ONLINE BAKERY SHOPPING SITE

Under Guidance Of

Mr. Joyjit Guha Biswas

Subject Matter Expert(Python)

Ardent Computech Pvt Ltd(An ISO 9001:2008 Certified)
Module 132 SDF building sector 5 Kolkata-700091

A Project Report
Submitted In Partial Fulfillment Of The Requirements
For The Award Of the
Bachelor of Technology
Project Carried Out At

Ardent Computech Pvt Ltd(An ISO 9001:2008 Certified)
Module 132 SDF building sector 5 Kolkata-700091
Submitted By:-

Sumon Sit
Shamik Banerjee
Triparna Chakraborty

Department Of Bachelor of Computer Application

Institute of Engineering and Management
Ashram Building, GN-34/2, Sector – V, Saltlake Electronics Complex Kolkata – 700 091, West Bengal, India.

(Note: All entries of the proforma of approval should be filled up with appropriate and complete information of approval in any respect will be summarily rejected.)

Name of the StudentWithGroup:
1: Sumon sit
2. Shamik Banerjee
3. Triparna Chakraborty

1. Title oftheProject : ONLINE BAKERY SHOPPING SITE
2. Name and Address oftheGuide : **Mr. Joyjit GuhaBiswas**
   Sr. Subject Matter Expert & Technical Head(Python)
   Ardent Computech Pvt Ltd(An ISO 9001:2008 Certified)
   Module 132 SDF building sector 5 Kolkata-700091
3. Educational Qualification of theGuide: Ph.d* M.tech* B.E*/B.Tech * MCA* M.Sc*

   ☐ ☐ ☐ ☐ ☐
4. Working and Teachingexperience of theGuide**:** **…….Years**
5. Software used in theProject:
   a. Sublime Text
   b. Xampserver
   c. My Sql Server

Signature oftheStudent
Date:
For OfficeUseOnly

Approved **Not Approved**

Signature of theGuide
Date:
Name: Mr. Joyjit Guha Biswas
Subject MatterExpert.
Signature, Designation, Stamp of the Project Proposal Evaluator

Project Responsibility Form

ONLINE BAKERY SHOPPING SITE

SERIAL NUMBER	NAME OF MEMBER	RESPONSIBILITY
1	Shamik Banerjee	CODING (Back End)
2	Sumon Sit	Designing (Front End)
3	Triparna Chakraborty	Documentation and Elaboration

Self Certificate

This is to certify that the dissertation/project proposal entitled **"ONLINE BAKERY SHOPPING SITE"** is done by us, is an Authentic work carried out for the partial fulfillment of the requirements for the award of the certificate of **Bachelor of Technology** under the guidance of **Mr. Joyjit Guha Biswas**. The matter embodied in this project work has not been submitted earlier for award of any certificate to the best of our knowledge and belief.

Name of the Students:-

Sumon Sit
Shamik Banerjee
Triparna Chakraborty

Signature of the students
a.
b.
c.

Certificate by Guide

This is to certify that this project entitled **"Online Study Management System** "submitted in partial fulfillment of the certificate of Bachelor of Computer Application through **Ardent Computech Pvt Ltd**, done by the
Group Members-:

Sumon Sit

Shamik Banerjee

Triparna Chakraborty

is an authentic work carried out under my guidance & best of our knowledge and belief..
a.

b.

c.

Signature ofthestudents

Date:

Signature of theGuide

Date:

Certificate of Approval

This is to certify that this proposal of Minor project, entitled **"ONLINE BAKERY SHOPPING SITE"**is a record of bona-fide work, carried out by: 1. Sumon Sit , 2. Shamik Banerjee, 3.Triparna Chakraborty under my supervision and guidance through the Ardent Computech Pvt Ltd. In my opinion, the report in its present form is in partial fulfillment of all the requirements, as specified by the Kanad Institute of Engineering and Management as per regulations of the ***Ardent®*** . In fact, it has attained the standard, necessary for submission. To the best of my knowledge, the results embodied in this report, are original in nature and worthy of incorporation in the present version of the report for Bachelor of Technology.

