized system for Silk Route that would handle billing restaurant records while allowing staff who were involved in the customer billing process access to the records as needed. To protect the confidentiality and security of the records, the employees are only given limited access. The database is kept up to date during the project.

1.3 Software context

The software is placed in a business or product line context. Strategic issues relevant to context are discussed. The intent is for the reader to understand the 'big picture'.
Creating a web-based restaurant management system where the customers can order the items, update, and cancel orders, staff can manage menus, discounts, display items and control inventory. Generating essential reports throughout the process. Providing detailed user manual.

1.4 Major constraints

- The system should provide a user-friendly environment including flexible interfaces.
- Person with average computer skills can work with the system within a short period of training.
- The system should be accurate and consistent, when manipulating the fed data in proper way and displaying correct information.

- The system should keep up security and reliability because the system handles important data related to business processes of the company.
- Occasionally backups should be taken to maintain reliability. The system should be reusable and maintainable.

2.0 Data design

Unified Modelling Language (UML) plays a significant role in Object Orient designing. UML allows programmers easy understand models of objects so that programmers can easily write software. Some of structural and behavioral UML diagrams use for design proposes systems are mentioned below.

- Use case diagrams – This makes clear what are the capabilities of the system.
- Use case narratives – Make use cases clearer.
- Activity diagrams – Shows how activities are coordinated to provide a service which can be at different levels of abstraction.
- DFD's – Data flow diagrams representing a flow of data through a process or a system
- Data flow Diagrams – Shows how the objects interact overtime.
- Class diagrams – This is the main building block in object-oriented modelling

2.1 Data structures

The process model we have used here is iterative model. The iterative model is a particular implementation of a software development life cycle (SDLC) that focuses on an initial, simplified implementation, which then progressively gains more complexity and a broader feature set until the final system is complete. When discussing the iterative method, the concept of incremental development will also often be used liberally and interchangeably, which describes the incremental alterations made during the design and implementation of each new iteration.

Unlike the more traditional waterfall model, which focuses on a stringent step-by-step process of development stages, the iterative model is best thought of as a cyclical process. The requirements of our project can be accomplished through iterative model.

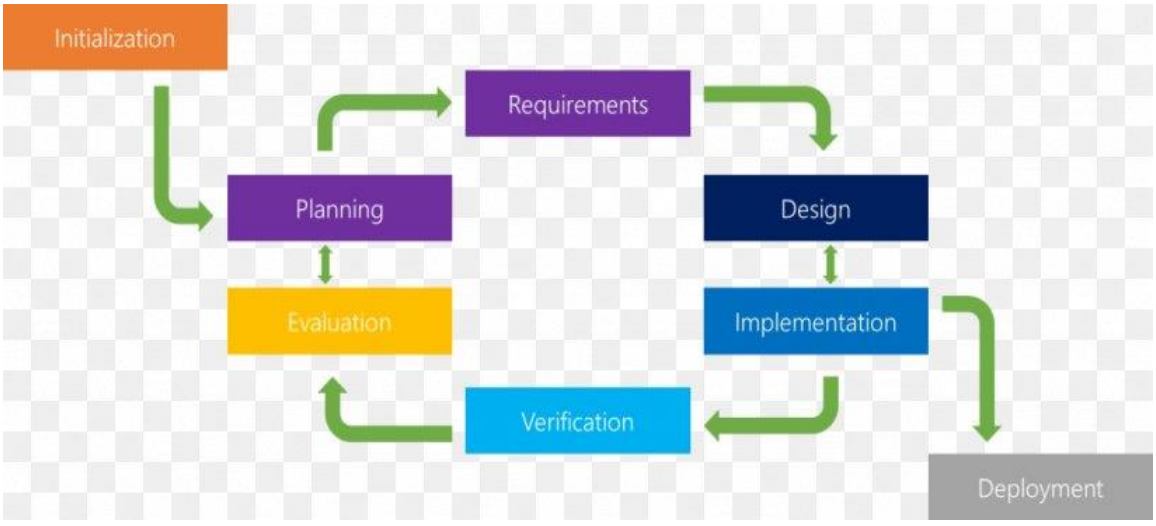


Fig: Iterative and Incremental development systems development life cycle software development

