project report - online food delivery

MCA/ASSIGN/SEMESTER-I (Visvesvaraya Technological University)

Online Food Ordering System

Introduction of the Project Online Food Ordering System:

The "Online Food Ordering System" has been developed to override the problems prevailing in the practicing manual system. This software is supported to eliminate and, in some cases, reduce the hardships faced by this existing system. Moreover, this system is designed for the particular need of the company to carry out operations in a smooth and effective manner.

The application is reduced as much as possible to avoid errors while entering the data. It also provides error message while entering invalid data. No formal knowledge is needed for the user to use this system. Thus, by this all it proves it is user-friendly. Online Food Ordering System, as described above, can lead to error free, secure, reliable and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping. Thus, it will help organization in better utilization of resources.

Every organization, whether big or small, has challenges to overcome and managing the information of Category, Food Item, Order, Payment, Confirm Order. Every Online Food Ordering System has different Food Item needs; therefore, we design exclusive employee management systems that are adapted to your managerial requirements. This is designed to assist in strategic planning and will help you ensure that your organization is equipped with the right level of information and details for your future goals. Also, for those busy executives who are always on the go, our systems come with remote access features, which will allow you to manage your workforce anytime, at all times. These systems will ultimately allow you to better manage resources.

Abstract of the Project Online Food Ordering System:

The purpose of Online Food Ordering System is to automate the existing manual system by the help of computerized equipment's and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same. The required software and hardware are easily available and easy to work with.

Online Food Ordering System, as described above, can lead to error free, secure, reliable and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping. Thus it will help organization in better utilization of resources. The organization can maintain computerized records without redundant entries. That means that one need not be distracted by information that is not relevant, while being able to reach the information.

The aim is to automate its existing manual system by the help of computerized equipment's and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same. Basically the project describes how to manage for good performance and better services for the clients.

Objective of Project on Online Food Ordering System:

The main objective of the Project on Online Food Ordering System is to manage the details of Food Item, Category, Customer, Order, Confirm Order. It manages all the information about Food Item, Payment, Confirm Order, Food Item. The project is totally built at administrative end and thus only the administrator is guaranteed the access. The purpose of the project is to build an application program to reduce the manual work for managing the Food Item, Category, Payment, Customer. It tracks all the details about the Customer, Order, Confirm Order.

Functionalities provided by Online Food Ordering System are as follows:

- Provides the searching facilities based on various factors. Such as Food Item, Customer, Order, Confirm Order
- Online Food Ordering System also manage the Payment details online for Order details, Confirm Order details, Food Item.
- It tracks all the information of Category, Payment, Order etc
- Manage the information of Category
- Shows the information and description of the Food Item, Customer
- To increase efficiency of managing the Food Item, Category
- It deals with monitoring the information and transactions of Order.
- Manage the information of Food Item
- Editing, adding and updating of Records is improved which results in proper resource management of Food Item data.
- Manage the information of Order
- Integration of all records of Confirm Order.

Scope of the project Online Food Ordering System

It may help collecting perfect management in detail. In a very short time, the collection will be obvious, simple and sensible. It will help a person to know the management of passed year perfectly and vividly. It also helps in current all works relative to Online Food Ordering System. It will be also reduced the cost of collecting the management & collection procedure will go on smoothly.

Our project aims at Business process automation, i.e. we have tried to computerize various processes of Online Food Ordering System.

- In computer system the person has to fill the various forms & number of copies of the forms can be easily generated at a time.
- In computer system, it is not necessary to create the manifest but we can directly print it, which saves our time.
- To assist the staff in capturing the effort spent on their respective working areas.
- To utilize resources in an efficient manner by increasing their productivity through automation.
- The system generates types of information that can be used for various purposes.
- It satisfy the user requirement
- Be easy to understand by the user and operator
- Be easy to operate
- Have a good user interface
- Be expandable
- Delivered on schedule within the budget.

Reports of Online Food Ordering System:

- It generates the report on Food Item, Category, Payment
- Provide filter reports on Customer, Order, Confirm Order
- You can easily export PDF for the Food Item, Payment, Order
- Application also provides excel export for Category, Customer, Confirm Order
- You can also export the report into csv format for Food Item, Category, Confirm Order

Modules of Online Food Ordering System:

- Food Item Management Module: Used for managing the Food Item details.
- Confirm Order Module : Used for managing the details of Confirm Order
- Payment Module : Used for managing the details of Payment
- Category Management Module: Used for managing the information and details of the Category.
- Customer Module : Used for managing the Customer details
- Order Module : Used for managing the Order information's
- Login Module: Used for managing the login details
- Users Module : Used for managing the users of the system

Input Data and Validation of Project on Online Food Ordering System

- All the fields such as Food Item, Customer, Confirm Order are validated and does not take invalid values
- Each form for Food Item, Category, Payment can not accept blank value fields
- Avoiding errors in data
- Controlling amount of input
- Integration of all the modules/forms in the system.
- Preparation of the test cases.
- Preparation of the possible test data with all the validation checks.
- Actual testing done manually.
- Recording of all the reproduced errors.
- Modifications done for the errors found during testing.
- Prepared the test result scripts after rectification of the errors.
- Functionality of the entire module/forms.
- Validations for user input.
- Checking of the Coding standards to be maintained during coding.
- Testing the module with all the possible test data.
- Testing of the functionality involving all type of calculations etc.
- Commenting standard in the source files.

The software quality plan we will use the following SQA Strategy:

- In the first step, we will select the test factors and rank them. The selected test factors such as reliability, maintainability, portability or etc., will be placed in the matrix according to their ranks.
- The second step is for identifying the phases of the development process. The phase should be recorded in the matrix.
- The third step is that identifying the business risks of the software deliverables. The risks will be ranked into three ranks such as high, medium and low.

<u>Features of the project Online Food Ordering System:</u>

- Product and Component based
- Creating & Changing Issues at ease
- Query Issue List to any depth
- Reporting & Charting in more comprehensive way
- User Accounts to control the access and maintain security
- Simple Status & Resolutions
- Multi-level Priorities & Severities.
- Targets & Milestones for guiding the programmers
- Attachments & Additional Comments for more information
- Robust database back-end
- Various level of reports available with a lot of filter criteria's
- It contains better storage capacity.
- Accuracy in work.
- Easy & fast retrieval of information.
- Well-designed reports.
- Decrease the load of the person involve in existing manual system.
- Access of any information individually.
- Work becomes very speedy.
- Easy to update information

Software Requirement Specification

The Software Requirements Specification is produced at the culmination of the analysis task. The function and performance allocated to software as part of system engineering are refined by establishing a complete information description, a detailed functional and behavioral description, an indication of performance requirements and design constraints, appropriate validation criteria, and other data pertinent to requirements.

The proposed system has the following requirements:

- System needs store information about new entry of Food Item.
- System needs to help the internal staff to keep information of Category and find them as per various queries.
- System need to maintain quantity record.
- System need to keep the record of Customer.
- System need to update and delete the record.
- System also needs a search area.
- It also needs a security system to prevent data.

Identification of need:

The old manual system was suffering from a series of drawbacks. Since whole of the system was to be maintained with hands the process of keeping, maintaining and retrieving the information was very tedious and lengthy. The records were never used to be in a systematic order. there used to be lots of difficulties in associating any particular transaction with a particular context. If any information was to be found it was required to go through the different registers, documents there would never exist anything like report generation. There would always be unnecessary consumption of time while entering records and retrieving records. One more problem was that it was very difficult to find errors while entering the records. Once the records were entered it was very difficult to update these records.

The reason behind it is that there is lot of information to be maintained and have to be kept in mind while running the business .For this reason we have provided features Present system is partially automated (computerized), actually existing system is quite laborious as one has to enter same information at three different places.

Following points should be well considered:

- Documents and reports that must be provided by the new system: there can also be few reports, which can help management in decision-making and cost controlling, but since these reports do not get required attention, such kind of reports and information were also identified and given required attention.
- Details of the information needed for each document and report.
- The required frequency and distribution for each document.
- Probable sources of information for each document and report.
- With the implementation of computerized system, the task of keeping records in an organized manner will be solved. The greatest of all is the retrieval of information, which will be at the click of the mouse. So the proposed system helps in saving the time in different operations and making information flow easy giving valuable reports.

Feasibility Study:

After doing the project Online Food Ordering System, study and analyzing all the existing or required functionalities of the system, the next task is to do the feasibility study for the project. All projects are feasible - given unlimited resources and infinite time.

Feasibility study includes consideration of all the possible ways to provide a solution to the given problem. The proposed solution should satisfy all the user requirements and should be flexible enough so that future changes can be easily done based on the future upcoming requirements.

A. Economical Feasibility

This is a very important aspect to be considered while developing a project. We decided the technology based on minimum possible cost factor.

- All hardware and software cost has to be borne by the organization.
- Overall we have estimated that the benefits the organization is going to receive from the proposed system will surely overcome the initial costs and the later on running cost for system.

B. Technical Feasibility

This included the study of function, performance and constraints that may affect the ability to achieve an acceptable system. For this feasibility study, we studied complete functionality to be provided in the system, as described in the System Requirement Specification (SRS), and checked if everything was possible using different type of frontend and backend plaformst.

C. Operational Feasibility

No doubt the proposed system is fully GUI based that is very user friendly and all inputs to be taken all self-explanatory even to a layman. Besides, a proper training has been conducted to let know the essence of the system to the users so that they feel comfortable with new system. As far our study is concerned the clients are comfortable and happy as the system has cut down their loads and doing.

System Design of Online Food Ordering System

In this phase, a logical system is built which fulfils the given requirements. Design phase of software development deals with transforming the client's requirements into a logically working system. Normally, design is performed in the following in the following two steps:

1. **Primary Design Phase:**
   In this phase, the system is designed at block level. The blocks are created on the basis of analysis done in the problem identification phase. Different blocks are created for different functions emphasis is put on minimizing the information flow between blocks. Thus, all activities which require more interaction are kept in one block.
2. **Secondary Design Phase:**
   In the secondary phase the detailed design of every block is performed.

The general tasks involved in the design process are the following:

1. Design various blocks for overall system processes.
2. Design smaller, compact and workable modules in each block.
3. Design various database structures.
4. Specify details of programs to achieve desired functionality.
5. Design the form of inputs, and outputs of the system.
6. Perform documentation of the design.
7. System reviews.

User Interface Design

User Interface Design is concerned with the dialogue between a user and the computer. It is concerned with everything from starting the system or logging into the system to the eventually presentation of desired inputs and outputs. The overall flow of screens and messages is called a dialogue

.

The following steps are various guidelines for User Interface Design:

1. The system user should always be aware of what to do next.
2. The screen should be formatted so that various types of information, instructions and messages always appear in the same general display area.
3. Message, instructions or information should be displayed long enough to allow the system user to read them.
4. Use display attributes sparingly.
5. Default values for fields and answers to be entered by the user should be specified.
6. A user should not be allowed to proceed without correcting an error.
7. The system user should never get an operating system message or fatal error.

Preliminary Product Description:

The first step in the system development life cycle is the preliminary investigation to determine the feasibility of the system. The purpose of the preliminary investigation is to evaluate project requests. It is not a design study nor does it include the collection of details to describe the business system in all respect. Rather, it is the collecting of information that helps committee members to evaluate the merits of the project request and make an informed judgment about the feasibility of the proposed project.

Analysts working on the preliminary investigation should accomplish the following objectives:

- Clarify and understand the project request
- Determine the size of the project.
- Assess costs and benefits of alternative approaches.
- Determine the technical and operational feasibility of alternative approaches.
- Report the findings to management, with recommendations outlining the acceptance or rejection of the proposal.

- **Benefit to Organization**

The organization will obviously be able to gain benefits such as savings in operating cost, reduction in paperwork, better utilization of human resources and more presentable image increasing goodwill.

- **The Initial Cost**

The initial cost of setting up the system will include the cost of hardware software (OS, add-on software, utilities) & labor (setup & maintenance). The same has to bear by the organization.

- **Running Cost**

Besides, the initial cost the long term cost will include the running cost for the system including the AMC, stationary charges, cost for human resources, cost for update/renewal of various related software.

- **Need for Training**

The users along with the administrator need to be trained at the time of implementation of the system for smooth running of the system. The client will provide the training site.

We talked to the management people who were managing a the financial issues of the center, the staff who were keeping the records in lots of registers and the reporting manager regarding their existing system, their requirements and their expectations from the new proposed system. Then, we did the system study of the entire system based on their requirements and the additional features they wanted to incorporate in this system.

Reliable, accurate and secure data was also considered to be a complex task without this proposed system. Because there was no such record for keeping track of all the activities, which was done by the Online Food Ordering System on the daily basis.

The new system proposed and then developed by me will ease the task of the organization in consideration. It will be helpful in generating the required reports by the staff, which will help them to track their progress and services.

Thus, it will ease the task of Management to a great extent as all the major activities to be performed, are computerized through this system.

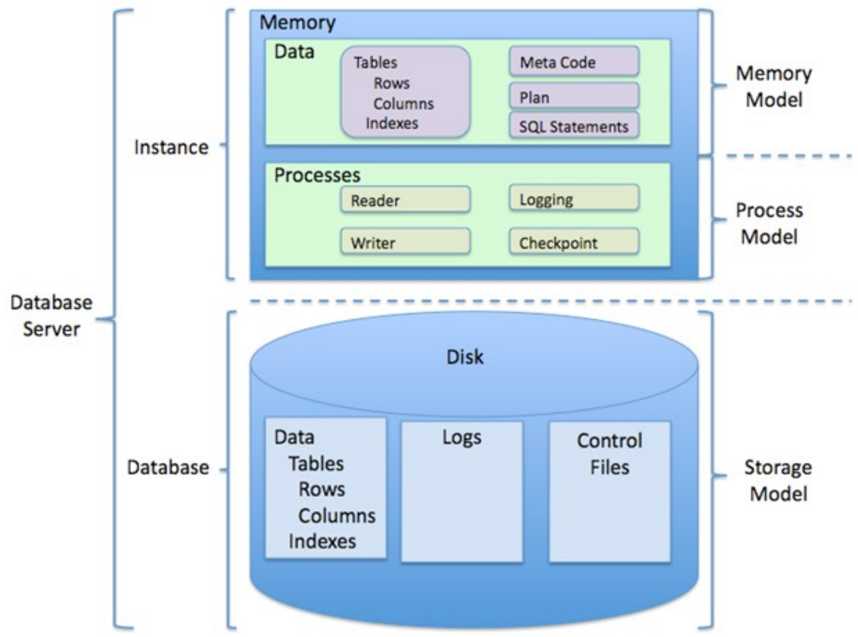
Project Category

Relational Database Management System (RDBMS) : This is an RDBMS based project which is currently using MySQL for all the transaction statements. MySQL is an opensource RDBMS System.

Brief Introduction about RDBSM :

A relational database management system (RDBMS) is a database management system (DBMS) that is based on the relational model as invented by E. F. Codd, of IBM's San Jose Research Laboratory. Many popular databases currently in use are based on the relational database model.

RDBMSs have become a predominant choice for the storage of information in new databases used for financial records, manufacturing and logistical information, personnel data, and much more since the 1980s. Relational databases have often replaced legacy hierarchical databases and network databases because they are easier to understand and use. However, relational databases have been challenged by object databases, which were introduced in an attempt to address the object-relational impedance mismatch in relational database, and XML databases.