Guide/Supervisor

Mr. Joyjit Guha Biswas
Subject Matter Expert & Technical Head (Python)
Ardent Computech Pvt Ltd (An ISO 9001:2008 Certified)
Module 132 SDF building sector 5 Kolkata-700091

ExternalExaminer(s)

Head of theDepartment
Department of Computer Science

Kolkata , W.B (Affiliated to WBUT,WB)

online Bakery Shopping Site

TABLE OF CONTENTS

1. ARDENT COMPUTECHPVT.LTD.

Ardent Computech Private Limited is an ISO 9001-2008 certified Software Development Company in India. It has been operating independently since 2003. It was recently merged with ARDENT TECHNOLOGIES.

Ardent Technologies

ARDENT TECHNOLOGIES is a Company successfully providing its services currently in UK, USA, Canada and India. The core line of activity at ARDENT TECHNOLOGIES is to develop customized application software covering the entire responsibility of performing the initial system study, design, development, implementation and training. It also deals with consultancy services and Electronic Security systems. Its primary clientele includes educational institutes, entertainment industries, resorts, theme parks, service industry, telecom operators, media and other business houses working in variouscapacities.

Ardent Collaborations

ARDENT COLLABORATIONS, the Research Training and Development Department of ARDENT COMPUTECH PVT LTD is a professional training Company offering IT enabled services & industrial trainings for B-Tech, MCA, BCA, MSc and MBA fresher's and experienced developers/programmers in various platforms. Summer Training / Winter Training / Industrial training will be provided for the students of B.TECH, M.TECH, MBA and MCA only. Deserving candidates may be awarded stipends, scholarships and other benefits, depending on their performance and recommendations of thementors.

Associations

Ardent is an ISO 9001:2008 company.

It is affiliated to National Council of Vocational Training (NCVT), Directorate General of Employment & Training (DGET), Ministry of Labor & Employment, and Government of India.

ONLINE BAKERY SHOPPING SITE

2. INTRODUCTION

An online Bakery shop that allows users to check for various bakery products available at the online store and purchase online. The project consists of list of bakery products also displayed their various categories. For enter the website you have to create an account in this website .Once user wishes to checkout he must register on the site first. He can then login using same id password next time.

Now he may pay through cash on delivery. Once the user makes a successful order he gets a copy of the shopping receipt on his email id or what's app. Here we use user friendly interface to make the entire frontend.

The middle tier or code behind model is designed for fast processing. And SQL serves as a backend to store bakery products lists data. Thus, the online Bakery shopping project brings an entire bakery shop online and makes it easy for both buyer and seller.

2a. OBJECTIVE

The objective of ONLINE BAKERY SHOPPING is to allow the people to celebrate there happiness with a beautiful cake without going outside of the house or any reason they have not enough time to go outside and order a cake using this website they can easily order cakes .Many time people can’t get their favourite cake because the problem of bring the cake to the vanue by ordering they don’t have to think about it .

Using this website you don’t have to give money online we have the facility of cash on delivery after checking the condition of the cake.

This website also acts as an aggregator for this purpose.

Throughout the project the focus has been on presenting information in an easy,wittily and intelligent manner.

The website provides facilities like online registration and profile creation for farther if you want to order again you have to login again for security purpose in this website .

2b. SCOPE

We can also add some pages who wants to make customized cake. You can add any flavour any fruit or any picture in the cake as per your requirement.we also try to add online transaction in the system for caseless and digital transaction.

And add search bar to search what type of cake you want to order . According to our website, role of admin is to insert and modify the details of cakes, manage the order and delivary process and the role of administrator is to insert and modify the details of the cakes every month or whenever an update is required.

SYSTEM ANALYSIS

3a. IDENTIFICATION OF NEED

System analysis is a process of gathering and interpreting facts, diagnosing problems and the information 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 studies the minutest detail and gets analyzed. The system analysist 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 organization are traced to the various processes. System analysis is concerned with becoming aware of the problem ,identifying the relevant and Decisional variables ,analysis and synthesizing the various factors and determining an optimal or at least a satisfactory solution or program ofaction.

A detailed study of the process must be made by various techniques like interviews, questionnaires etc. The data collected by these sources must be 9scrutinized 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 area are identified .The designer now function as a problem solver and tries to sort out the difficulties that the enterprise faces. The solution 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 requestand

suitable changes are made. This is loop that ends as soon as the user is satisfied with proposal.