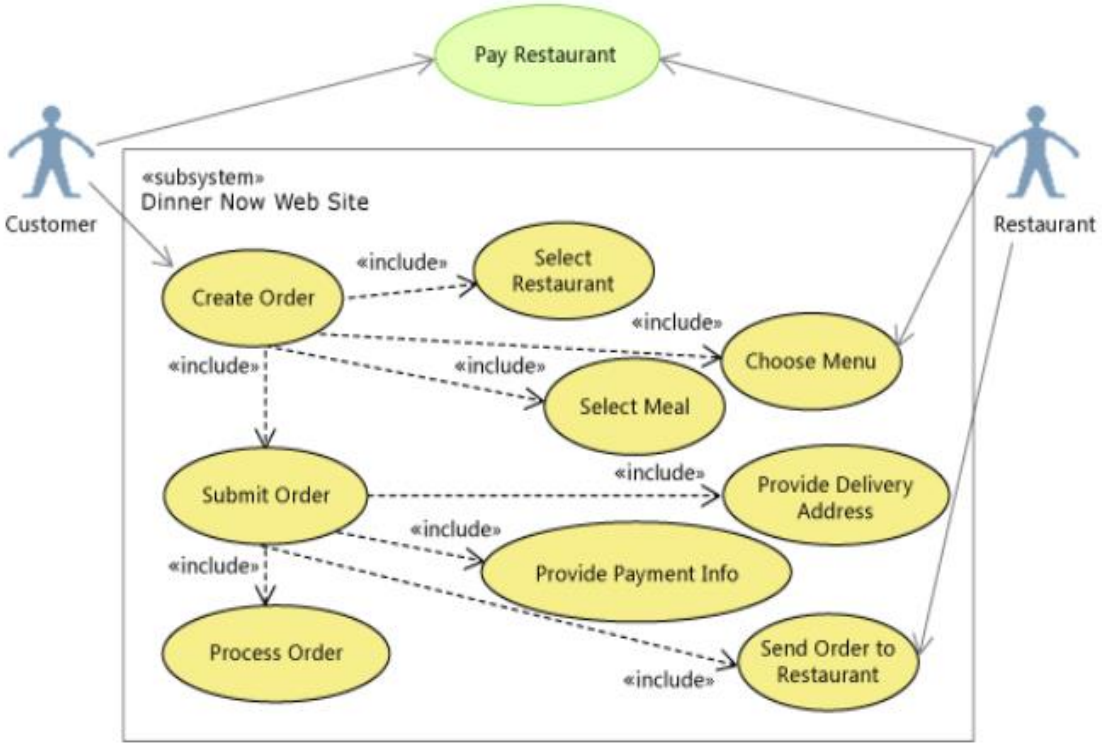


Fig: Use-Case Diagram for Restaurant Management System.

2.2 Database description

Database design is the organization of data according to a database model. The following Database Design determines what data must be stored and how the data elements interrelate. With this information, Developer would begin to fit the data to the database model. Database design involves classifying data and identifying interrelationships. Developer has done conceptual database design on the restaurant management system those has been given below.

Conceptual Database Design: It helps to build conceptual design of database which includes identification, entities, relationship, and attributes. Developer has developed conceptual designed on Restaurant Management System.

Class diagram is the main building box of object- oriented modelling. General conceptual modeling of the structure of the application, and for detailed modeling translating the models into programming code was done using this. Databases created for this restaurant management system are User Table, Restaurant Table, Cart Table, Checkout Table. Other tables are not yet decide still working on it.

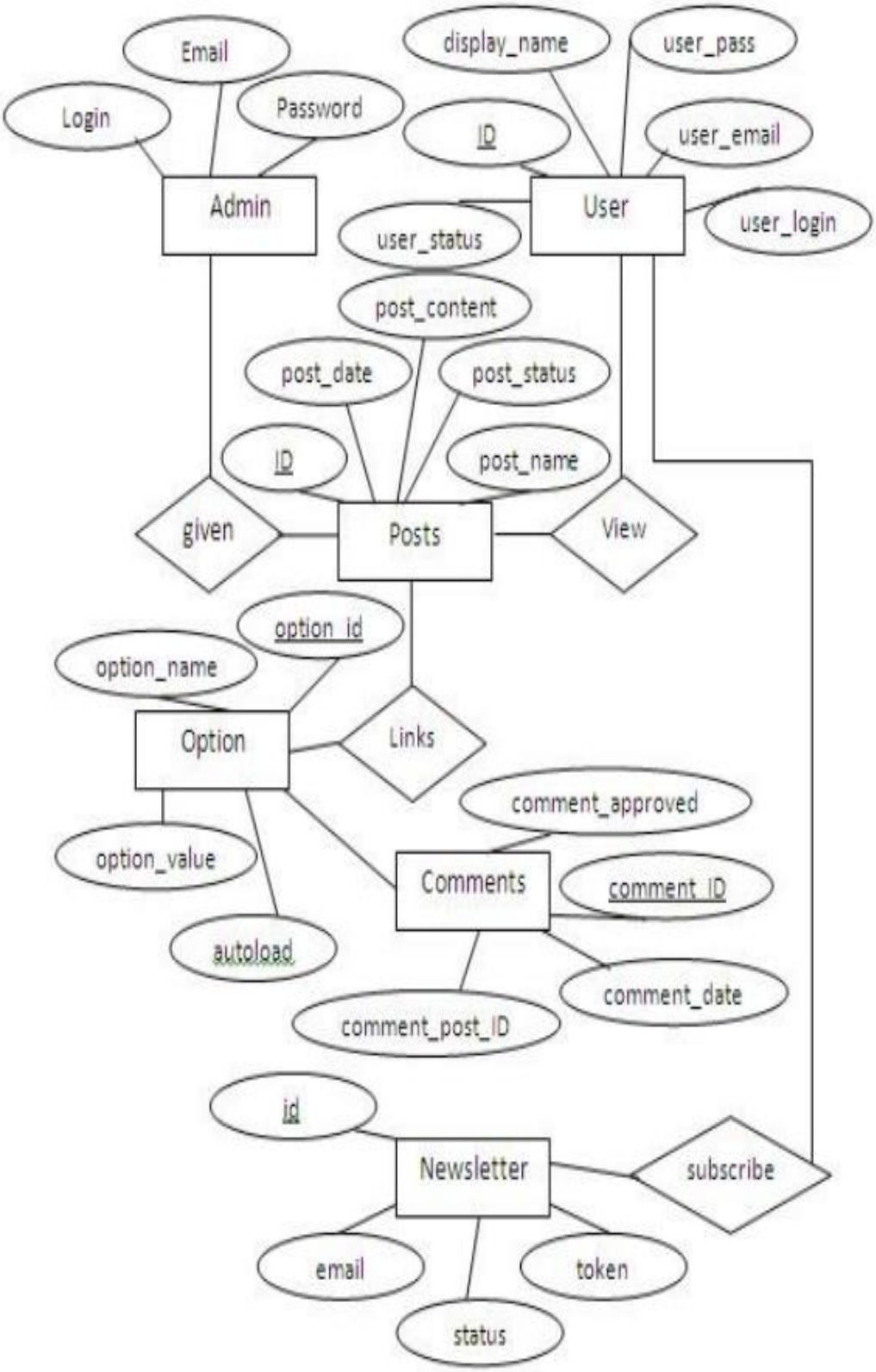


Fig: E-R diagram for Restaurant Management System.