<u>Implementation Methodology:</u>

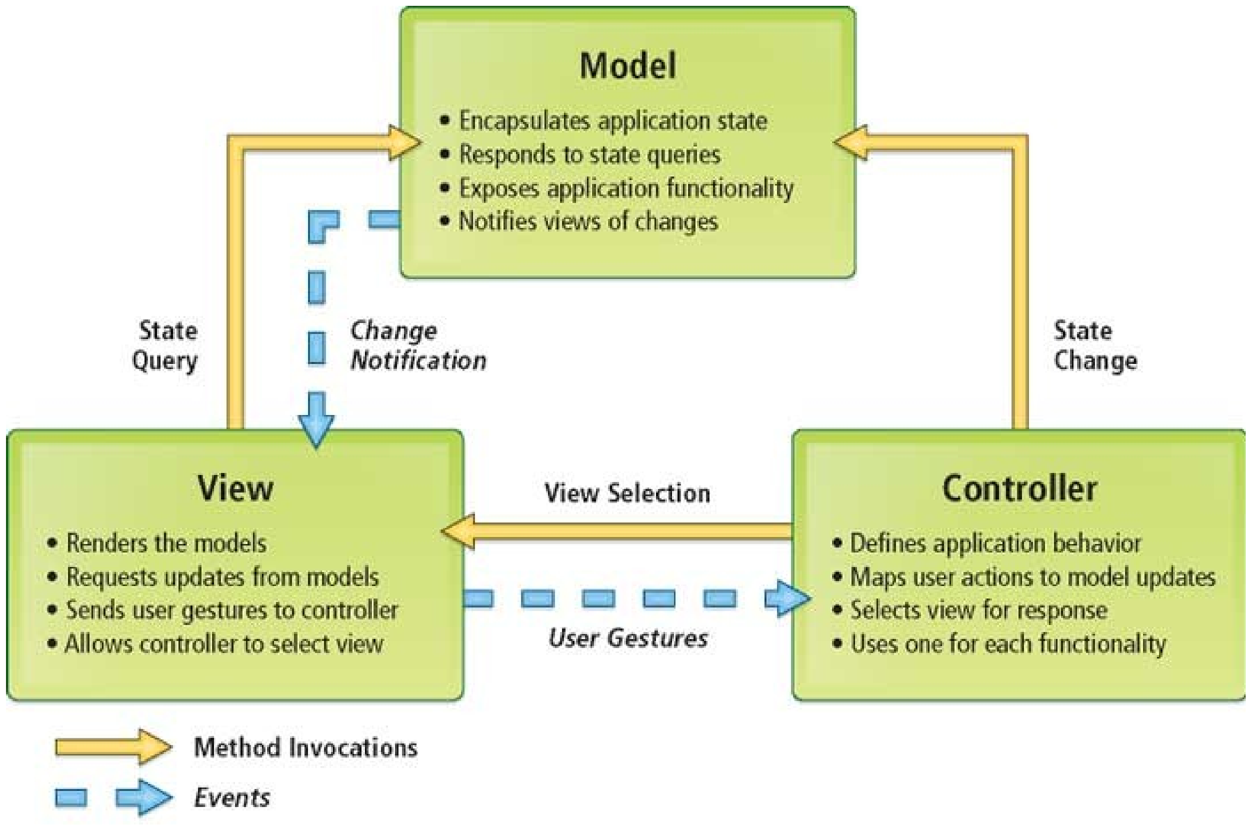
Model View Controller or MVC as it is popularly called, is a software design pattern for developing web applications. A Model View Controller pattern is made up of the following three parts:

- **Model** - The lowest level of the pattern which is responsible for maintaining data.
- **View** - This is responsible for displaying all or a portion of the data to the user.
- **Controller** - Software Code that controls the interactions between the Model and View.

MVC is popular as it isolates the application logic from the user interface layer and supports separation of concerns. Here the Controller receives all requests for the application and then works with the Model to prepare any data needed by the View. The View then uses the data prepared by the Controller to generate a final presentable response. The MVC abstraction can be graphically represented as follows.

<u>MVC (Model View Controller Flow) Diagram</u>

<u>DATA FLOW DIAGRAMS</u>


Model
• Encapsulates application state
• Responds to state queries
• Exposes application functionality
• Notifies views of changes
State Query
Change Notification
State Change
View
• Renders the models
• Requests updates from models
• Sends user gestures to controller
• Allows controller to select view
View Selection
User Gestures
Controller
• Defines application behavior
• Maps user actions to model updates
• Selects view for response
• Uses one for each functionality
Method Invocations
Events

<u>Project Planning:</u>

Software project plan can be viewed as the following:

1) **<u>Within the organization:</u>** How the project is to be implemented? What are various constraints (time, cost, staff)? What is market strategy?
2) **<u>With respect to the customer:</u>** Weekly or timely meetings with the customer with presentation on status reports. Customers feedback is also taken and further modification and developments are done. Project milestones and deliverables are also presented to the customer.

<u>For a successful software project, the following steps can be followed:</u>

- Select a project
  - Identifying project's aims and objectives
  - Understanding requirements and specification
  - Methods of analysis, design and implementation
  - Testing techniques
  - Documentation
- Project milestones and deliverables
- Budget allocation
  - Exceeding limits within control
- Project Estimates
  - Cost
  - Time
  - Size of code

        - Duration

- Resource Allocation
    - Hardware
    - Software
    - Previous relevant project information
    - Digital Library
- Risk Management
        - Risk avoidance
        - Risk detection

Cost estimation of the project:

Software cost comprises a small percentage of overall computer-based system cost. There are a number of factors, which are considered, that can affect the ultimate cost of the software such as - human, technical, Hardware and Software availability etc.

The main point that was considered during the cost estimation of **project** was its sizing. In spite of complete software sizing, function point and approximate lines of code were also used to "size" each element of the Software and their costing.

The cost estimation done by me for **Project** also depend upon the baseline metrics collected from past projects and these were used in conjunction with estimation variables to develop cost and effort projections.

We have basically estimated this project mainly on two bases -

1) Effort Estimation - This refers to the total man-hours required for the development of the project. It even includes the time required for doing documentation and user manual.

2) Hardware Required Estimation - This includes the cost of the PCs and the hardware cost required for development of this project.

Tools/Platform, Hardware and Software Requirement specifications:

Software Requirements:

Name of component	Specification
Operating System	Windows 98, Windows XP, Windows7, Linux
Language	Python 3.5
Database	SQL lite Server
Browser	Any of Mozilla, Opera, Chrome etc.
Framework	Django(2.0)
Web Technologies	Html, CSS, JavaScript

Hardware Requirements:

Name of component	Specification
Processor	Pentium III 630MHz
RAM	128 MB
Hard disk	20 GB
Monitor	15” color monitor
Keyboard	122 keys

Project Profile

There has been continuous effort to develop tools, which can ease the process of software development. But, with the evolving trend of different programming paradigms today's software developers are really challenged to deal with the changing technology. Among other issues, software re-engineering is being regarded as an important process in the software development industry. One of the major tasks here is to understand software systems that are already developed and to transform them to a different software environment. Generally, this requires a lot of manual effort in going through a program that might have been developed by another programmer. This project makes a novel attempt to address the issued of program analysis and generation of diagrams, which can depict the structure of a program in a better way. Today, UML is being considered as an industrial standard for software engineering design process. It essential provides several diagramming tools that can express different aspects/ characteristics of program such as

Use cases: Elicit requirement from users in meaningful chunks. Construction planning is built around delivering some use cases n each interaction basis for system testing.

Class diagrams: shows static structure of concepts, types and class. Concepts how users think about the world; type shows interfaces of software components; classes shows implementation of software components.

Interaction diagrams: shows how several objects collaborate in single use case.

Package diagram: show group of classes and dependencies among them.

State diagram: show how single object behaves across many use cases.

Activity diagram: shows behavior with control structure. Can show many objects over many uses, many object in single use case, or implementations methods encourage parallel behavior, etc.

The end-product of this project is a comprehensive tool that can parse any vb.net program and extract most of the object oriented features inherent in the program such as polymorphism, inheritance, encapsulation and abstraction.

What is UML?

UML stands for Unified Modeling Language is the successor to the wave of Object Oriented Analysis and Design (OOA&D) methods that appeared in the late 80's. It most directly unifies the methods of Booch, Rumbaugh (OMT) and Jacobson. The UML is called a modeling language, not a method. Most methods consist at least in principle, of both a modeling language and a process. The Modeling language is that notation that methods used to express design.

Notations and meta-models:

The notation is the graphical stuff; it is the syntax of the modeling language. For instance, class diagram notation defines how items are concepts such as class, association, and multiplicity is represented. These are:

Class Diagram: The class diagram technique has become truly central within object-oriented methods. Virtually every method has included some variation on this technique. Class diagram is also subject to the greatest range of modeling concept. Although the basic elements are needed by everyone, advanced concepts are used less often. A class diagram describes the types of objects in the system and the various kinds of static relationship that exist among them. There are two principal kinds of static relationship:

- Association
- Subtype

Class diagram also show the attributes and operations of a class and the constraints that apply to the way objects are connected.

Association: Association represent between instances of class. From the conceptual perspective, association represents conceptual relations between classes. Each association has two roles. Each role is a direction on the association. A role also has multiplicity, which is a indication of how many object may participate in the given relationship.

Generalization: A typical example of generalization evolves the personal and corporate customer of a business. They have differences but also many similarity. The similarities can be placed in generalization with personal customer and corporate customer sub type.

Aggregation: aggregation is the part of relationship. It is like saying a car has engine and wheels as its parts. This sounds good, but difficult thing is considering, what is the difference is aggregation and association.

Interaction: interaction diagrams are models that describes how groups of objects collaboration in some behavior.

Typically, an interaction diagram captures the behavior a single use cases. The diagram shows a number of example objects and the messages that are passed between these objects in use cases. These are following approaches with simple use case that exhibits the following behavior.

Objects can send a message to another. Each message is checks with given stock item. There are two diagrams: Sequence and Collaboration diagram.

Package Diagram: One of the oldest questions in software methods is: how do you break down a large system into smaller systems? It becomes difficult to understand and the changes we make to them.

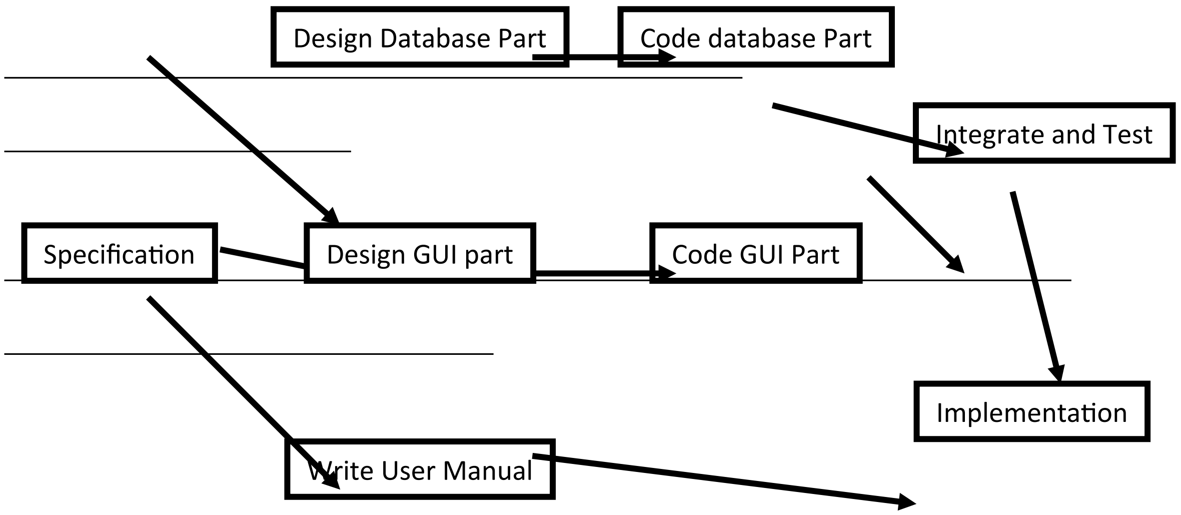
Structured methods used functional decomposition in which the overall system was mapped as a function broken down into sub function, which is further broken down into sub function and so forth. The separation of process data is gone, functional decomposition is gone, but the old question is still remains. One idea is to group the classes together into higher-level unit. This idea, applied very loosely, appears in many objects. In UML, this grouping mechanism is package. The term package diagram for a diagram that shows packages of classes and the dependencies among them.

A dependency exists between two elements if changes to the definition of one element may cause to other. With classes, dependencies exist for various reasons: one class sends a message to another; one class has another as part of its data; one class mentions another as a parameter to an operation. A dependency between two packages exists; and any dependencies exist between any two classes in the package.

State diagram: State diagram are a familiar technique to describe the behavior of a system. They describe all the possible states a particular object can get into and how the objects state changes as a result of events that reach the objects. In most OO technique, state diagrams are drawn for a single class to show the lifetime behavior of a singe object. There are many form of state diagram, each with slightly different semantics. The most popular one used in OO technique is based on David Harel's state chart.

PERT CHART (Program Evaluation Review Technique)

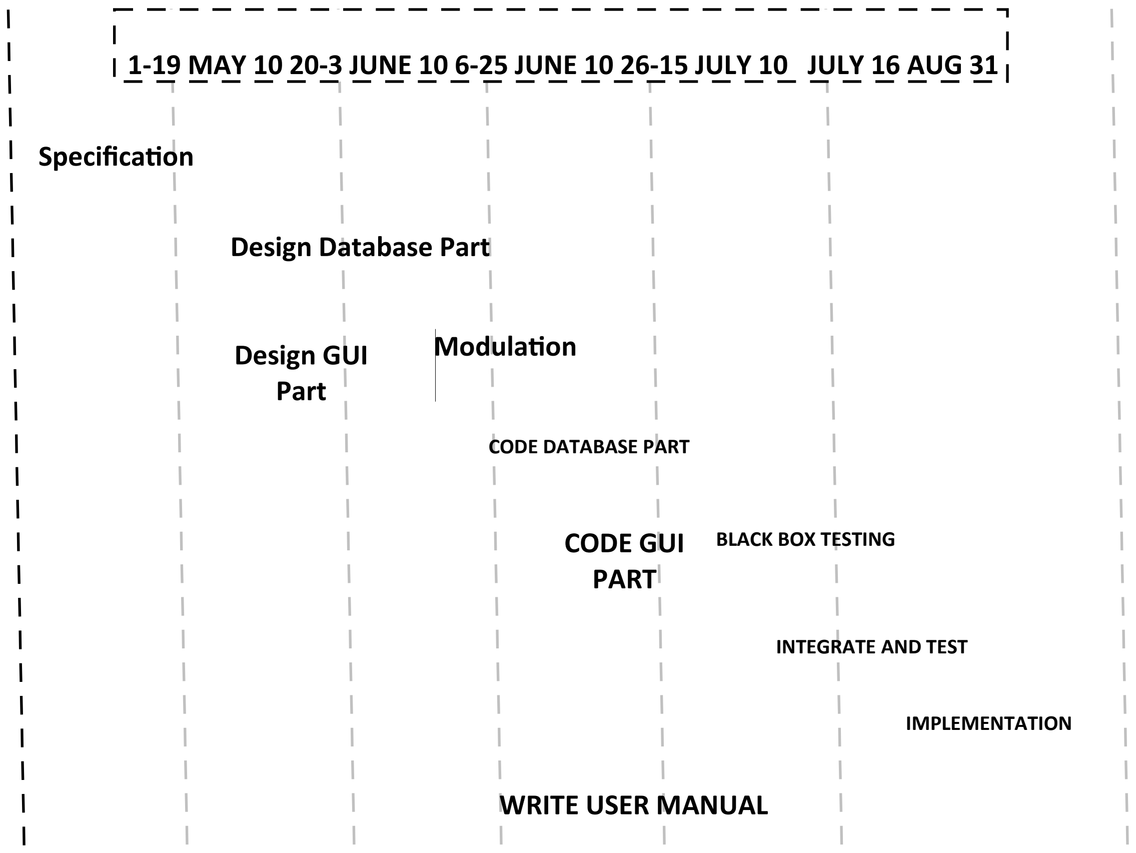
PERT chart is organized for events, activities or tasks. It is a scheduling device that shows graphically the order of the tasks to be performed. It enables the calculation of the critical path. The time and cost associated along a path is calculated and the path requires the greatest amount of elapsed time in critical path.