3b. FEASIBILITY STUDY

Feasibility study is made to see if the project on completion will serve the purpose the organization for the amount of work.

Effort and the time that spend on it. Feasibility study lets the developer foresee the future of the project and the usefulness. A Feasibility study of a system proposal is according to its workability, which is the impact on the organization, ability to meet their user needs and effective use of resources .Thus when a new application is proposed it normally goes through a feasibility study before it is approved fordevelopment.

The document provide the feasibility of the project that is being designed and lists various area that were considered very carefully during the feasibility study of this project such as Technical , Economic and operationalfeasibilities.

3c. WORK FLOW

This Document plays a vital role in the development life cycle (SDLC) as it describes the complete requirement of the system. It is meant for use by the developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approvalprocess.

The Waterfall Model was first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

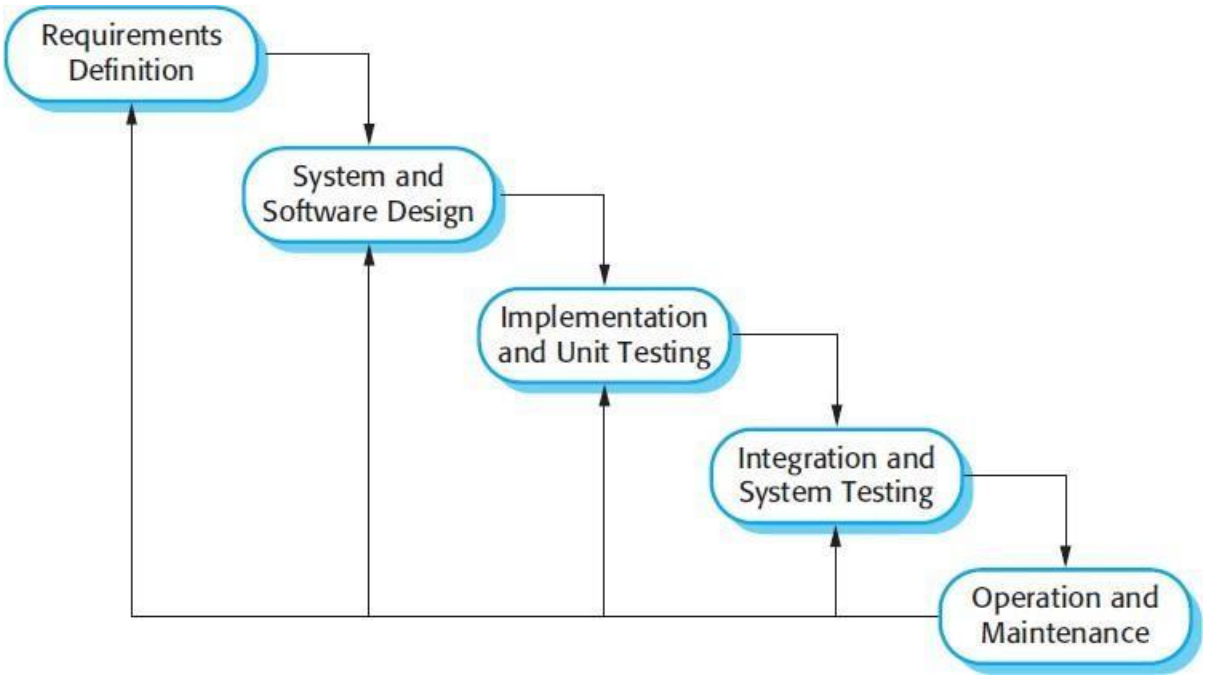
Waterfall model is the earliest SDLC approach that was used for software development .

The waterfall Model illustrates the software development process in a linear sequential flow; hence it is also referred to as a linear-sequential life cycle model. This means that any phase in the development process begins only if the previous phase is complete. In waterfall model phases do not overlap.

Waterfall Model design

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In Waterfall model, typically, the outcome of one phase acts as the input for the next phasesequentially.

Following is a diagrammatic representation of different phases of waterfall model.