User Table:

id	email	password	role	budget
1	aaa@gmail.com	aaa	1	700
2	bbb@gmail.com	zzz	1	300
3	ccc@gmail.com	zzz	1	500

Restaurant Table:

id	name	price	url
1	Lobster Bisque	5.95	assets/img/menu/lobster-bisque.jpg
2	Crab Cake	7.95	assets/img/menu/cake.jpg
3	Tuscan	9.95	assets/img/menu/tuscan-grilled.jpg
4	Greek Salad	9.95	assets/img/menu/greek-salad.jpg
5	Lobster Roll	12.95	assets/img/menu/lobster-roll.jpg
6	Bread Barrel	6.95	assets/img/menu/bread-barrel.jpg
7	Caesar Selections	8.95	assets/img/menu/caesar.jpg
8	Mozzarella Stick	4.95	assets/img/menu/mozzarella.jpg
9	Spinash Salad	9.95	assets/img/menu/spinach-salad.jpg
(Auto)	*(NULL)*	*(NULL)*	*(NULL)*

Cart Table:

id	email	foodname
24	aaa@gmail.com	Caesar Selections
(Auto)	*(NULL)*	*(NULL)*

Checkout Table:

id	email	foodname
(Auto)	*(NULL)*	*(NULL)*

Fig: Class diagram of Restaurant Management System.

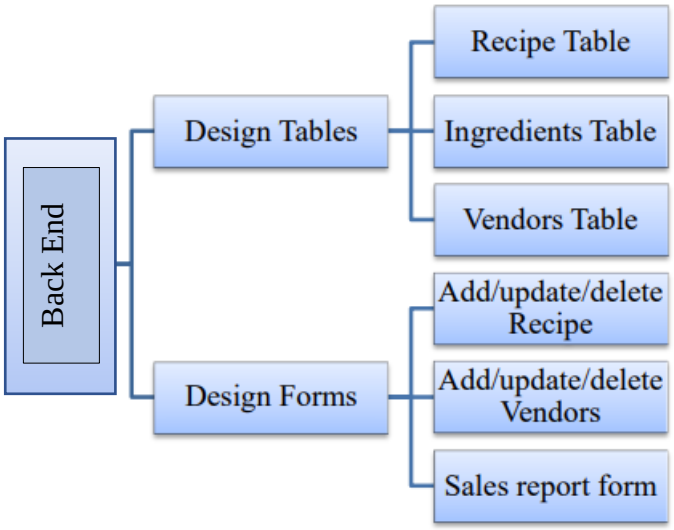
3.0 Architectural and component-level design

MVC or Model View Controller architecture was used to develop this system. As a software design pattern for developing web applications MVC is popular.
MVC architecture divides web application in to three parts. All those parts are interconnected. It is fully capable to support rapid web application development and dynamic interactivity with the database.

Front End:

- Visual Studio
- GUI Design
- Database Modelling
- Design Control

Back End:



3.1 Architecture diagrams

The most common architecture style for a distributed system such as web application is the conventional client-server architecture. It divides the system into server components that offer services and client components that provide user interface to consume the services through connected networks. The server could serve request from multiple clients. This architecture style is also known as 2-tier architecture.

Clients are often represented by range of applications with GUI to capture user inputs and transform them into requests to the server. The server contains data storage to collect, modify and distribute data. Client could be either a thin client if application processing logic is located at server side or a fat client if it is embedded with client application. However, this architecture style lacks scalability because both the client and the server have limited resources. Reusing the application logic for different modules is also increasingly complex as systems expand due to their tight coupling between either data tier or presentation logic. In addition, it is difficult to maintain because it is hard to distribute new changes to system users.