PERT Chart representation

GANTT CHART

It is also known as Bar chart is used exclusively for scheduling purpose. It is a project controlling technique. It is used for scheduling. Budgeting and resourcing planning. A Gantt is a bar chart with each bar representing activity. The bars are drawn against a time line. The length of time planned for the activity. The Gantt chart in the figure shows the Gray parts is slack time that is the latest by which a task has been finished.



GANTT CHART REPRESENTATION

<u>Use Case Model of the Project:</u>

The use case model for any system consists of “use cases”. Use cases represent different ways in which the system can be used by the user. A simple way to find all the use case of a system is to ask the questions “What the user can do using the system?” The use cases partition the system behavior into transactions such that each transaction performs some useful action from the users’ point of view.

The purpose of the use case to define a piece of coherent behavior without reveling the internal structure of the system. An use case typically represents a sequence of interaction between the user and the system. These interactions consists of one main line sequence is represent the normal interaction between the user and the system. The use case model is an important analysis and design artifact (task).Use cases can be represented by drawing a use case diagram and writing an accompany text elaborating the drawing.

In the use case diagram each use case is represented by an ellipse with the name of use case written inside the ellipse. All the ellipses of the system are enclosed with in a rectangle which represents the system boundary. The name of the system being moduled appears inside the rectangle. The different users of the system are represented by using stick person icon. The stick person icon is normally referred to as an Actor. The line connecting the actor and the use cases is called the communication relationship. When a stick person icon represents an external system it is annotated by the stereo type<<external system>>.

<u>Dataflow Diagram:</u>

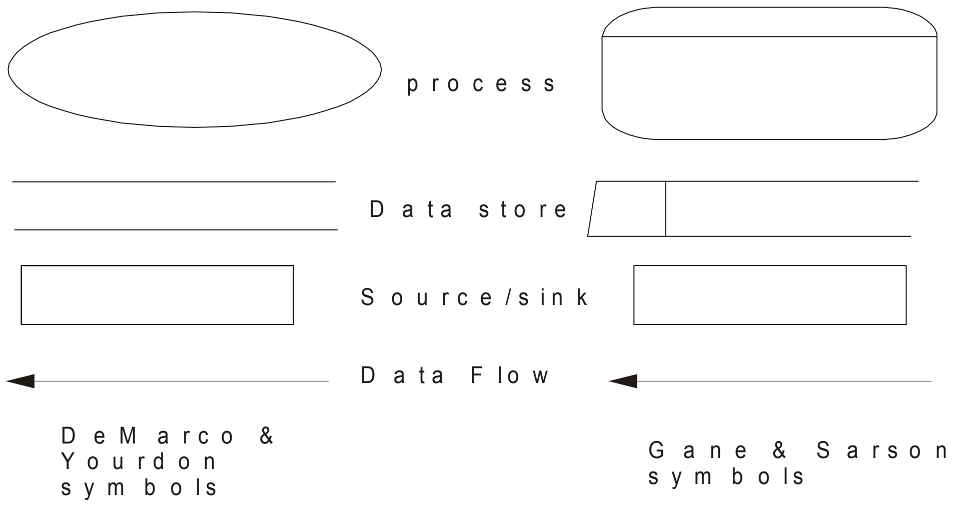
Data flow diagram is the starting point of the design phase that functionally decomposes the requirements specification. A DFD consists of a series of bubbles joined by lines. The bubbles represent data transformation and the lines represent data flows in the system. A DFD describes what data flow rather than how they are processed, so it does not hardware, software and data structure.

A **data-flow diagram** (**DFD**) is a graphical representation of the "flow" of data through an information system. DFDs can also be used for the visualization of data processing (structured design). A **data flow diagram** (DFD) is a significant modeling technique for analyzing and constructing information processes. DFD literally means an illustration that explains the course or movement of information in a process. DFD illustrates this flow of information in a process based on the inputs and outputs. A DFD can be referred to as a Process Model.

The data flow diagram is a graphical description of a system's data and how to

Process transform the data is known as Data Flow Diagram (DFD).

Unlike details flow chart, DFDs don't supply detail descriptions of modules that graphically describe a system's data and how the data interact with the system. Data flow diagram number of symbols and the following symbols are of by DeMarco.



<u>There are seven rules for construct a data flow diagram.</u>

i) Arrows should not cross each other.
ii) Squares, circles and files must wears names.
iii) Decomposed data flows must be balanced.
iv) No two data flows, squares or circles can be the same names.
v) Draw all data flows around the outside of the diagram.
vi) Choose meaningful names for data flows, processes & data stores.
vii) Control information such as record units, password and validation requirements are not penitent to a data flow diagram.

Additionally, a **DFD** can be utilized to visualize data processing or a structured design.

This basic DFD can be then disintegrated to a lower level diagram demonstrating smaller steps exhibiting details of the system that is being modeled.

On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process. It is common practice to draw a context-level data flow diagram first, which shows the interaction between the system and external agents, which act as data sources and data sinks. On the context diagram (also known as the Level 0 DFD'), the system's interactions with the outside world are modeled purely in terms of data flows across the system boundary. The context diagram shows the entire system as a single process, and gives no clues as to its internal organization.

This context-level DFD is next "exploded", to produce a Level 1 DFD that shows some of the detail of the system being modeled. The Level 1 DFD shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. The level 1 DFD is further spreaded and split into more descriptive and detailed description about the project as level 2 DFD.The level 2 DFD can be a number of data flows which will finally show the entire description of the software project.

<u>About ER Diagram:</u>

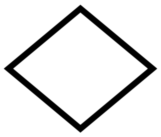
<u>Entity Relationship Diagram</u>

E-R Model is a popular high level conceptual data model. This model and its variations are frequently used for the conceptual design of database application and many database design tools employ its concept.

A database that confirms to an E-R diagram can be represented by a collecton of tables in the relational system. The mapping of E-R diagram to the entities are:

- Attributes
- Relations
  - Many-to-many
  - Many-to-one
  - One-to-many
  - One-to-one
- Weak entities
- Sub-type and super-type

The entities and their relationshops between them are shown using the following conventions.

- An entity is shown in rectangle.
- A diamond represent the relationship among number of entities. 
- The attributes shown as ovals are connected to the entities or relationship by lines.

- Diamond,oval and relationships are labeled.

  - Model is an abstraction process that hides super details while highlighting details relation to application at end.
  - A data model is a mechanism that provides this abstraction for database application.
  - Data modeling is used for representing entities and their relationship in the database.
  - Entities are the basic units used in modeling database entities can have concrete existence or constitute ideas or concepts.
  - Entity type or entity set is a group of similar objects concern to an organization for which it maintain data,
  - Properties are characteristics of an entity also called as attributes.
  - A key is a single attribute or combination of 2 or more attributes of an entity set is used to identify one or more instances of the set.
  - In relational model we represent the entity by a relation and use tuples to represent an instance of the entity.
  - Relationship is used in data modeling to represent in association between an entity set.
  - An association between two attributes indicates that the values of the associated attributes are independent.

Security Testing of the Project

Testing is vital for the success of any software. no system design is ever perfect. Testing is also carried in two phases. first phase is during the software engineering that is during the module creation. second phase is after the completion of software. this is system testing which verifies that the whole set of programs hanged together.

White Box Testing:

In this technique, the close examination of the logical parts through the software are tested by cases that exercise species sets of conditions or loops. all logical parts of the software checked once. errors that can be corrected using this technique are typographical errors, logical expressions which should be executed once may be getting executed more than once and error resulting by using wrong controls and loops. When the box testing tests all the independent part within a module a logical decisions on their true and the false side are exercised , all loops and bounds within their operational bounds were exercised and internal data structure to ensure their validity were exercised once.

Black Box Testing:

This method enables the software engineer to device sets of input techniques that fully exercise all functional requirements for a program. black box testing tests the input, the output and the external data. it checks whether the input data is correct and whether we are getting the desired output.

Alpha Testing:

Acceptance testing is also sometimes called alpha testing. Be spoke systems are developed for a single customer. The alpha testing proceeds until the system developer and the customer agree that the provided system is an acceptable implementation of the system requirements.

Beta Testing:

On the other hand, when a system isto be marked as a software product, another process called beta testing is often conducted. During beta testing, a system is delivered among a number of potential users who agree to use it. The customers then report problems to the developers. This provides the product for real use and detects errors which may not have been anticipated by the system developers.

Unit Testing:

Each module is considered independently. it focuses on each unit of software as implemented in the source code. it is white box testing.

Integration Testing:

Integration testing aims at constructing the program structure while at the same constructing tests to uncover errors associated with interfacing the modules. modules are integrated by using the top down approach.

Validation Testing:

Validation testing was performed to ensure that all the functional and performance requirements are met.

System Testing:

It is executing programs to check logical changes made in it with intention of finding errors. a system is tested for online response, volume of transaction, recovery from failure etc. System testing is done to ensure that the system satisfies all the user requirements.

Implementation and Software Specification Testings

Detailed Design of Implementation

This phase of the systems development life cycle refines hardware and software specifications, establishes programming plans, trains users and implements extensive testing procedures, to evaluate design and operating specifications and/or provide the basis for further modification.

Technical Design

This activity builds upon specifications produced during new system design, adding detailed technical specifications and documentation.

Test Specifications and Planning

This activity prepares detailed test specifications for individual modules and programs, job streams, subsystems, and for the system as a whole.

Programming and Testing

This activity encompasses actual development, writing, and testing of program units or modules.

User Training

This activity encompasses writing user procedure manuals, preparation of user training materials, conducting training programs, and testing procedures.

Acceptance Test

A final procedural review to demonstrate a system and secure user approval before a system becomes operational.

Installation Phase

In this phase the new Computerized system is installed, the conversion to new procedures is fully implemented, and the potential of the new system is explored.

<u>System Installation</u>

The process of starting the actual use of a system and training user personnel in its operation.

<u>Review Phase</u>

This phase evaluates the successes and failures during a systems development project, and to measure the results of a new Computerized Transystem in terms of benefits and savings projected at the start of the project.

<u>Development Recap</u>

A review of a project immediately after completion to find successes and potential problems in future work.

<u>Post-Implementation Review</u>

A review, conducted after a new system has been in operation for some time, to evaluate actual system performance against original expectations and projections for cost-benefit improvements. Also identifies maintenance projects to enhance or improve the system.

<u>THE STEPS IN THE SOFTWARE TESTING</u>

The steps involved during Unit testing are as follows:

a. Preparation of the test cases.
b. Preparation of the possible test data with all the validation checks.
c. Complete code review of the module.
d. Actual testing done manually.
e. Modifications done for the errors found during testing.
f. Prepared the test result scripts.

<u>The unit testing done included the testing of the following items:</u>

1. Functionality of the entire module/forms.
2. Validations for user input.
3. Checking of the Coding standards to be maintained during coding.

4. Testing the module with all the possible test data.
5. Testing of the functionality involving all type of calculations etc.
6. Commenting standard in the source files.

After completing the Unit testing of all the modules, the whole system is integrated with all its dependencies in that module. While System Integration, We integrated the modules one by one and tested the system at each step. This helped in reduction of errors at the time of the system testing.

<u>The steps involved during System testing are as follows:</u>

- Integration of all the modules/forms in the system.
- Preparation of the test cases.
- Preparation of the possible test data with all the validation checks.
- Actual testing done manually.
- Recording of all the reproduced errors.
- Modifications done for the errors found during testing.
- Prepared the test result scripts after rectification of the errors.

<u>The System Testing done included the testing of the following items:</u>

1. Functionality of the entire system as a whole.
2. User Interface of the system.
3. Testing the dependent modules together with all the possible test data scripts.
4. Verification and Validation testing.
5. Testing the reports with all its functionality.

After the completion of system testing, the next following phase was the Acceptance Testing. Clients at their end did this and accepted the system with appreciation. Thus, we reached the final phase of the project delivery.

<u>There are other six tests, which fall under special category. They are described below:</u>

- Peak Load Test: It determines whether the system will handle the volume of activities that occur when the system is at the peak of its processing demand. For example, test the system by activating all terminals at the same time.

- Storage Testing: It determines the capacity of the system to store transaction data on a disk or in other files.
- Performance Time Testing: it determines the length of time system used by the system to process transaction data. This test is conducted prior to implementation to determine how long it takes to get a response to an inquiry, make a backup copy of a file, or send a transmission and get a response.
- Recovery Testing: This testing determines the ability of user to recover data or re-start system after failure. For example, load backup copy of data and resume processing without data or integrity loss.
- Procedure Testing: It determines the clarity of documentation on operation and uses of system by having users do exactly what manuals request. For example, powering down system at the end of week or responding to paper-out light on printer.
- Human Factors Testing: It determines how users will use the system when processing data or preparing reports.

System Analysis:

System analysis is a process of gathering and interpreting facts, diagnosing problems and the information about the Online Food Ordering System to recommend improvements on the system. It is a problem solving activity that requires intensive communication between the system users and system developers. System analysis or study is an important phase of any system development process. The system is studied to the minutest detail and analyzed. The system analyst plays the role of the interrogator and dwells deep into the working of the present system. The system is viewed as a whole and the input to the system are identified. The outputs from the organizations are traced to the various processes. System analysis is concerned with becoming aware of the problem, identifying the relevant and decisional variables, analyzing and synthesizing the various factors and determining an optimal or at least a satisfactory solution or program of action. A detailed study of the process must be made by various techniques like interviews, questionnaires etc. The data collected by these sources must be scrutinized to arrive to a conclusion. The conclusion is an understanding of how the system functions. This system is called the existing system. Now the existing system is subjected to close study and problem areas are identified. The designer now functions as a problem solver and tries to sort out the difficulties that the enterprise faces. The solutions are given as proposals. The proposal is then weighed with the existing system analytically and the best one is selected. The proposal is presented to the user for an endorsement by the user. The proposal is reviewed on user request and suitable changes are made. This is loop that ends as soon as the user is satisfied with proposal. Preliminary study is the process of gathering and interpreting facts, using the information for further studies on the system. Preliminary study is problem solving activity that requires intensive communication between the system users and system developers. It does various feasibility studies. In these studies a rough figure of the system activities can be obtained, from which the decision about the strategies to be followed for effective system study and analysis can be taken.

Existing System of Online Food Ordering System:

In the existing system the exams are done only manually but in proposed system we have to computerize the exams using this application.

- Lack of security of data.
- More man power.
- Time consuming.
- Consumes large volume of pare work.
- Needs manual calculations.
- No direct role for the higher officials

Proposed System of Online Food Ordering System:

The aim of proposed system is to develop a system of improved facilities. The proposed system can overcome all the limitations of the existing system. The system provides proper security and reduces the manual work.

- Security of data.
- Ensure data accuracy's.
- Proper control of the higher officials.
- Minimize manual data entry.
- Minimum time needed for the various processing.
- Greater efficiency.
- Better service.
- User friendliness and interactive.
- Minimum time required.

Data Dictionary:

This is normally represented as the data about data. It is also termed as metadata some times which gives the data about the data stored in the database. It defines each data term encountered during the analysis and design of a new system. Data elements can describe files or the processes.