The sequential phases in Waterfall model are:

- **Requirement Gathering and analysis:** All possible requirements of the system to be developed are captured in this phase and documented in a requirement specificationdoc.
- **System Design:** The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall systemarchitecture.
- **Implementation:** With inputs from system design, the system is first developed in small programs called units, whichare integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as UnitTesting.
- **Integration and Testing:** All the units developed in the implementationphaseareintegratedintoasystemaftertestingof

each unit. Post integration the entire system is tested for any faults and failures.

- **Deployment of system:** Once the functional and non functional testing is done, the product is deployed in the customer environment or released into the market.
- **Maintenance:** There are some issues which come up in the client environment. To fix those issues patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model phases do not overlap.

Waterfall Model Application

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are:

- Requirements are very well documented, clear andfixed.
- Product definition isstable.
- Technology is understood and is notdynamic.
- There are no ambiguousrequirements.
- Ample resources with required expertise are available to support theproduct.
- The project isshort.

The advantage of waterfall development is that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.

Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order.

3d. STUDY OF THE SYSTEM

Modules:

The modules used in this software are as follows:

- LOGIN: This module is for ADMIN/COUNCILLORandUSER. ADMIN/COUNCILLORhas the authority to Insert, Update and Delete, branch, courses.
  Whereas, USER can simply avail all the information on all courses hassle free.
- HOME :This page contains an overview of highlights forother pages.
- ABOUT-US:This page contains the information aboutSMS.
- COURSES: This page contains the updated information aboutthe courses that are available.
- FREE COUNSELLING: This page conations the contact info of the organization for any of the query and also gives free counseling for the students .
- TEST YOURSELF: This page contains thhe questions required for the students to check their capability and test their self for the given courses.
- SEMESTER: Different coding languages have been classified and placed according to the required specification under the specified semester as per the university norms.
- REGISTRATION: This page asks for the information that is required to fulfill the website criteria to do the courses.
- PROGRAMMING LANGUAGES: This page have the languages accordingly for the user to directly open and learn.

3e. INPUT AND OUTPUT

The main inputs ,outputs and the major function the details are :
INPUT

- Councillor can login using op-id andpassword.
- Admin can login using admin-id andpassword.
- Admin insert and modify the details councillor, branch,course, installment, and paymentdetails.
- Councillor can make admission of a student by registeringstudent details.
- Councillor can modify student details searching them by idand name.

OUTPUT

- can view the details, course, as well as test themselves with a separate coding test provided.
- Admin can view the details of admin, course, installment,student and payment details.

3f. SOFTWARE REQUIREMENT SPECIFICATIONS

Software Requirements Specification provides an overview of the entire project. It is a description of a software system to be developed, laying out functional and nonfunctional requirements. The software requirements specification document enlists enough and necessary requirements that are required for the project development. To derive the requirements we need to have clear and thorough understanding of the project to be developed. This is prepared after the detailed communication with project team and the customer.

The developer is responsiblefor:-

- ✓ Developing the system, which meets the SRS and solving all the requirements of thesystem?
- ✓ Demonstrating the system and installing the system at client's location after acceptance testing issuccessful.
- ✓ Submitting the required user manual describing the system interfaces to work on it and also the documents of thesystem.
- ✓ Conducting any user training that might be needed for using the system.
- ✓ Maintain the system for a period of one year afterinstallation.

HARDWARE REQUIREMENTS:

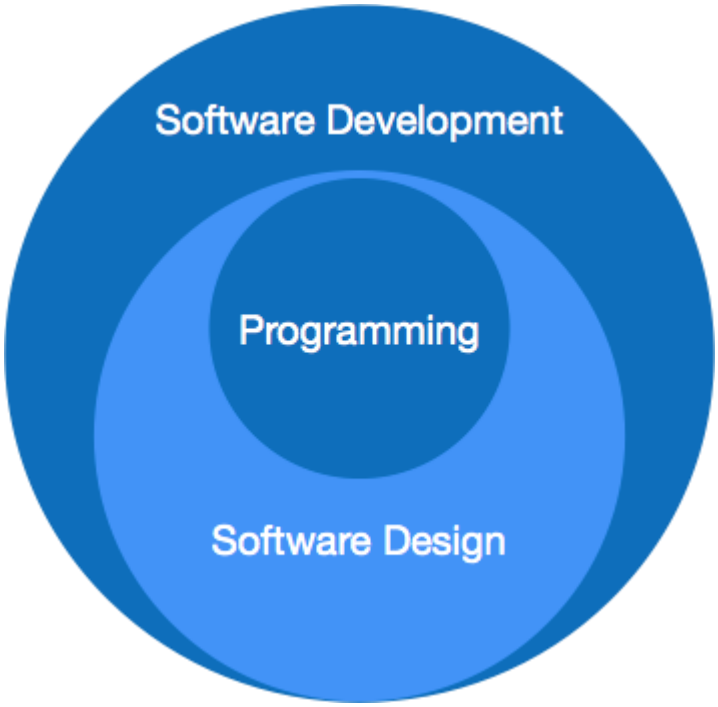
- Computer that has a 1.6GHz or fasterprocessor
- 1 GB (32 Bit) or 2 GB (64 Bit) RAM (Add 512 MB if running in a virtualmachine)
- HDD 20 GB Hard Disk Space and Above Hardware Requirements
- 5400 RPM hard diskdrive
- DVD-ROMDrive