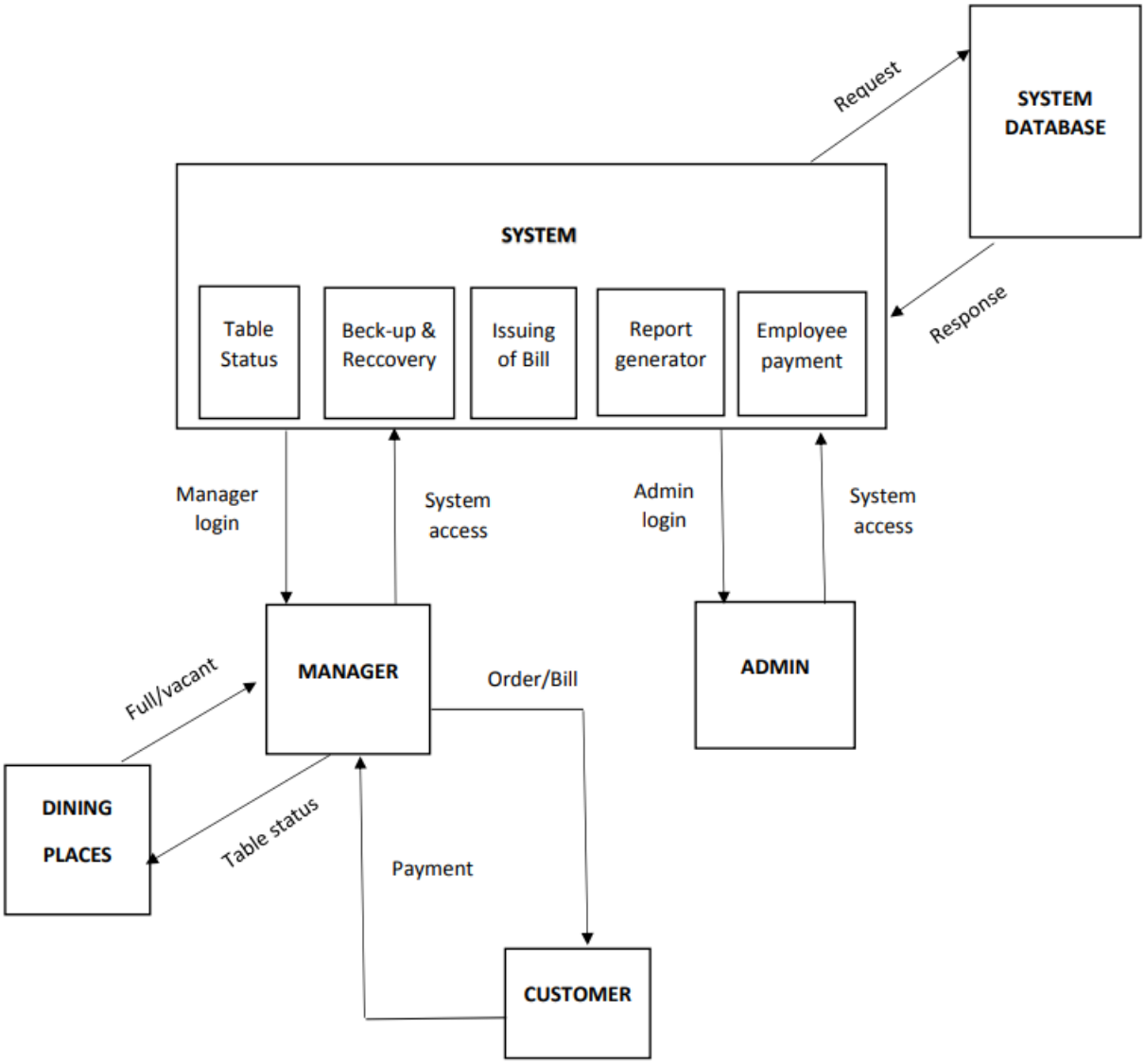


Fig: Architecture Design

3.2 Description for Components

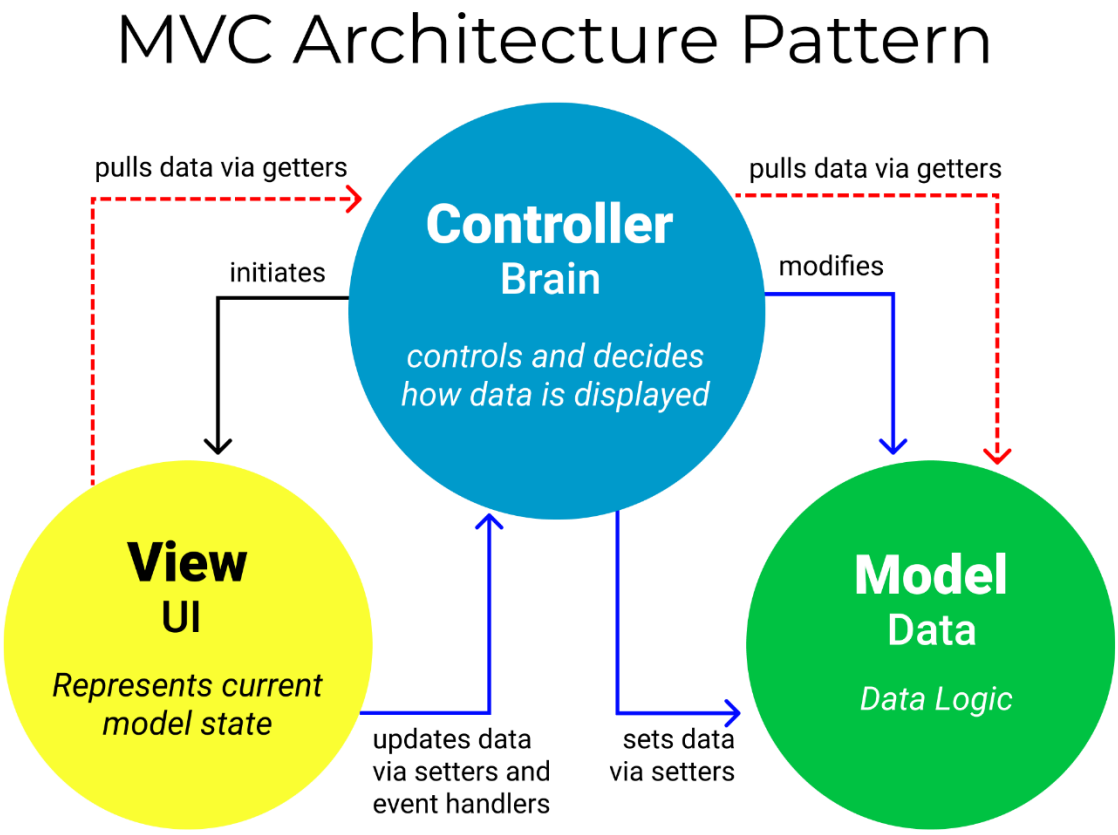


Fig: The Model View Controller Pattern Architecture.