Following are some major symbols used in the data dictionary

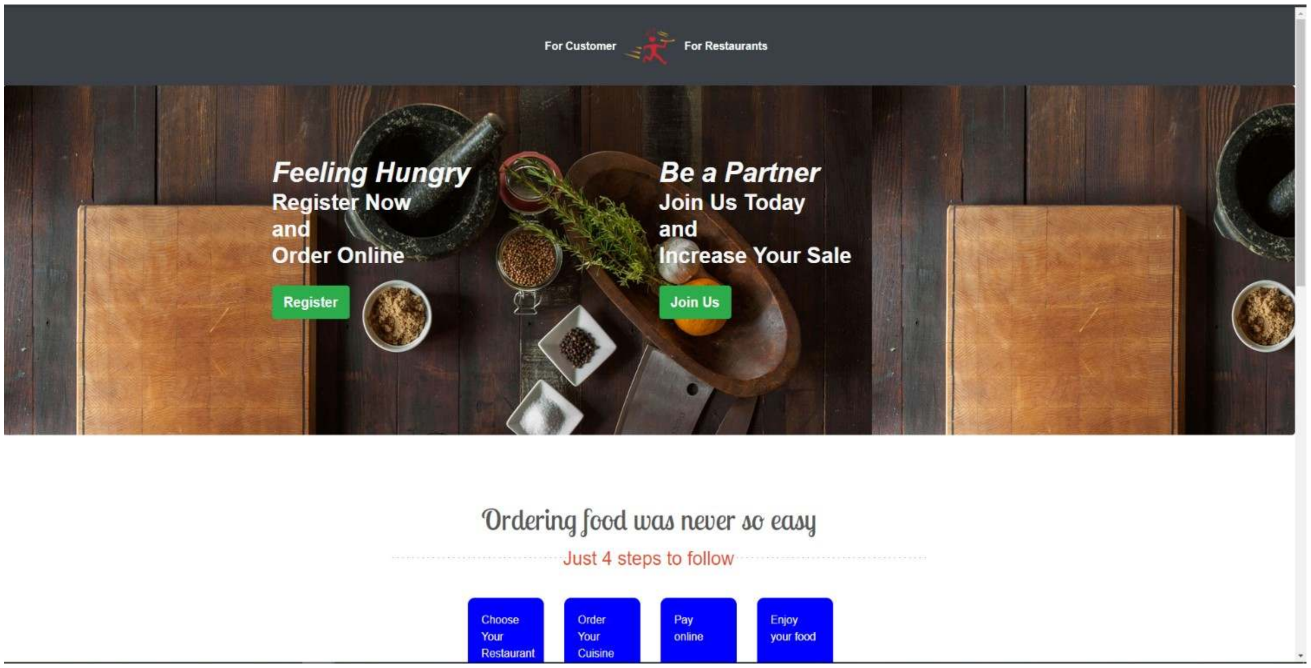
- = equivalent to
- + and
- [] either/ or
- () Optional entry

Following are some rules, which defines the construction of data dictionary entries:

1. Words should be defined to understand for what they need and not the variable need by which they may be described in the program .
2. Each word must be unique. We cannot have two definition of the same client.
3. Aliases or synonyms are allowed when two or more enters shows the same meaning. For example a vendor number may also be called as customer number.
4. A self-defining word should not be decomposed. It means that the reduction of any information in to subpart should be done only if it is really required that is it is not easy to understand directly.

Data dictionary includes information such as the number of records in file, the frequency a process will run, security factor like pass word which user must enter to get excess to the information.

Screenshot of the Project Online Food Ordering System



Feeling Hungry
Register Now
and
Order Online

Register

Be a Partner
Join Us Today
and
Increase Your Sale

Join Us

Ordering food was never so easy

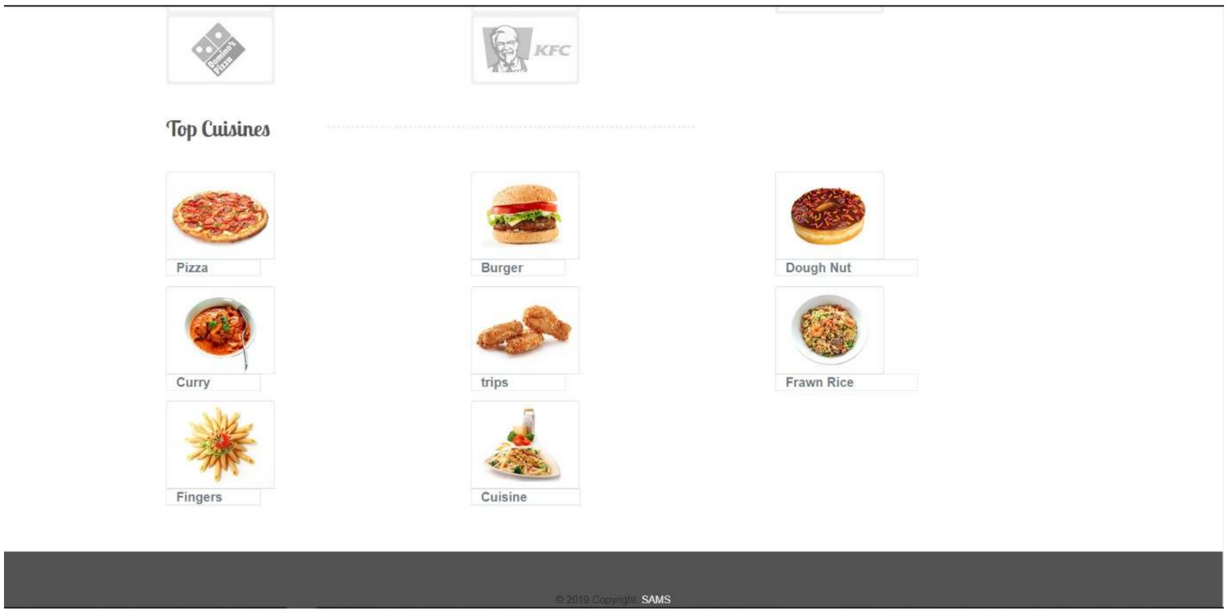
Just 4 steps to follow

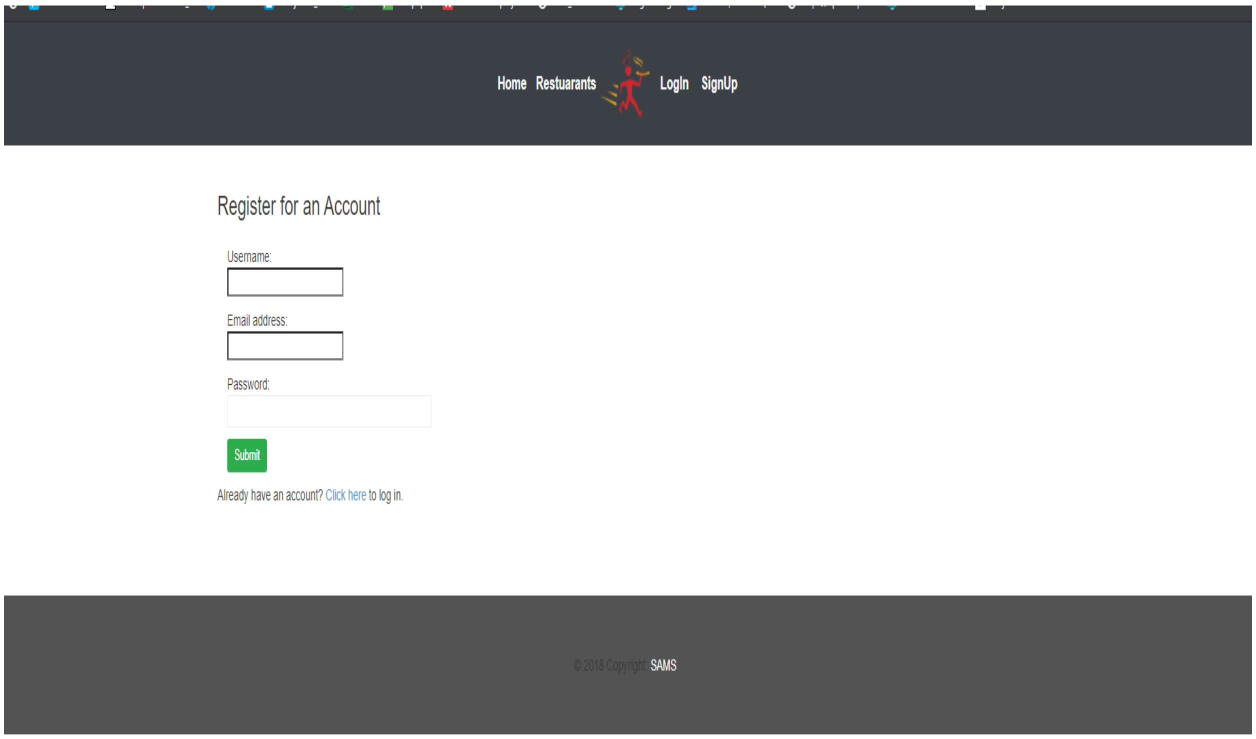
Choose Your Restaurant

Order Your Cuisine

Pay online

Enjoy your food


Top Cuisines
Pizza
Burger
Dough Nut
Curry
trips
Frawn Rice
Fingers
Cuisine
© 2019 Copyright: SAMS



Register for an Account

Username:

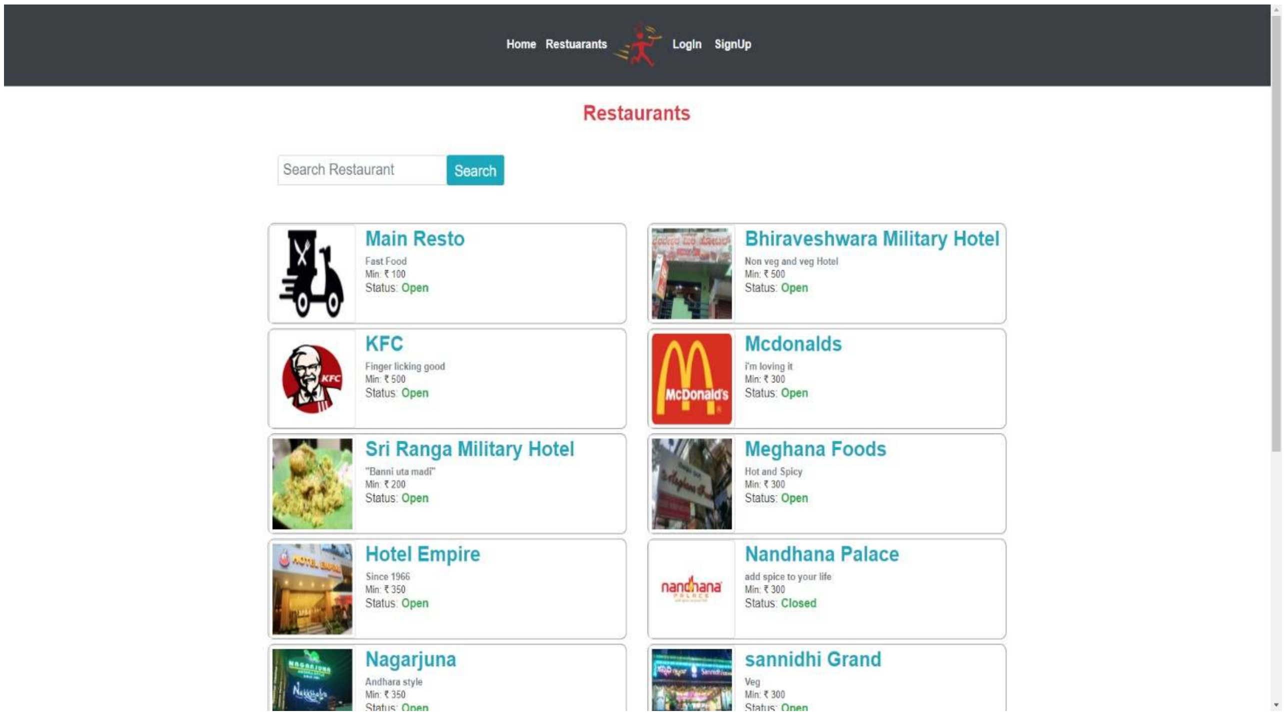
Email address:

Password:

Submit

Already have an account? Click here to log in.

© 2018 Copyright: SAMS



Restaurants

Main Resto
Fast Food
Min: ₹ 100
Status: Open

Bhiraveshwara Military Hotel
Non veg and veg Hotel
Min: ₹ 500
Status: Open

KFC
Finger licking good
Min: ₹ 500
Status: Open

Mcdonalds
i'm loving it
Min: ₹ 300
Status: Open

Sri Ranga Military Hotel
"Banni uta madi"
Min: ₹ 200
Status: Open

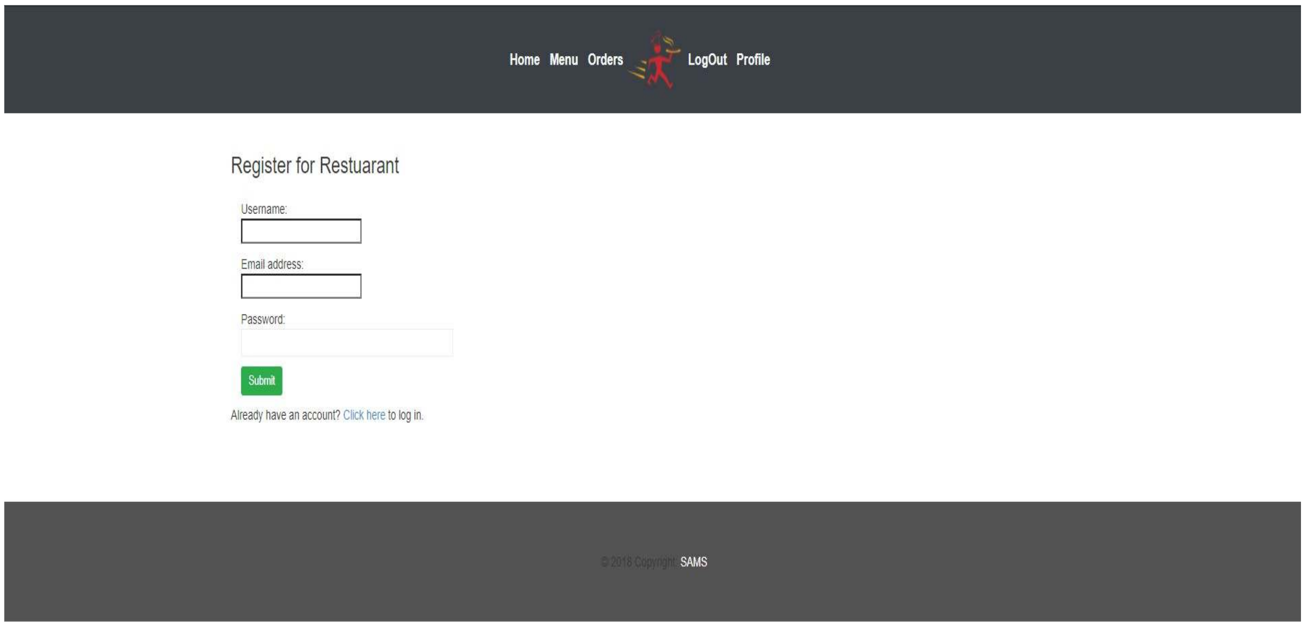
Meghana Foods
Hot and Spicy
Min: ₹ 300
Status: Open

Hotel Empire
Since 1966
Min: ₹ 350
Status: Open

Nandhana Palace
add spice to your life
Min: ₹ 300
Status: Closed

Nagarjuna
Andhara style
Min: ₹ 350
Status: Open

sannidhi Grand
Veg
Min: ₹ 300
Status: Open



Register for Restuarant

Username:

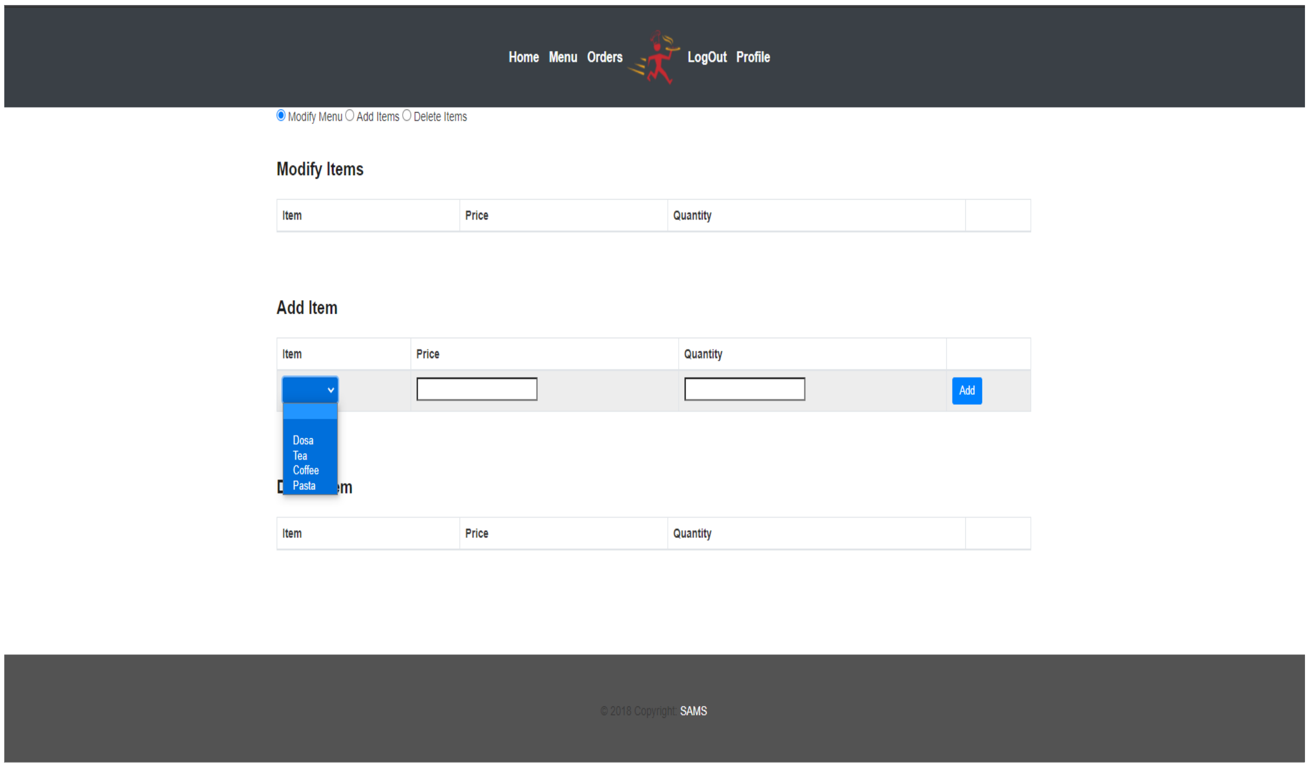
Email address:

Password:

Submit

Already have an account? Click here to log in.