SOFTWARE REQUIREMENTS:

- WINDOWS OS (XP/2000/2003 or 2000 Server/Vista or7)
- DB Browser

3g. SOFTWARE ENGINEERING PARADIGM APPLIED

Software paradigms refer to the methods and steps, which are taken while designing the software. There are many methods proposed and are in work today, but we need to see where in the software engineering these paradigms stand. These can be combined into various categories, though each of them is contained in one another.



Programming paradigm is a subset of Software design paradigm which is further a subset of Software developmentparadigm.

There are two levels of reliability. The first is meeting the right requirement. A carefully and through systems study is needed to satisfy this aspect of reliability. The second level of systems reliability involves the actual working delivered to the user. At this level, the systems

reliability is interwoven with software engineering and development. There are three approaches to reliability.

1. Error avoidance: Prevents errors from occurring in software.

2. Error detection and correction: In this approach errors are recognized whenever they are encountered and correcting the error by effect of error of the system does notfail.

3. Error tolerance: In this approach errors are recognized whenever they occur, but enables the system to keep running through degraded perform or Appling values that instruct the system to continue process.

Maintenance:

The key to reducing need for maintenance, while working, if possible to do essential tasks.

1. More accurately defining user requirement during system development.

2. Assembling better systems documents.

3. Using some effective methods for designing, processing, and login and communicating information with project team members.

4. Making better use of existing tools and techniques.

SYSTEM DESIGN

4a. DATA FLOW DIAGRAM

A **data flow diagram** (**DFD**) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A **DFD** is often used as a preliminary step to create an overview of the system, which can later beelaborated.

DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of process or information about whether processes will operate in sequence or in parallel (which is shown on a flowchart).

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 whi1ch 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. It also identifies internal data stores that must be present in order for the system to do its job, and shows the flow of data between the various parts of thesystem.

Data flow diagrams are one of the three essential perspectives of the structured-systems analysis and design method SSADM. The sponsor of a project and the end users will need to be briefed and consulted throughout all stages of a system's evolution. With a data flow diagram, users are able to visualize how the system will operate, what the system will accomplish, and how the system will be implemented. The old system's dataflow diagrams can be drawn up and compared with the new system's data flow diagrams to draw comparisons to implement a more efficient system. Data flow diagrams can be used to provide the end user with a physical idea of where the data they input ultimately has an effect upon the structure of the whole system from order to dispatch to report.

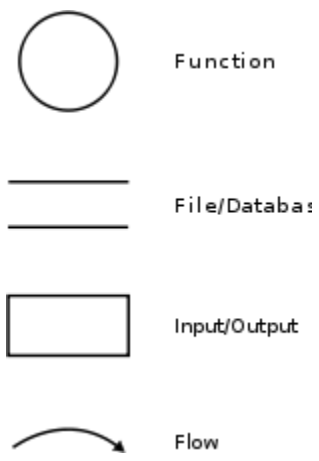
How any system is developed can be determined through a data flow diagram model.

In the course of developing a set of *leveled* data flow diagrams the analyst/designer is forced to address how the system may be decomposed into component sub-systems, and to identify the transactiondatain the datamodel.