3.2.1 Model Component and description

3.2.1.1 Interface description

The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. For example, a customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data.

3.2.3.2 Static models

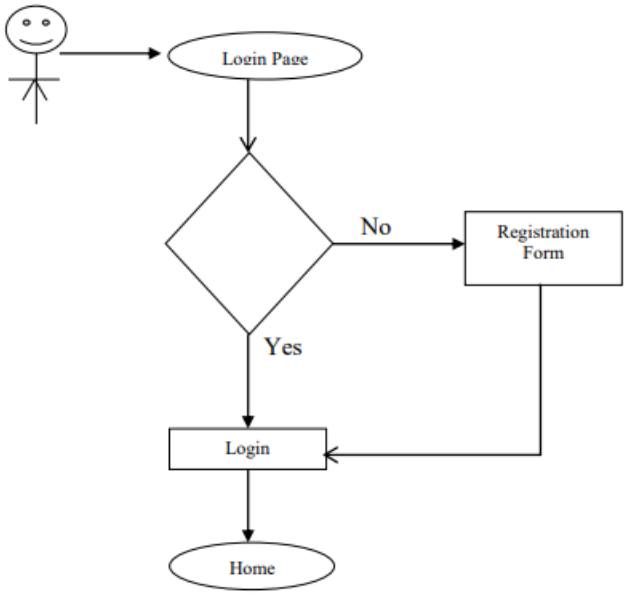


Fig: Registration Workflow Process

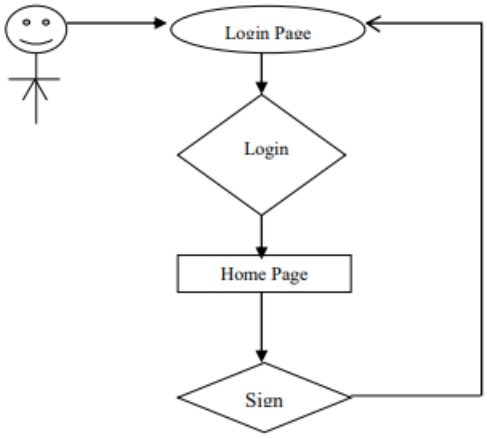


Fig: Login Workflow Process

3.2.1 View Component n description

3.2.1.1 Interface description

The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

3.2.1 Controller Component n description

3.2.1.1 Interface description

Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.

3.2.3.2 Static models

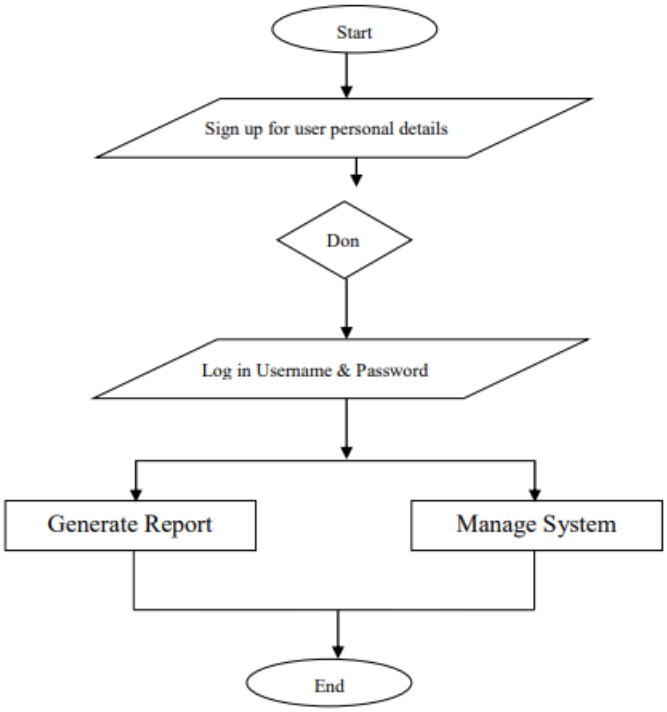


Fig: Admin Workflow Process.