© 2018 Copyright: SAMS



[✓ Modify Menu] Add Items Delete Items

Modify Items

Item	Price	Quantity	

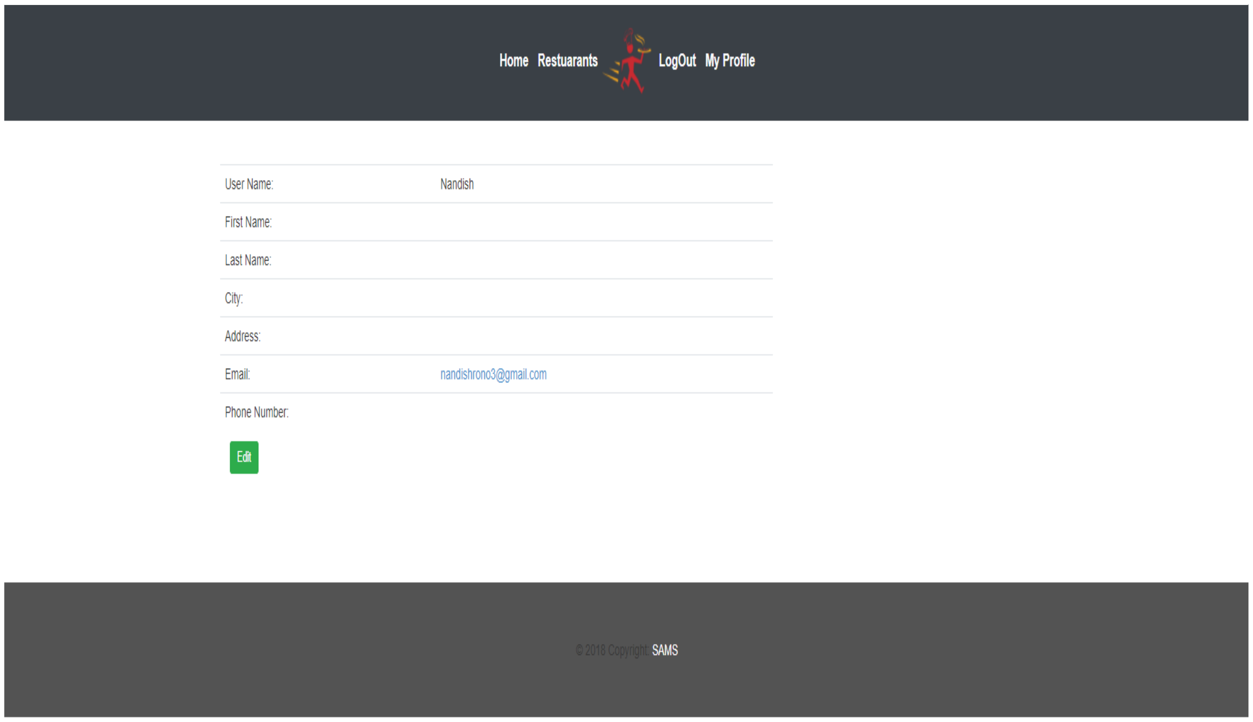
Add Item

Item	Price	Quantity	
Dosa / Tea / Coffee / Pasta			Add

D em

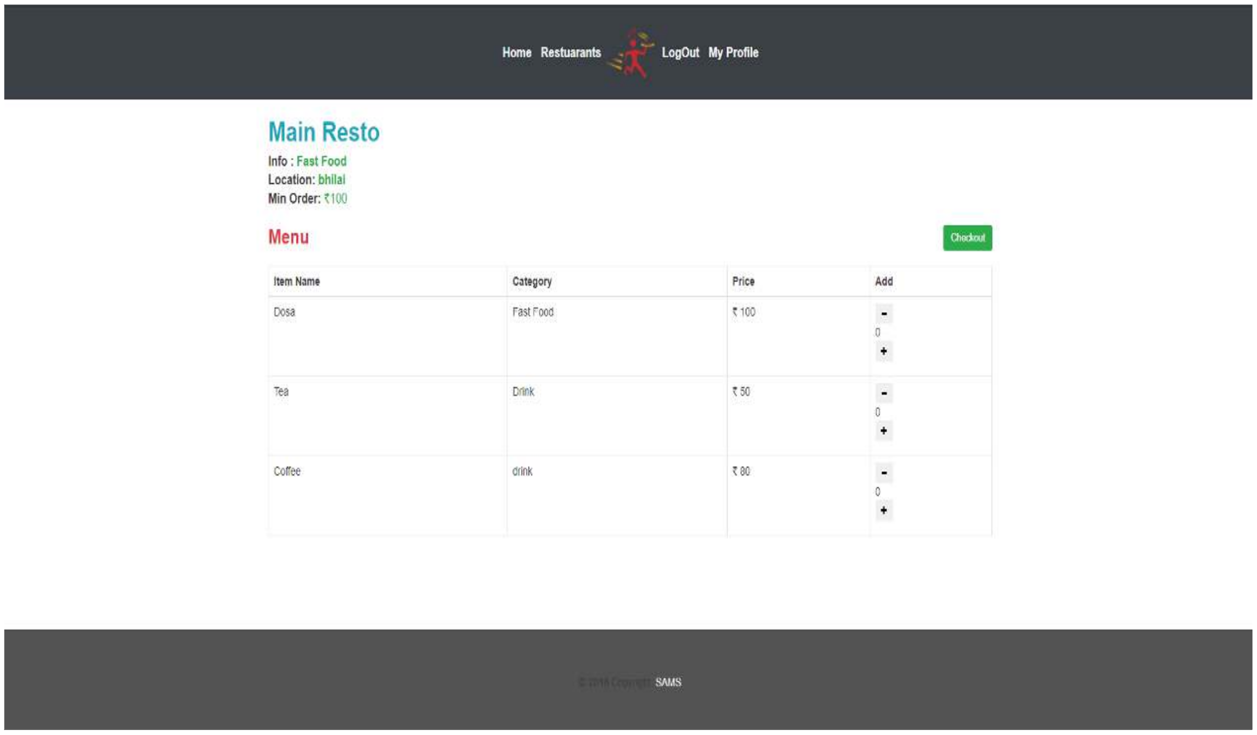
Item	Price	Quantity	

© 2018 Copyright: SAMS



User Name:	Nandish
First Name:	
Last Name:	
City:	
Address:	
Email:	nandishrono3@gmail.com
Phone Number:	

© 2018 Copyright: SAMS



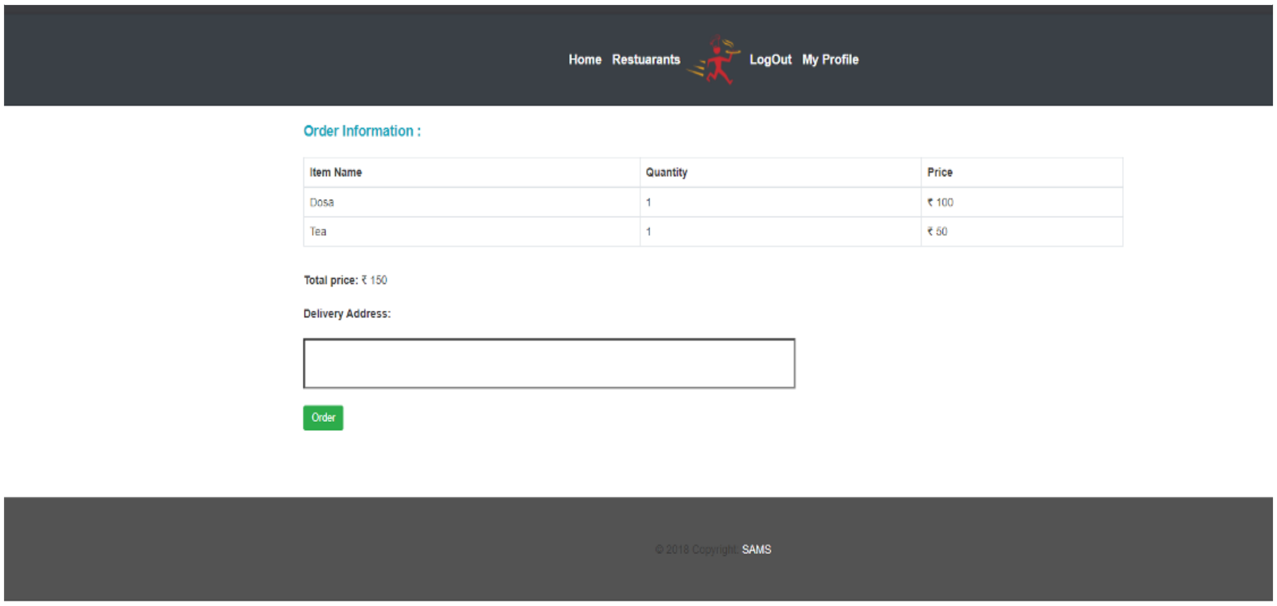
Main Resto

Info : Fast Food
Location: bhilai
Min Order: ₹100

Menu

Item Name	Category	Price	Add
Dosa	Fast Food	₹ 100	- 0 +
Tea	Drink	₹ 50	- 0 +
Coffee	drink	₹ 80	- 0 +

© 2018 Copyright: SAMS

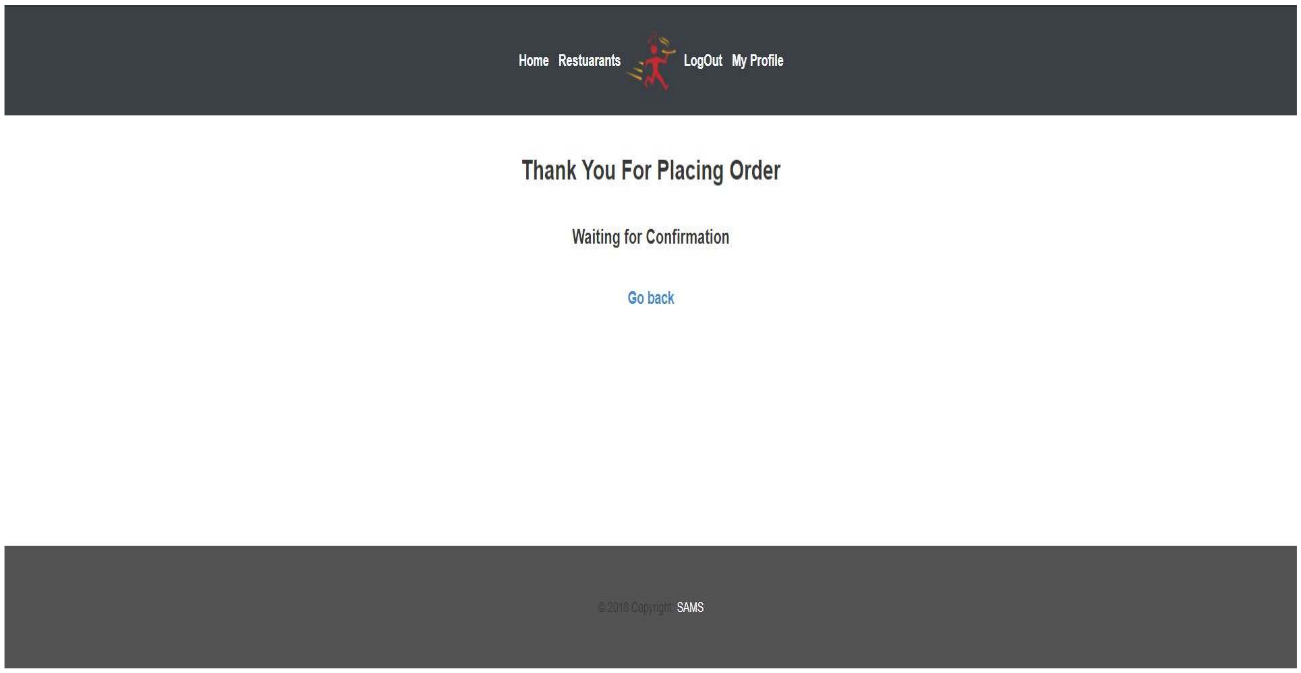


Order Information :

Item Name	Quantity	Price
Dosa	1	₹ 100
Tea	1	₹ 50

Total price: ₹ 150

Delivery Address:



Thank You For Placing Order

Waiting for Confirmation

Code of the Project Online Food Ordering System

```
<!DOCTYPE html>

<html>
```

```
<head>

  {% load staticfiles %}

 <meta charset="UTF-8">

   <meta name="viewport" content="width=device-width, initial-scale=1">

   <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">

 <title>{% block title %}FoodApp{% endblock title %}</title>

 <link rel="stylesheet" type="text/css" href="{% static
'webapp/css/bootstrap.min.css'%}">


 <link href="/static/webapp/css/style.css" rel="stylesheet" type="text/css" media="all" />

 <link rel='stylesheet' href='https://maxcdn.bootstrapcdn.com/font-
awesome/4.5.0/css/font-awesome.min.css' >

 <link href="//netdna.bootstrapcdn.com/bootstrap/3.0.0/css/bootstrap-glyphicons.css"
rel="stylesheet">



   <link href='http://fonts.googleapis.com/css?family=Open+Sans:400,300,700'
rel='stylesheet' type='text/css'>

   <link href='http://fonts.googleapis.com/css?
family=Lobster+Two:400,400italic,700,700italic' rel='stylesheet' type='text/css'>

 <!--Animation-->

 <script src="/static/webapp/js/wow.min.js"></script>

 <link href="static/webapp/css/animate.css" rel='stylesheet' type='text-center/css' />


 </head>

 <body>

         <nav class="navbar navbar-expand-lg navbar-light bg-dark">

                   <div class="container">
```

```
                        <div class="navbar-collapse collapse w-100 dual-collapse2 order-1
order-md-0">
      <ul class="navbar-nav ml-auto text-center">
         <li class="nav-item active">
            <a class="nav-link text-light" href="{% url 'index' %}"><font size="3">&nbsp;
<strong>Home</strong> </font></a>
         </li>
         <li class="nav-item">
            <a class="nav-link text-light" href="{% url 'restuarant' %}"><font
size="3">&nbsp;<strong>Restuarants</strong></font></a>
         </li>


      </ul>
   </div>
   <div class="mx-auto my-2 order-0 order-md-1 position-relative">
      <a class="mx-auto" href="#">
         <img  src="/static/webapp/images/logo.png" style="height: 90px;width: 170px;"
alt="Home" class="rounded-circle img-responsive">
      </a>
      <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target=".dual-collapse2">
         <span class="navbar-toggler-icon"></span>
      </button>
   </div>
   <div class="navbar-collapse collapse w-100 dual-collapse2 order-2 order-md-2">
      <ul class="navbar-nav mr-auto text-center">
        {% if request.user.is_authenticated %}
```

```
            <li class="nav-item">
                <a class="nav-link text-light" href="{% url 'logout' %}"><font size="3">
&nbsp;<span aria-hidden="true"></span><strong>LogOut </strong> </font></a>
            </li>
            <li class="nav-item">
                <a class="nav-link text-light" href="{% url 'profile' %}"><font size="3">
&nbsp;<span aria-hidden="true"></span><strong>My Profile</strong></font></a>
            </li>
          {% else %}
            <li class="nav-item">
                <a class="nav-link text-light" href="{% url 'login' %}"><font
size="3">&nbsp;<span aria-hidden="true"></span> <strong>LogIn</strong></font></a>
            </li>
            <li class="nav-item">
                <a class="nav-link text-light" href="{% url 'register' %}"><font
size="3">&nbsp;<span aria-hidden="true"></span>
<strong>SignUp</strong></font></a>
            </li>
            {% endif %}
        </ul>
    </div>
</div>
</nav>
{% block body %}{% endblock %}
<br>
<br>
<br>
```

```
<br>


<div class="footer">
    <div class="container-fluid">
    <div class="footer-copyright text-center py-3">© 2018 Copyright:
        <a href="http://www.sambhram.org/sams.html"> SAMS </a>
      </div>
  </div>
</div>
<script type="text/javascript" src="{%static 'webapp/js/jquery.min.js'%}">
            </script>
            <script type="text/javascript" src="{%static
'webapp/js/bootstrap.min.js'%}"></script>
<script>
            new WOW().init();
</script>
<script src="/static/webapp/js/simpleCart.min.js"> </script>
<script type="text/javascript" src="/static/webapp/js/move-top.js"></script>
<script type="text/javascript" src="/static/webapp/js/easing.js"></script>
</body>
</html>


<!DOCTYPE html>
<html>
<head>
```

```
 {% load staticfiles %}

 <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1">

  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">

 <title>{% block title %}FoodApp{% endblock title %}</title>

 <link rel="stylesheet" type="text/css" href="{% static
'webapp/css/bootstrap.min.css'%}">


 <link href="/static/webapp/css/style.css" rel="stylesheet" type="text/css" media="all" />

 <link rel='stylesheet' href='https://maxcdn.bootstrapcdn.com/font-
awesome/4.5.0/css/font-awesome.min.css' >

 <link href="//netdna.bootstrapcdn.com/bootstrap/3.0.0/css/bootstrap-glyphicons.css"
rel="stylesheet">


  <link href='http://fonts.googleapis.com/css?family=Open+Sans:400,300,700'
rel='stylesheet' type='text/css'>

  <link href='http://fonts.googleapis.com/css?
family=Lobster+Two:400,400italic,700,700italic' rel='stylesheet' type='text/css'>

 <!--Animation-->

 <script src="/static/webapp/js/wow.min.js"></script>

 <link href="static/webapp/css/animate.css" rel='stylesheet' type='text-center/css' />


 </head>

 <body>

        <nav class="navbar navbar-expand-lg navbar-light bg-dark">

                <div class="container-fluid">
```

```
                                <div class="navbar-collapse collapse w-100 dual-collapse2 order-1
order-md-0">
        <ul class="navbar-nav ml-auto text-center">
            <li class="nav-item active">
                <a class="nav-link text-light" href="{% url 'index' %}"><font size="3">&nbsp;
<strong>Home</strong> </font></a>
            </li>
            <li class="nav-item">
                <a class="nav-link text-light" href="{% url  'mmenu' %}"><font
size="3">&nbsp;<strong>Menu </strong></font></a>
            </li>
             <li class="nav-item">
                <a class="nav-link text-light" href="{% url  'orderlist' %}"><font
size="3">&nbsp;<strong>Orders</strong></font></a>
            </li>


        </ul>
    </div>
    <div class="mx-auto my-2 order-0 order-md-1 position-relative">
        <a class="mx-auto" href="#">
            <img  src="/static/webapp/images/logo.png" style="height: 90px;width: 170px;"
alt="Home" class="rounded-circle img-responsive">
        </a>
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target=".dual-collapse2">
            <span class="navbar-toggler-icon"></span>
        </button>
    </div>

    <div class="navbar-collapse collapse w-100 dual-collapse2 order-2 order-md-2">

        <ul class="navbar-nav mr-auto text-center">

          {% if request.user.is_authenticated %}

            <li class="nav-item">

                <a class="nav-link text-light " href="{% url 'logout' %}"><font size="3">
&nbsp;<span aria-hidden="true"></span><strong>LogOut </strong> </font></a>

            </li>

            <li class="nav-item">

                <a class="nav-link text-light " href="{%url 'rprofile' %}"><font size="3">
&nbsp;<span aria-hidden="true"></span><strong>Profile</strong></font></a>

            </li>

          {% else %}

            <li class="nav-item">

                <a class="nav-link text-light " href="{% url 'rlogin' %}"><font
size="3">&nbsp;<span aria-hidden="true"></span> <strong>LogIn</strong></font></a>

            </li>

            <li class="nav-item">

                <a class="nav-link text-light " href="{% url 'rregister' %}"><font
size="3">&nbsp;<span aria-hidden="true"></span> <strong>Join
Us</strong></font></a>

            </li>

            {% endif %}

        </ul>

    </div>

</div>

</nav>
```

```
{% block body %}
{% endblock %}
<br>
<br>
<br>
<br>

<div class="footer">
   <div class="container-fluid">
   <div class="footer-copyright text-center py-3">© 2018 Copyright:
      <a href="http://www.sambhram.org/sams.html"> SAMS </a>
     </div>

  </div>
</div>

<script type="text/javascript" src="{%static 'webapp/js/jquery.min.js'%}">
        </script>
<script type="text/javascript" src="{%static 'webapp/js/bootstrap.min.js'%}"></script>
<script>
        new WOW().init();
</script>
<script src="/static/webapp/js/simpleCart.min.js"> </script>
<script type="text/javascript" src="/static/webapp/js/move-top.js"></script>
```

```
<script type="text/javascript" src="/static/webapp/js/easing.js"></script>

</body>

</html>


{% for field in form %}

    <div class="form-group">

        <div class="col-sm-offset-2 col-sm-12">

            <span class="text-danger small">{{ field.errors }}</span>

        </div>

        <label class="control-label col-sm-4">{{ field.label_tag }}</label>

        <div class="col-sm-7">{{ field }}</div>

    </div>

{% endfor %}


<!DOCTYPE html>

<html>

<head>

   {% load staticfiles %}

  <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">

  <title>{% block title %}FoodApp{% endblock title %}</title>

  <link rel="stylesheet" type="text/css" href="{% static
'webapp/css/bootstrap.min.css'%}">
```

```
  <link href="/static/webapp/css/style.css" rel="stylesheet" type="text/css" media="all" />

  <link rel='stylesheet' href='https://maxcdn.bootstrapcdn.com/font-
awesome/4.5.0/css/font-awesome.min.css' >

    <link href='http://fonts.googleapis.com/css?family=Open+Sans:400,300,700'
rel='stylesheet' type='text/css'>

    <link href='http://fonts.googleapis.com/css?
family=Lobster+Two:400,400italic,700,700italic' rel='stylesheet' type='text/css'>

  <!--Animation-->

  <script src="/static/webapp/js/wow.min.js"></script>

  <link href="static/webapp/css/animate.css" rel='stylesheet' type='text-center/css' />


<style type="text/css">

  .big-banner{

    background-image: url('/static/webapp/images/slide_1.jpg');

  }

  .i-am-center{

    margin-left:26%; max-width: 100%;

  }

  .borders{

    background-color: blue;

    border-radius: 10px;

    padding: 20px;

  }

  .wrapper {

    min-height: 100%;

    height: 100%;
    margin: 0 auto -155px;
}
  .footer, .push {
    height: 155px;
}
</style>
</head>
<body>

    <nav class="navbar navbar-expand-lg navbar-light bg-dark navbar-fixed-top">
                    <div class="container-fluid">
                              <div class="navbar-collapse collapse w-100 dual-collapse2 order-1
order-md-0">
        <ul class="navbar-nav ml-auto text-center">
            <li class="nav-item active">
                <a class="nav-link text-light" href="{% url 'restuarant'%}">&nbsp; <strong>For
Customer</strong> </a>
            </li>
        </ul>
    </div>
    <div class="mx-auto my-2 order-0 order-md-1 position-relative">
        <a class="mx-auto" href="#">
            <img  src="/static/webapp/images/logo.png" style="height: 80px;" alt="Home"
class="rounded-circle img-responsive">
        </a>
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target=".dual-collapse2">
```

```
            <span class="navbar-toggler-icon"></span>

        </button>

    </div>

    <div class="navbar-collapse collapse w-100 dual-collapse2 order-2 order-md-2">

        <ul class="navbar-nav mr-auto text-center">

            <li class="nav-item">

                <a class="nav-link text-light" href="{% url 'rlogin'%}"> &nbsp;<strong>For
Restaurants</strong> </a>

            </li>

        </ul>

    </div>

</div>


</nav>


<div class="jumbotron big-banner" style="height: 500px; width: 100%; padding-
top:100px;" >

<div class="container">

<div class="row">

<div class="col">

<div class="col4 text-light ">

<h1 style="font-style:italic; font-family:sans-serif; "><strong>Feeling
Hungry</strong></h1>

<h2><strong>Register Now<br/>and<br/>Order Online</strong></h2>

</div>

<br>
```

```
<button type="button" class="btn btn-success btn-lg"><strong><a href="{% url
'register'%}" class="text-light">Register</a></strong></button>

</div>

<div class="col">

<div class="col4 text-light">

<h1 style="font-style:italic; font-family:sans-serif; "><strong>Be a Partner</strong></h1>

<h2><strong>Join Us Today<br/>and<br>Increase Your Sale</strong></h2>

</div>

<br>

<button type="button" class="btn btn-success btn-lg"><strong><a href="{% url
'rregister'%}" class="text-light">Join Us</a></strong></button>

</div>

</div>

</div>


</div>

</div>


<div class="content">

<div class="ordering-section" id="Order">

<div class="container">

<div class="ordering-section-head text-center wow bounceInRight" data-wow-
delay="0.4s">

<h3>Ordering food was never so easy</h3>

<div class="dotted-line">

<h4>Just 4 steps to follow </h4>
```

```
</div>

</div>

<div class="i-am-center">

<div class="row ">

<div class="col-sm-2 col-md-2">

<div class="thumbnail borders text-light">

<p>Choose <span>Your Restaurant</span></p>

</div>

</div>

<div class="col-sm-2 col-md-2 ">

<div class="thumbnail borders text-light">


<p>Order  <span>Your Cuisine</span></p>

</div>

</div>

<div class="col-sm-2 col-md-2">

<div class="thumbnail borders text-light" >

<p>Pay   <span> online </span></p>

<label></label>

</div>

</div>

<div class="col-sm-2 col-md-2">

<div class="thumbnail borders text-light">

<p>Enjoy <span>your food </span></p>

<label></label>
```

```
</div>

</div>

<div class="clearfix"></div>

</div>

</div>

</div>

</div>

</div>


<div class="container col-md-9">

<div class="top-restaurent-head">

<h3>Top Restaurants</h3>

</div>

<div class="row">


<div class="col-md-4 ">

<div class="thumbnail">

<a href="">

<img src="/static/webapp/images/restaurent-2.jpg" class="img-responsive" alt="" />

</a>


</div>

</div>

<br>
```

```
<div class="col-md-4 ">

<div class="thumbnail">

<a href="">

<img src="/static/webapp/images/restaurent-3.jpg" class="img-responsive" alt="" />

</a>


</div>

</div>

<br>

<div class="col-md-4 ">

<div class="thumbnail">

<a href="">

<img src="/static/webapp/images/restaurent-4.jpg" class="img-responsive" alt="" />

</a>


</div>

</div>

<br>

<div class="col-md-4 ">

<div class="thumbnail">

<a href="">

<img src="/static/webapp/images/restaurent-5.jpg" class="img-responsive" alt="" />

</a>


</div>
```

```
</div>

<br>

<div class="col-md-4 ">

<div class="thumbnail">

<a href="">

<img src="/static/webapp/images/restaurent-6.jpg" class="img-responsive" alt="" />

</a>


</div>

</div>


</div>


</div>


<br>

<br>

<div class="container col-md-9">

        <div class="top-cuisine-head">

                        <h3>Top Cuisines</h3>

        </div>

        <div class="row">

                <div class="col-md-4 ">

               <div class="thumbnail">

                <a href="">
```

```
                    <img src="/static/webapp/images/cuisine1.jpg" class="img-responsive"
alt="" />
                   </a>
                   </div>
                   <div class="caption border col-sm-4">
                           <h5 class="text-secondary"><strong>Pizza</strong></h5>
                   </div>
               </div>
                <div class="col-md-4 mb-3">
                   <div class="thumbnail">
                    <a href="">
                    <img src="/static/webapp/images/cuisine2.jpg" class="img-responsive"
alt="" />
                   </a>
                   </div>
                   <div class="caption border col-sm-4">
                           <h5 class="text-secondary"><strong>Burger</strong></h5>
                   </div>
               </div>


                <div class="col-md-4 ">
                   <div class="thumbnail">
                    <a href="">
                    <img src="/static/webapp/images/cuisine3.jpg" class="img-responsive"
alt="" />
                   </a>
                   </div>

                   <div class="caption border col-sm-6">

                           <h5 class="text-secondary"><strong>Dough Nut</strong></h5>

                   </div>

               </div>

                <div class="col-md-4 mb-3">

                   <div class="thumbnail">

                    <a href="">

                    <img src="/static/webapp/images/cuisine4.jpg" class="img-responsive"
alt="" />

                   </a>

                   </div>

                   <div class="caption border col-sm-4">

                           <h5 class="text-secondary"><strong>Curry</strong></h5>

                   </div>

               </div>

                <div class="col-md-4 ">

                   <div class="thumbnail">

                    <a href="">

                    <img src="/static/webapp/images/cuisine5.jpg" class="img-responsive"
alt="" />

                   </a>

                   </div>

                   <div class="caption border col-sm-4">

                           <h5 class="text-secondary"><strong>trips</strong></h5>

                   </div>
            </div>
             <div class="col-md-4 ">
                  <div class="thumbnail">
                    <a href="">
                    <img src="/static/webapp/images/cuisine6.jpg" class="img-responsive"
alt="" />
                  </a>
                  </div>
                  <div class="caption border col-md-6">
                              <h5 class="text-secondary"><strong>Frawn Rice</strong></h5>
                  </div>
            </div>
             <div class="col-md-4 ">
                  <div class="thumbnail">
                    <a href="">
                    <img src="/static/webapp/images/cuisine7.jpg" class="img-responsive"
alt="" />
                  </a>
                  </div>
                  <div class="caption border col-md-4">
                              <h5 class="text-secondary"><strong>Fingers</strong></h5>
                  </div>
            </div>
                 <div class="col-md-6 ">
                  <div class="thumbnail">
```

```
                    <img src="/static/webapp/images/cuisine8.jpg" class="img-responsive"
alt="" />

                   <div class="caption border col-md-3">

                      <h5 class="text-secondary"><strong>Cuisine</strong></h5>

                   </div>

                   </div>


               </div>

               <div class="clearfix"></div>

           </div>

</div>


<br>

<br>

<br>

<div class="footer">

                    <div class="container">

                           <div class="footer-copyright text-center py-3">© 2019 Copyright:

       <a href="http://www.sambhram.org/sams.html"> SAMS </a>


                    </div>

           </div>



<script type="text/javascript" src="{%static 'webapp/js/jquery.min.js'%}">
```

```
        </script>

        <script type="text/javascript" src="{%static
'webapp/js/bootstrap.min.js'%}"></script>

<script>

        new WOW().init();

</script>

<script src="/static/webapp/js/simpleCart.min.js"> </script>

<script type="text/javascript" src="/static/webapp/js/move-top.js"></script>

<script type="text/javascript" src="/static/webapp/js/easing.js"></script>

</body>

</html>


{% extends 'webapp/base.html'%}

{% block title %}Log In{% endblock %}


{% block body %}

<div class="container col-md-8">


   <div class="row">

      <div class="col-sm-12 col-md-6 col-md-offset-3">

         <div class="panel panel-default">

<div class="panel-body">

<br>

<br>

<br>
```

```
<br>

<h3 class="col-sm-4 col-md-offset-4">Log In</h3>

{% if error_message %}

<p><strong>{{ error_message }}</strong></p>

{% endif %}

<br>

<form class="form-horizontal" role="form" action="#" method="post"
enctype="multipart/form-data">

{% csrf_token %}

<div class="form-group">

<label class="control-label col-sm-2" for="id_username">

Username:

</label>

<div class="col-sm-6">

<input id="id_username" maxlength="30" name="username" type="text">

</div>

</div>

<div class="form-group">

<label class="control-label col-sm-2" for="id_password">

Password:

</label>

<div class="col-sm-7">

<input id="id_password" maxlength="30" name="password" type="password">

</div>

</div>
```

```
<div class="form-group">

<div class="col-sm-offset-2 col-sm-10">

<button type="submit" class="btn btn-success">Submit</button>

</div>

</div>

</form>

</div>

<div class="panel-footer">

Don't have an account? <a href="{% url 'register' %}">Click here</a> to register.

</div>

</div>

</div>

</div>

</div>

<br>

<br>

{% endblock %}


{% extends 'webapp/base.html'%}

{% block title %}Menu{% endblock %}


{% block body %}

<div class="container">

<h1 class="text-info mb-2"><font size="6"><strong>{{rname}}</strong></font></h1>

<p>
```

```
<font size="3"><b>Info : </b><a class="text-success"><strong>{{rinfo|
safe}}</strong></a>

<br/>

<b>Location: </b><a class="text-success"><strong>{{rlocation}}</strong></a>

<br>

<b >Min Order:</b> <a class="text-success">&#x20b9;{{rmin}}</a>

</font>

</p>

</div>


<br>

<div class="container">

<input type="button" class="btn btn-success btn-lg pull-right" onclick="storearray();"
value="Checkout"/>

</div>

<div class="container ">

<h3 class="text-danger"><font size="5"><strong>Menu</strong></font></h3>

<br>

<table class="table table-hover table-bordered">

<thead>

<th>Item Name</th>

<th>Category</th>

<th>Price</th>

<th>Add</th>

</thead>

{% for x in items %}
```

```
<tr>

<td>{{x.0}}</td>

<td>{{x.1}}</td>

<td>&#x20b9; {{x.2}}</td>

{% if x.4 == "Open" %}

{% if x.5 > 0 %}

<td>

<div class="form-group" style="width:40%;">


<div class="input-group-btn">

<button id="down" class="btn btn-default"

onclick="if(document.querySelector('#itemno{{x.3}}').innerHTML>0){

document.querySelector('#itemno{{x.3}}').innerHTML--;

minuscounter('{{x.3}}');

}">

<span class="glyphicon glyphicon-minus"></span></button>

</div>

<span id='itemno{{x.3}}'> 0 </span>

<div class="input-group-btn">

<button id="up" class="btn btn-default"

onclick="

if(document.querySelector('#itemno{{x.3}}').innerHTML<{{x.5}}){

document.querySelector('#itemno{{x.3}}').innerHTML++;

pluscounter('{{x.3}}');

}"><span class="glyphicon glyphicon-plus"></span></button>
```

```
</div>

</div>

</td>

{% else %}

<td>

NA

</td>

{% endif %}

{% else %}

<td>

Closed

</td>

{% endif %}

</tr>

{% endfor %}

</tr>

</table>

</div>

<br>

<script type="text/javascript">


var cart = [];

        function pluscounter(pk){

                cart.push(pk);

                console.log(cart);
        };
        function minuscounter(pk){
                if (cart!=[]){
                        var removeindex = cart.indexOf(pk);
                        cart.splice(removeindex, 1);
                        }
                        console.log(cart);
        };

        function storearray(){
                if(cart.length === 0)
                {
                        alert('No items selected');
                        return false;
                }
                document.cookie = "cart="+cart+";Path=/";
                document.cookie ="rest={{restid}};Path=/";
                window.location = "/checkout/";
        };
</script>
{% endblock %}

{% extends 'webapp/base.html'%}
{% block title %} Cart {% endblock %}
{% block body %}

<div class="container ">

<br>

<h3 class="text-info"><strong>Order Information :</strong></h3>

<br>

<table class="table table-hover table-bordered">

<thead>

<th>Item Name</th>

<th>Quantity</th>

<th>Price</th>

</thead>

{% for x in items %}

<tr><td>{{x.0}}</td><td>{{x.1}}</td><td>&#x20b9; {{x.2}}</td></tr>

{% endfor %}


</table>

<br>

<span><b>Total price: </b></span><span> &#x20b9; {{totalprice}}

</span>

<br>

<br>

<form action="" method="POST">

{% csrf_token %}

<span><strong>Delivery Address:</strong></span>

<br>
```

```
<br>

<div><input type="text" name="address" style="width:60%; height: 60px;"
required/></div>

<input type="hidden" name="oid" value="{{oid}}">

<br>

<input type="submit" class="btn btn-success btn-lg" value="Order" name="submit">

</form>

</div>


{% endblock %}




{% extends 'webapp/base2.html' %}

{% block title %}Order List{% endblock %}


{% block body %}

{% for order in orders %}

      <div class="container">

          <div class="col-md-12 restaurent-title">

            <div class="logo-title" style="border: 0;">

              <h4>{{order.0}} - {{order.1}} - {{order.6}}</h4>

            </div>

          </div>

          <div class="col-md-12 order-form-grid  wow fadeInLeft">

            <h5>Order Information</h5>
```

```
    <span>Items List:</span>
    <div>
        <table class="table table-hover table-bordered">
            <thead>
                <th>
                    Name
                </th>
                <th>
                    Quantity
                </th>
                <th>
                    Price
                </th>
            </thead>
            {% for x in order.2 %}
                <tr><td>{{x.0}}</td><td>{{x.1}}</td><td>&#x20b9;{{x.2}}</td></tr>
            {% endfor %}
        </table>
    </div>
</div>
<div class="col-md-12 buy"  style="display: inline-block;">
    <form action="" method="POST">{% csrf_token %}
    <span>Total price: &#x20b9;{{order.3}}</span>
    <span  style="display: inline;">Status:</span>
    <select name="orderstatus">
```

```
{% if order.5 == 1 %}
<option value="1" selected>Placed</option>
<option value="2">Acknowledged</option>
<option value="3">Completed</option>
<option value="4">Dispatched</option>
<option value="5">Cancelled</option>
{% endif %}
{% if order.5 == 2 %}
<option value="1">Placed</option>
<option value="2" selected>Acknowledged</option>
<option value="3">Completed</option>
<option value="4">Dispatched</option>
<option value="5">Cancelled</option>
{% endif %}
{% if order.5 == 3 %}
<option value="1">Placed</option>
<option value="2">Acknowledged</option>
<option value="3" selected>Completed</option>
<option value="4">Dispatched</option>
<option value="5">Cancelled</option>
{% endif %}
{% if order.5 == 4 %}
<option value="1">Placed</option>
<option value="2">Acknowledged</option>
<option value="3">Completed</option>
```

```
                    <option value="4" selected>Dispatched</option>
                    <option value="5">Cancelled</option>
                    {% endif %}
                    {% if order.5 == 5 %}
                    <option value="1">Placed</option>
                    <option value="2">Acknowledged</option>
                    <option value="3">Completed</option>
                    <option value="4">Dispatched</option>
                    <option value="5" selected>Cancelled</option>
                    {% endif %}
                  </select>
                  <input type="hidden" name="orderid" value="{{order.4}}">
                  <input type="submit" value="Set" name="submit"/>
                  </form>
                </div>
                <div class="clearfix"></div>



          </div>
      {% endfor %}
      {% endblock %}
      {% extends 'webapp/base.html'%}
      {% block title %}orderplaced{% endblock %}

      {% block body %}
```

```
<div class="container">

<br>

<h1 class="text-center"><strong>Thank You For Placing Order</strong></h1>

<br>

<h2 class="text-center"><strong>Waiting for Confirmation</strong></h2>

<br>

<br>

<h3 class="text-center"><a href="{% url 'restuarant'%}"><strong>Go
back</strong></a></h3>

</div>

<br>

<br>

<br>

<br>

<br>

<br>

<br>

<br>
{% endblock%}



{% extends 'webapp/base.html'%}

{% block title %}Profile{% endblock %}


{% block body %}
```

```
<br>

<div class="container col-md-8">


<div class="row">

<div class=" col-md-11 toppad">


<div class="panel panel-info">

<div class="panel-heading">

<h3 class="panel-title text-info "><strong>{{user.customer.f_name}}
{{user.customer.l_name}}</strong></h3>

</div>

<br>

<div class="panel-body">

<div class="row">


<div class=" col-md-9 col-lg-9 ">

<table class="table table-user-information">

<tbody>

<tr>

<td>User Name:</td>

<td>{{user.username}}</td>

</tr>

<tr>

<td>First Name:</td>

<td>{{user.customer.f_name}}</td>
```

```
</tr>
<tr>
<td>Last Name:</td>
<td>{{user.customer.l_name}}</td>
</tr>
<tr>
<td>City:</td>
<td>{{user.customer.city}}</td>
</tr>

<td>Address:</td>
<td>{{user.customer.address}}</td>
</tr>
<tr>
<td>Email:</td>
<td><a href="#">{{user.email}}</a></td>
</tr>
<tr>
<td>Phone Number:</td>
<td>{{user.customer.phone}}
</td>

</tr>

</tbody>
```

```
</table>

<div class="container">

<div class="row">

<div class="col">

<a class="btn btn-success btn-lg" href="{% url 'cupdate' user.id %}">Edit</a>

</div>

<!--<div class="col">

<a class="btn btn-success btn-lg" href="{%url 'ccreate' %}">Complete profile</a>

</div>-->

</div>

</div>


</div>

</div>

</div>

</div>

</div>

</div>

</div>

<br>

{% endblock%}

{% extends 'webapp/base2.html'%}

{% block title %}Profile{% endblock %}


{% block body %}
```

```
<br>
<div class="container col-md-8">
<img src="{{user.restaurant.r_logo.url}}" height="180px" width="60%">
</div>
<br>
<div class="container col-md-8">

<div class="row">
<div class=" col-md-11 toppad">

<div class="panel panel-info">
<div class="panel-heading">
<h1 class="panel-title text-info "><strong>{{user.restaurant.rname}}</strong></h1>
</div>

<div class="panel-body">
<div class="row">

<div class=" col-md-9 col-lg-9 ">
<table class="table table-user-information">
<tbody>
<tr>
<td> Restaurant UserName:</td>
<td>{{user.username}}</td>
</tr>
```

```
<tr>

<td>Name:</td>

<td>{{user.restaurant.rname}}</td>

</tr>


<tr>

<td>info:</td>

<td>{{user.restaurant.info}}</td>

</tr>

<tr>

<td>Location:</td>

<td>{{user.restaurant.location}}</td>

</tr>

<tr>

<td>Status:</td>

<td>{{user.restaurant.status}}</td>

</tr>

<tr>

<td>Approved:</td>

<td>{{user.restaurant.approved}}</td>

</tr>

<tr>

<td>Email:</td>

<td><a href="#">{{user.email}}</a></td>

</tr>
```

```
</tbody>

</table>

<div class="container">

<div class="row">

<div class="col">

<a class="btn btn-success btn-lg" href="{% url 'rupdate' user.id %}">Edit</a>

</div>

<!--<div class="col">

<a class="btn btn-success btn-lg" href="{%url 'rcreate' %}">Complete profile</a>

</div>-->

</div>

</div>

</div>

</div>

</div>

</div>

</div>



{% endblock%}
```