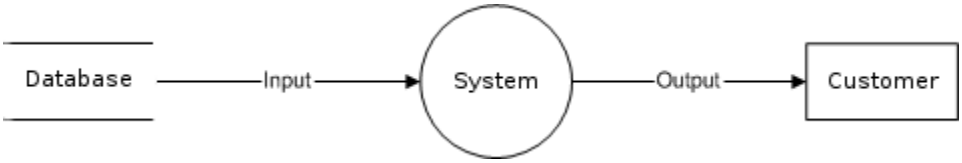
Data flow diagrams can be used in both Analysis and Design phase of theSDLC.

There are different notations to draw data flow diagrams. defining different visual representations for processes, data stores, data flow, and external entities.[6]

DFD NOTATION



DFD EXAMPLE



Steps to Construct Data Flow Diagram:-

Four Steps are generally used to construct a DFD.

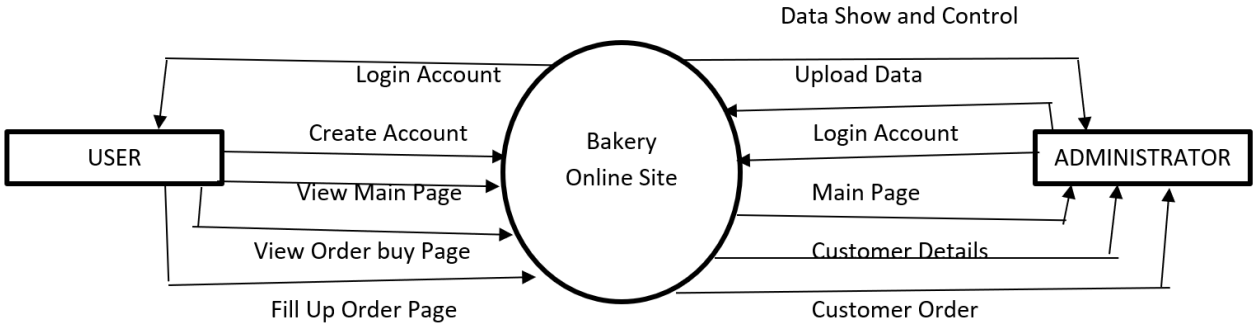
- Process should be named and referred for easy reference. Each name should be representative of thereference.
- The destination of flow is from top to bottom and from left toright.
- When a process is distributed into lower level details they are numbered.
- The names of data stores, sources and destinations are written in capitalletters.

Rules for constructing a Data Flow Diagram:-

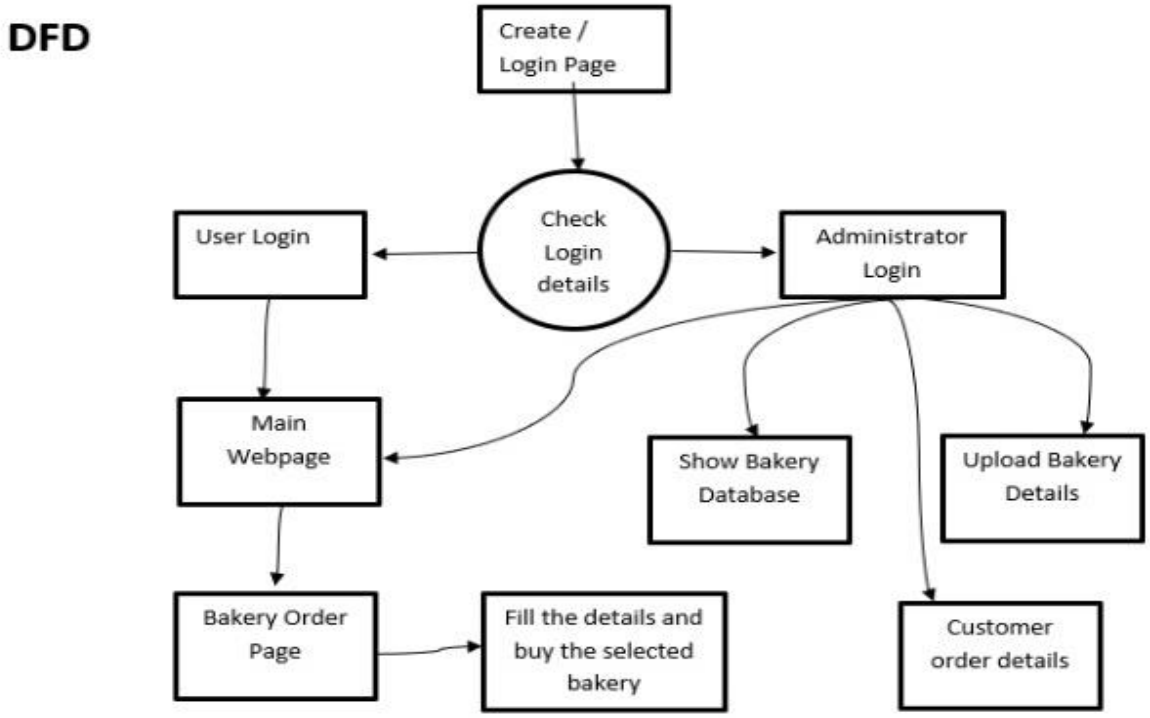
- Arrows should not cross eachother.
- Squares, Circles, Files must bear aname.
- Decomposed data flow squares and circles can have samenames.
- Draw all data flow around the outside of thediagram.