3.2.3.3 Dynamic models

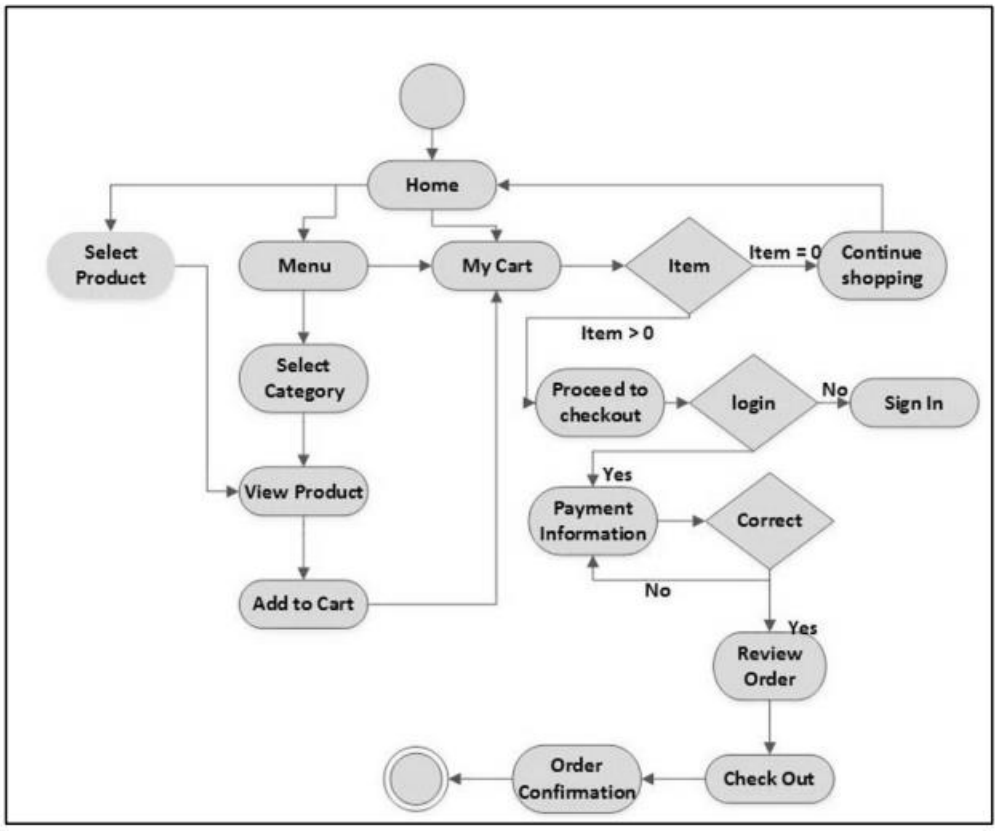


Fig: Activity Diagram for customer order.

3.3 External Interface Description

The software's external interface is the graphical user interface (GUI). The user can input data into the system through the GUI. The system then processes the data and displays the results on the GUI. The user can also view the results of the processing on the GUI.

The system also has an interface to access the status of the order. The user can check through logging in and checking the status of the order. The system then processes the data and displays the results on the screen. The user can also view the results of the processing on the screen.

4.0 User interface design

A description of the user interface design of the software is presented.