Business logic

```
from django.contrib.auth.forms import UserCreationForm

from django import forms
from .models import User,Customer,Restaurant,Item,Menu

class CustomerSignUpForm(forms.ModelForm):
password = forms.CharField(widget=forms.PasswordInput)
class Meta:
model = User
fields=['username','email','password']
def save(self, commit=True):
user = super().save(commit=False)
user.is_customer=True
if commit:
user.save()
return user


class RestuarantSignUpForm(forms.ModelForm):
password = forms.CharField(widget=forms.PasswordInput)
class Meta:
model =User
fields=['username','email','password']
def save(self,commit=True):
user=super().save(commit=False)
user.is_restaurant=True
if commit:
user.save()
return user
```

```
class CustomerForm(forms.ModelForm):

class Meta:

model = Customer

fields =['f_name','l_name','city','phone','address']



class RestuarantForm(forms.ModelForm):

class Meta:

model = Restaurant

fields =['rname','info','location','r_logo','min_ord','status','approved']



from django.db import models

from django.contrib.auth.models import AbstractUser

from django.conf import settings

from django.db.models.signals import post_save


class User(AbstractUser):

        is_customer   = models.BooleanField(default=False)

        is_restaurant = models.BooleanField(default=False)



class Customer(models.Model):

        user                =
models.OneToOneField(settings.AUTH_USER_MODEL,on_delete=models.CASCADE)

        f_name              = models.CharField(max_length=20,blank=False)
```

```
    l_name          = models.CharField(max_length=20,blank=False)
    city        = models.CharField(max_length=40,blank=False)
    phone       = models.CharField(max_length=10,blank=False)
    address         = models.TextField()

    def __str__(self):
        return self.user.username

class Restaurant(models.Model):
    user    =
models.OneToOneField(settings.AUTH_USER_MODEL,on_delete=models.CASCADE)
    rname       = models.CharField(max_length=100,blank=False)
    info        = models.CharField(max_length=40,blank=False)
    min_ord         = models.CharField(max_length=5,blank=False)
    location    = models.CharField(max_length=40,blank=False)
    r_logo      = models.FileField(blank=False)

    REST_STATE_OPEN    = "Open"
    REST_STATE_CLOSE   = "Closed"
    REST_STATE_CHOICES =(
            (REST_STATE_OPEN,REST_STATE_OPEN),
            (REST_STATE_CLOSE,REST_STATE_CLOSE)
        )
    status  =
models.CharField(max_length=50,choices=REST_STATE_CHOICES,default=REST_STATE_O
PEN,blank=False)
    approved = models.BooleanField(blank=False,default=True)
```

```
        def __str__(self):
                return self.rname


class Item(models.Model):
        id                              = models.AutoField(primary_key=True)
        fname             = models.CharField(max_length=30,blank=False)
        category          = models.CharField(max_length=50,blank=False)

        def __str__(self):
                return self.fname


class Menu(models.Model):
        id                    = models.AutoField(primary_key=True)
        item_id  = models.ForeignKey(Item,on_delete=models.CASCADE)
        r_id     = models.ForeignKey(Restaurant,on_delete=models.CASCADE)
        price    = models.IntegerField(blank=False)
        quantity = models.IntegerField(blank=False,default=0)

        def __str__(self):
                return self.item_id.fname+' - '+str(self.price)



class Order(models.Model):
        id                                          = models.AutoField(primary_key=True)
        total_amount    = models.IntegerField(default=0)
        timestamp       = models.DateTimeField(auto_now_add=True)
```

```
    delivery_addr   = models.CharField(max_length=50,blank=True)
    orderedBy       = models.ForeignKey(User ,on_delete=models.CASCADE)
    r_id                        = models.ForeignKey(Restaurant ,on_delete=models.CASCADE)

    ORDER_STATE_WAITING    = "Waiting"
    ORDER_STATE_PLACED               = "Placed"
    ORDER_STATE_ACKNOWLEDGED = "Acknowledged"
    ORDER_STATE_COMPLETED    = "Completed"
    ORDER_STATE_CANCELLED    = "Cancelled"
    ORDER_STATE_DISPATCHED   = "Dispatched"

    ORDER_STATE_CHOICES = (
        (ORDER_STATE_WAITING,ORDER_STATE_WAITING),
     (ORDER_STATE_PLACED, ORDER_STATE_PLACED),
     (ORDER_STATE_ACKNOWLEDGED, ORDER_STATE_ACKNOWLEDGED),
     (ORDER_STATE_COMPLETED, ORDER_STATE_COMPLETED),
     (ORDER_STATE_CANCELLED, ORDER_STATE_CANCELLED),
     (ORDER_STATE_DISPATCHED, ORDER_STATE_DISPATCHED)
    )
    status =
models.CharField(max_length=50,choices=ORDER_STATE_CHOICES,default=ORDER_STAT
E_WAITING)


    def __str__(self):
        return str(self.id) +' '+self.status
```

```
class orderItem(models.Model):
    id                          = models.AutoField(primary_key=True)
    item_id         = models.ForeignKey(Menu ,on_delete=models.CASCADE)
    ord_id          = models.ForeignKey(Order,on_delete=models.CASCADE)
    quantity        = models.IntegerField(default=0)

    def __str__(self):
        return str(self.id)



from django.shortcuts import render, redirect,get_object_or_404
from django.contrib.auth import authenticate, login,logout
from .forms import
CustomerSignUpForm,RestuarantSignUpForm,CustomerForm,RestuarantForm
from django.contrib.auth.decorators import login_required
from collections import Counter
from django.urls import reverse
from django.db.models import Q
from .models import Customer,Restaurant,Item,Menu,Order,orderItem,User



#### ---------- General Side -------------------#####


# Showing index page
def index(request):
    return render(request,'webapp/index.html',{})
```

```
def orderplaced(request):
        return render(request,'webapp/orderplaced.html',{})


# Showing Restaurants list to Customer
def restuarent(request):
        r_object = Restaurant.objects.all()
        query   = request.GET.get('q')
        if query:
                r_object=Restaurant.objects.filter(Q(rname__icontains=query)).distinct()
                return render(request,'webapp/restaurents.html',{'r_object':r_object})
        return render(request,'webapp/restaurents.html',{'r_object':r_object})



# logout
def Logout(request):
        if request.user.is_restaurant:
                logout(request)
                return redirect("rlogin")
        else:
                logout(request)
                return redirect("login")




#### -----------------Customer Side--------------------- ######
```

```
# Creating Customer Account
def customerRegister(request):
        form =CustomerSignUpForm(request.POST or None)
        if form.is_valid():
                user      = form.save(commit=False)
                username  =   form.cleaned_data['username']
                password  = form.cleaned_data['password']
                user.is_customer=True
                user.set_password(password)
                user.save()
                user = authenticate(username=username,password=password)
                if user is not None:
                        if user.is_active:
                                login(request,user)
                                return redirect("ccreate")
        context ={
                'form':form
        }
        return render(request,'webapp/signup.html',context)



# Customer Login
def customerLogin(request):
        if request.method=="POST":
                username = request.POST['username']
                password = request.POST['password']
```

```
            user    = authenticate(username=username,password=password)
            if user is not None:
                if user.is_active:
                    login(request,user)
                    return redirect("profile")
                else:
                    return render(request,'webapp/login.html',{'error_message':'Your
account disable'})
            else:
                return render(request,'webapp/login.html',{'error_message': 'Invalid
Login'})
    return render(request,'webapp/login.html')



# customer profile view
def customerProfile(request,pk=None):
    if pk:
        user = User.objects.get(pk=pk)
    else:
        user=request.user

    return render(request,'webapp/profile.html',{'user':user})



#Create customer profile
def createCustomer(request):
    form = CustomerForm(request.POST or None)
```

```
        if form.is_valid():

                instance = form.save(commit=False)

                instance.user = request.user

                instance.save()

                return redirect("profile")

        context={

        'form':form,

        'title':"Complete Your profile"

        }

        return render(request,'webapp/profile_form.html',context)



#  Update customer detail

def updateCustomer(request,id):

        form     = CustomerForm(request.POST or None,instance=request.user.customer)

        if form.is_valid():

                form.save()

                return redirect('profile')

        context={

        'form':form,

        'title':"Update Your profile"

        }

        return render(request,'webapp/profile_form.html',context)


def restuarantMenu(request,pk=None):
```

```
menu = Menu.objects.filter(r_id=pk)
rest = Restaurant.objects.filter(id=pk)

items =[]
for i in menu:
    item = Item.objects.filter(fname=i.item_id)
    for content in item:
        temp=[]
        temp.append(content.fname)
        temp.append(content.category)
        temp.append(i.price)
        temp.append(i.id)
        temp.append(rest[0].status)
        temp.append(i.quantity)
        items.append(temp)
context = {
    'items'  : items,
    'rid'    : pk,
    'rname': rest[0].rname,
    'rmin'   : rest[0].min_ord,
    'rinfo' : rest[0].info,
    'rlocation':rest[0].location,
}
return render(request,'webapp/menu.html',context)
```

```
@login_required(login_url='/login/user/')
def checkout(request):
    if request.POST:
        addr  = request.POST['address']
        ordid = request.POST['oid']
        Order.objects.filter(id=int(ordid)).update(delivery_addr = addr,
                                status=Order.ORDER_STATE_PLACED)
        return redirect('/orderplaced/')
    else:
        cart = request.COOKIES['cart'].split(",")
        cart = dict(Counter(cart))
        items = []
        totalprice = 0
        uid = User.objects.filter(username=request.user)
        oid = Order()
        oid.orderedBy = uid[0]
        for x,y in cart.items():
            item = []
            it = Menu.objects.filter(id=int(x))
            if len(it):
                oiid=orderItem()
                oiid.item_id=it[0]
                oiid.quantity=int(y)
                oid.r_id=it[0].r_id
                oid.save()
                oiid.ord_id =oid
```

```
                        oiid.save()
                        totalprice += int(y)*it[0].price
                        item.append(it[0].item_id.fname)
                        it[0].quantity = it[0].quantity - y
                        it[0].save()
                        item.append(y)
                        item.append(it[0].price*int(y))

                    items.append(item)
                oid.total_amount=totalprice
                oid.save()
                context={
                    "items":items,
                    "totalprice":totalprice,
                    "oid":oid.id
                }
                return render(request,'webapp/order.html',context)




####### ------------------- Restaurant Side ------------------- #####

# creating restuarant account
def restRegister(request):
    form =RestuarantSignUpForm(request.POST or None)
```

```
        if form.is_valid():
            user      = form.save(commit=False)
            username  =   form.cleaned_data['username']
            password  = form.cleaned_data['password']
            user.is_restaurant=True
            user.set_password(password)
            user.save()
            user = authenticate(username=username,password=password)
            if user is not None:
                if user.is_active:
                    login(request,user)
                    return redirect("rcreate")
        context ={
            'form':form
        }
        return render(request,'webapp/restsignup.html',context)



# restuarant login
def restLogin(request):
        if request.method=="POST":
            username = request.POST['username']
            password = request.POST['password']
            user     = authenticate(username=username,password=password)
            if user is not None:
                if user.is_active:
                    login(request,user)

                    return redirect("rprofile")

                else:

                    return render(request,'webapp/restlogin.html',
{'error_message':'Your account disable'})

            else:

                return render(request,'webapp/restlogin.html',{'error_message': 'Invalid
Login'})

    return render(request,'webapp/restlogin.html')




# restaurant profile view

def restaurantProfile(request,pk=None):

    if pk:

        user = User.objects.get(pk=pk)

    else:

        user=request.user


    return render(request,'webapp/rest_profile.html',{'user':user})


# create restaurant detail

@login_required(login_url='/login/restaurant/')

def createRestaurant(request):

    form=RestuarantForm(request.POST or None,request.FILES or None)

    if form.is_valid():

        instance = form.save(commit=False)

        instance.user = request.user
```

```
                instance.save()

                return redirect("rprofile")

        context={

        'form':form,

        'title':"Complete Your Restaurant profile"

        }

        return render(request,'webapp/rest_profile_form.html',context)


#Update restaurant detail

@login_required(login_url='/login/restaurant/')

def updateRestaurant(request,id):

        form     = RestuarantForm(request.POST or None,request.FILES or 
None,instance=request.user.restaurant)

        if form.is_valid():

                form.save()

                return redirect('rprofile')

        context={

        'form':form,

        'title':"Update Your Restaurant profile"

        }

        return render(request,'webapp/rest_profile_form.html',context)



# add  menu item for restaurant

@login_required(login_url='/login/restaurant/')

def menuManipulation(request):

        if not request.user.is_authenticated:
```

```
        return redirect("rlogin")

rest=Restaurant.objects.filter(id=request.user.restaurant.id);
rest=rest[0]
if request.POST:
        type=request.POST['submit']
        if type =="Modify":
                menuid = int(request.POST['menuid'])
                memu= Menu.objects.filter(id=menuid).\

update(price=int(request.POST['price']),quantity=int(request.POST['quantity']))
        elif type == "Add" :
                itemid=int(request.POST['item'])
                item=Item.objects.filter(id=itemid)
                item=item[0]
                menu=Menu()
                menu.item_id=item
                menu.r_id=rest
                menu.price=int(request.POST['price'])
                menu.quantity=int(request.POST['quantity'])
                menu.save()
        else:
                menuid = int(request.POST['menuid'])
                menu = Menu.objects.filter(id=menuid)
                menu[0].delete()

menuitems=Menu.objects.filter(r_id=rest)
```

```
        menu=[]
        for x in menuitems:
                cmenu=[]
                cmenu.append(x.item_id)
                cmenu.append(x.price)
                cmenu.append(x.quantity)
                cmenu.append(x.id)
                menu.append(cmenu)

        menuitems = Item.objects.all()
        items = []
        for y in menuitems:
                citem = []
                citem.append(y.id)
                citem.append(y.fname)
                items.append(citem)

        context={
                "menu":menu,
                "items":items,
                "username":request.user.username,
        }
        return render(request,'webapp/menu_modify.html',context)

def orderlist(request):
        if request.POST:
```

```
        oid = request.POST['orderid']
        select = request.POST['orderstatus']
        select = int(select)
        order = Order.objects.filter(id=oid)
        if len(order):
            x = Order.ORDER_STATE_WAITING
            if select == 1:
                x = Order.ORDER_STATE_PLACED
            elif select == 2:
                x = Order.ORDER_STATE_ACKNOWLEDGED
            elif select == 3:
                x = Order.ORDER_STATE_COMPLETED
            elif select == 4:
                x = Order.ORDER_STATE_DISPATCHED
            elif select == 5:
                x = Order.ORDER_STATE_CANCELLED
            else:
                x = Order.ORDER_STATE_WAITING
            order[0].status = x
            order[0].save()

    orders = Order.objects.filter(r_id=request.user.restaurant.id).order_by('-timestamp')
    corders = []

    for order in orders:
```

```
user = User.objects.filter(id=order.orderedBy.id)

user = user[0]

corder = []

if user.is_restaurant:

        corder.append(user.restaurant.rname)

        corder.append(user.restaurant.info)

else:

        corder.append(user.customer.f_name)

        corder.append(user.customer.phone)

items_list = orderItem.objects.filter(ord_id=order)


items = []

for item in items_list:

        citem = []

        citem.append(item.item_id)

        citem.append(item.quantity)

        menu = Menu.objects.filter(id=item.item_id.id)

        citem.append(menu[0].price*item.quantity)

        menu = 0

        items.append(citem)


corder.append(items)

corder.append(order.total_amount)

corder.append(order.id)


x = order.status
```

```
            if x == Order.ORDER_STATE_WAITING:
                continue
            elif x == Order.ORDER_STATE_PLACED:
                x = 1
            elif x == Order.ORDER_STATE_ACKNOWLEDGED:
                    x = 2
            elif x == Order.ORDER_STATE_COMPLETED:
                    x = 3
            elif x == Order.ORDER_STATE_DISPATCHED:
                    x = 4
            elif x == Order.ORDER_STATE_CANCELLED:
                    x = 5
            else:
                    continue


            corder.append(x)
            corder.append(order.delivery_addr)
            corders.append(corder)


        context = {"orders" : corders,}
        return render(request,"webapp/order-list.html",context)
```