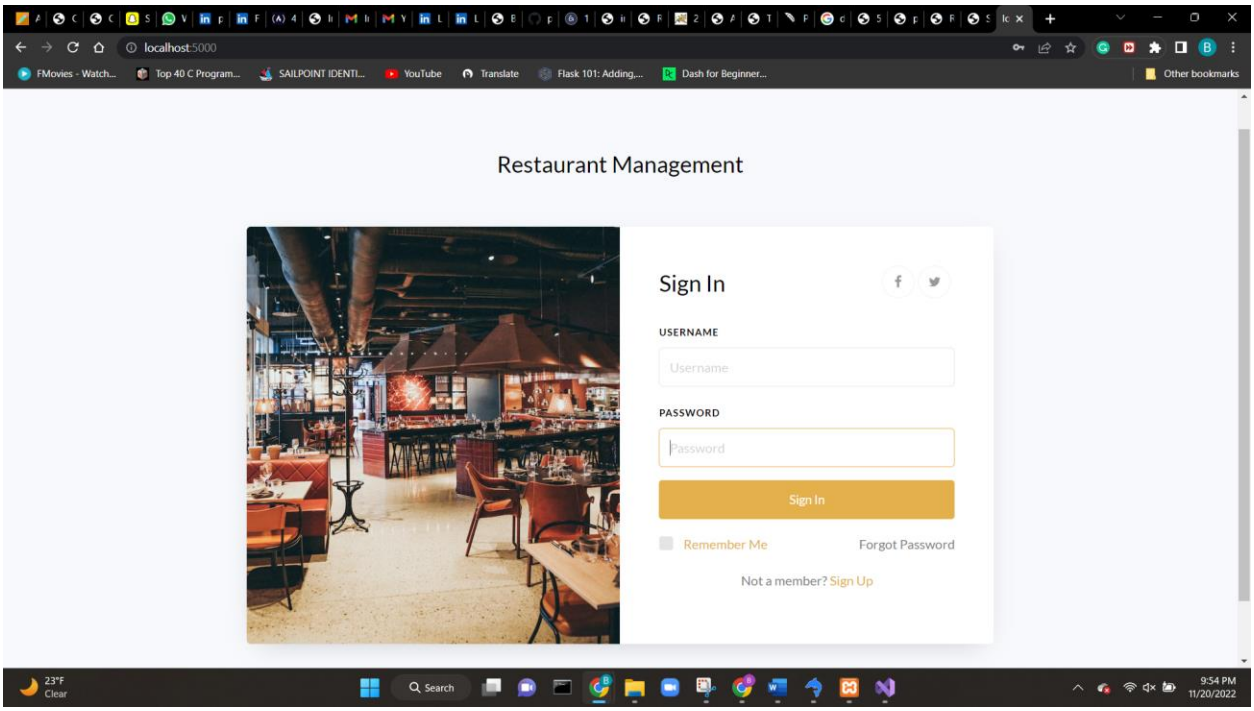


Fig: Sign in Page

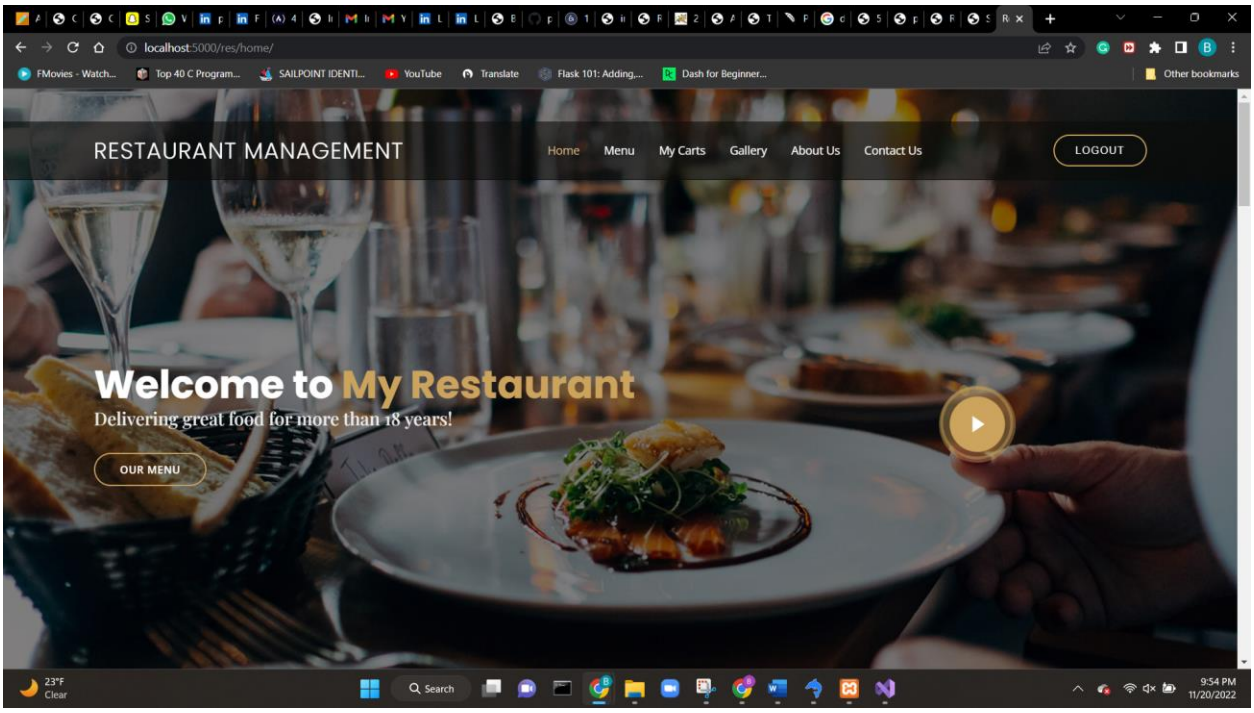


Fig: Home Page

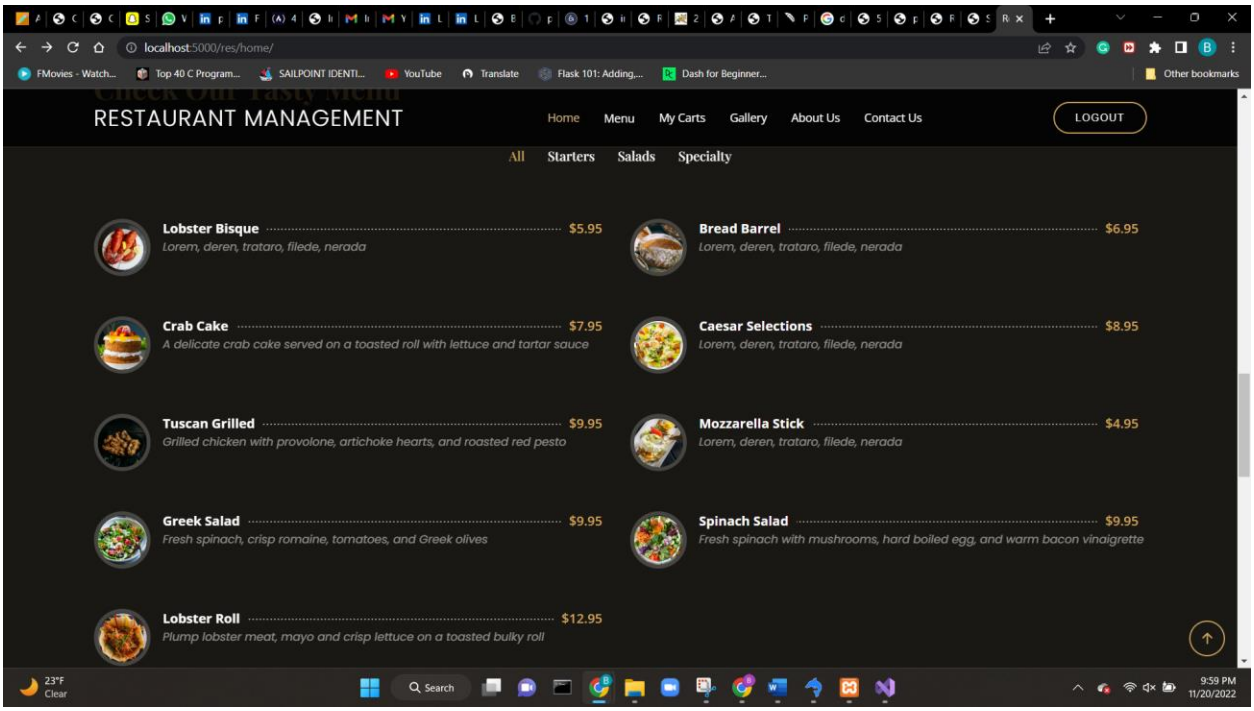


Fig: Menu Page

4.1 Description of the user interface

User interface is the primary and very first interaction with user and system. So, it should be interactive to the user. User Interface (UI) Design focuses on what users expecting through system and ensuring that the interface has fundamental facilities which are easy to understand, access and use. There are some properties that user interface must have,