<u>Conclusion of the Project Online Food Ordering System:</u>

Our project is only a humble venture to satisfy the needs to manage their project work. Several user friendly coding have also adopted. This package shall prove to be a powerful package in satisfying all the requirements of the school. The objective of software planning is to provide a frame work that enables the manger to make reasonable estimates made within a limited time frame at the beginning of the software project and should be updated regularly as the project progresses.

At the end it is concluded that we have made effort on following points…

- A description of the background and context of the project and its relation to work already done in the area.
- Made statement of the aims and objectives of the project.
- The description of Purpose, Scope, and applicability.
- We define the problem on which we are working in the project.
- We describe the requirement Specifications of the system and the actions that can be done on these things.
- We understand the problem domain and produce a model of the system, which describes operations that can be performed on the system.
- We included features and operations in detail, including screen layouts.
- We designed user interface and security issues related to system.
- Finally the system is implemented and tested according to test cases.

Future Scope of the Project:

In a nutshell, it can be summarized that the future scope of the project circles around maintaining information regarding:

- We can add printer in future.
- We can give more advance software for Online Food Ordering System including more facilities
- We will host the platform on online servers to make it accessible worldwide
- Integrate multiple load balancers to distribute the loads of the system
- Create the master and slave database structure to reduce the overload of the database queries
- Implement the backup mechanism for taking backup of codebase and database on regular basis on different servers

The above-mentioned points are the enhancements which can be done to increase the applicability and usage of this project. Here we can maintain the records of Food Item and Category. Also, as it can be seen that now-a-days the players are versatile, i.e. so there is a scope for introducing a method to maintain the Online Food Ordering System. Enhancements can be done to maintain all the Food Item, Category, Customer, Order, Confirm Order.

We have left all the options open so that if there is any other future requirement in the system by the user for the enhancement of the system then it is possible to implement them. In the last we would like to thanks all the persons involved in the development of the system directly or indirectly. We hope that the project will serve its purpose for which it is develop there by underlining success of process.

Limitation of Project on Online Food Ordering System

Although I have put my best efforts to make the software flexible, easy to operate but limitations cannot be ruled out even by me. Though the software presents a broad range of options to its users some intricate options could not be covered into it; partly because of logistic and partly due to lack of sophistication. Paucity of time was also major constraint, thus it was not possible to make the software foolproof and dynamic. Lack of time also compelled me to ignore some part such as storing old result of the candidate etc.

Considerable efforts have made the software easy to operate even for the people not related to the field of computers but it is acknowledged that a layman may find it a bit problematic at the first instance. The user is provided help at each step for his convenience in working with the software.

List of limitations which is available in the Online Food Ordering System:

- Excel export has not been developed for Food Item, Category due to some criticality.
- The transactions are executed in off-line mode, hence on-line data for Customer, Order capture and modification is not possible.
- Off-line reports of Food Item, Confirm Order, Customer cannot be generated due to batch mode execution.

References and Bibliography:

- Google for problem solving
- https://docs.python.org/3/reference/
- https://www.sqlite.org/index.html
- https://www.tutorialspoint.com/django/index.html
- https://www.w3schools.com/python/python_classes.asp/
- https://www.w3schools.com/html/
- https://www.w3schools.com/tags/tag_object.asp
- https://www.w3schools.com/w3css/default.asp
- https://www.python.org/about/gettingstarted/
- https://www.tutorialspoint.com/python/index.htm