- Avoid unnecessary elements and keep interface simple.
- Purposefully use color and texture: make direct attention toward or redirect attention away from items using color, texture, contrast.
- Use auto select options as well as let the user's select options for better flexibility.
- Use common UI elements and build consistency by using more comfortable common elements, for users and able to get things done fast.
- Use typography to create hierarchy and clarity by using different sizes, fonts, and arrangement of the text to help increase scalability, legibility, and readability.

4.2 Interface design rules

- The user interface should be designed using a standard GUI toolkit.
- The user interface should be designed using standard widgets such as buttons, labels, text fields, etc.
- The user interface layout should be simple and easy to understand.
- The user interface should be designed to be easy to use and navigate.

5.0 Restrictions, limitations, and constraints

One of the most important design issues is the need to create an easy and effective user interface. Another design issue is the need to ensure that the software can handle large data. Finally, the design issue of security is also important, as the software must be able to protect the data it stores from unauthorized access. Due to lack of knowledge about development, the client may have little computer knowledge and due to that it may be difficult to capture the requirement. Usually, staff who works in this kind of place also less knowledgeable and they might be afraid to work with computers. It may be difficult to explain about this system and gather the requirements they need.

6.0 Appendices

Presenting information that supplements the design specification through use case diagram.

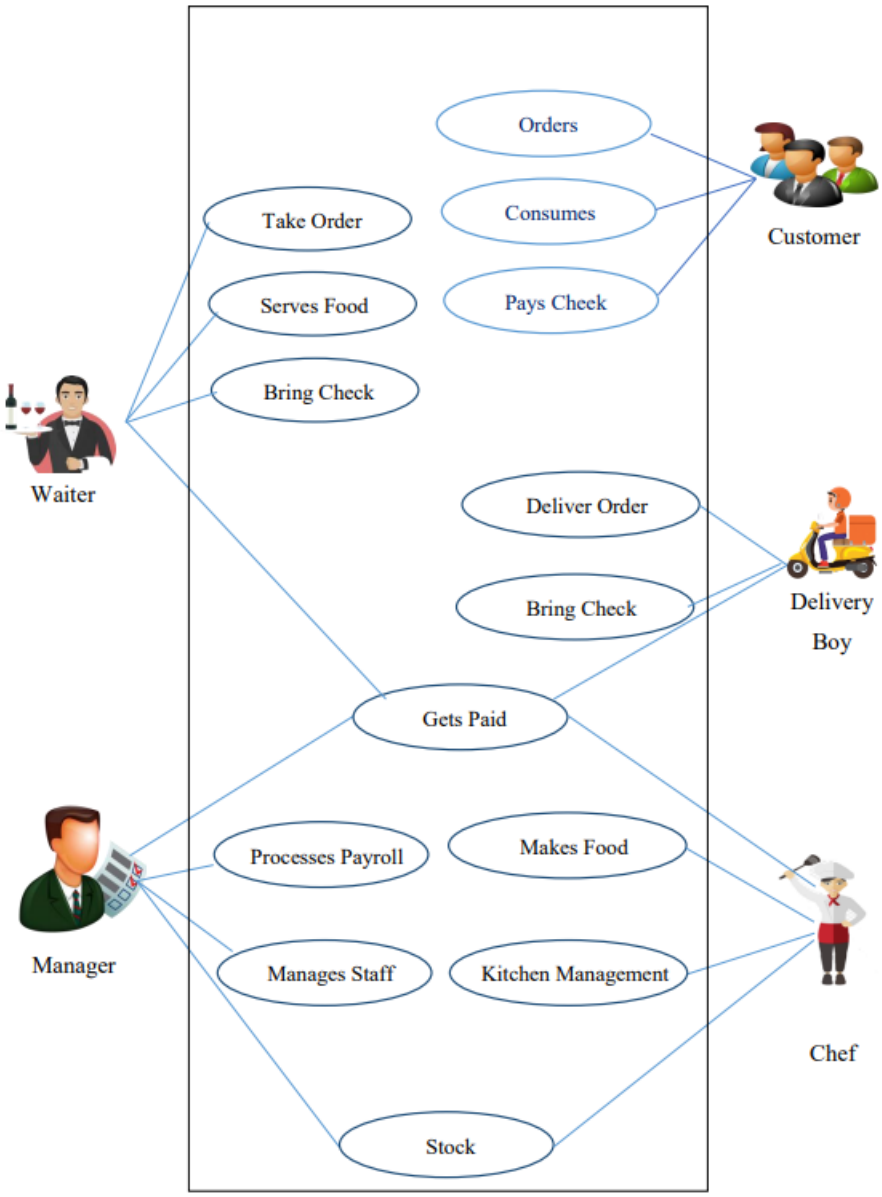


Fig: User case diagram.

6.1 Implementation issues

In Implementation process both server side and client environments were considered. There are environment requirements in this process. Those requirements are divided as software and hardware requirements. Mainly the internet connection and MySQL database connection must be up and should be running since we are hosting in the local host.