DATA FLOW DIAGRAM

LEVEL-0 DFD DIAGRAM



LEVEL-1 DFD DIAGRAM



4b. SEQUENCE DIAGRAM

A **Sequence diagram** is an interaction diagramthat shows how processes operate with one another and what is their order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called **event diagrams** or **event scenarios**.

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

Sequence diagram is the most common kind of **interaction diagram**, which focuses on the **message**interchange between a number of**lifelines**.
Sequence diagram describes an interaction by focusing on the sequence of messages that are exchanged, along with their corresponding occurrence specifications on the lifelines.
The following nodes and edges are typically drawn in a **UML sequence diagram**: **lifeline**, **execution-specification**, **message**, fragment, interaction, **state invariant**, **continuation**, **destruction occurrence**.

4c. ENTITY RELATIONSHIP DIAGRAM

In software engineering, an **entity–relationship model**(**ER model**) is a data model for describing the data or information aspects of a business domain or its process requirements, in an abstract way that lends itself to ultimately being implemented in a database such as a relational. The main components of ER models are entities (things) and the relationships that can exist amongthem.

An entity–relationship model is the result of using a systematic process to describe and define a subject area of business data. It does not define business process; only visualize business data. The data is represented as components (entities) that are linked with each other by relationships that express the dependencies and requirements between them, such as: one building may be divided into zero or more apartments, but one apartment can only be located in one building. Entities may have various properties (attributes) that characterize them. Diagrams created to represent these entities, attributes, and relationships graphically are called entity–relationship diagrams.

An ER model is typically implemented as a database. In the case of a relational database, which stores data in tables, every row of each table represents one instance of an entity. Some data fields in these tables point to indexes in other tables; such pointers are the physical implementation of therelationships.

The three schema approach to software engineering uses three levels of ER models that may bedeveloped.

Conceptual data model

The conceptual ER model normally defines master reference data entities that are commonly used by the organization. Developing

37

an enterprise-wide conceptual ER model is useful to support documenting the data architecture for an organization.A conceptual ER model may be used as the foundation for one or more logical data models . The purpose of the conceptual ER model is then to establish structural metadata commonality for the master data entities between the set of logical ER models. The conceptual data model may be used to form commonality relationships between ER models as a basis for data model integration.

Logical data model

The logical ER model contains more detail than the conceptual ER model. In addition to master data entities, operational and transactional data entities are now defined. The details of each data entity are developed and the relationships between these data entities are established. The logical ER model is however developed independent of technology into which it can beimplemented.

Physical data model

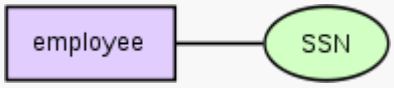
One or more physical ER models may be developed from each logical ER model. The physical ER model is normally developed to be instantiated as a database. Therefore, each physical ER model must contain enough detail to produce a database and each physical ER model is technology dependent since each database managementsystem is somewhatdifferent.
The physical model is normally instantiated in the structural metadata of a database management system as relational database objects such as database tables, database indexes such as unique keyindexes, and

database constraints such as a foreign key constraint or a commonality constraint. The ER model is also normally used to design modifications to the relational database objects and to maintain the structural metadata of the database.

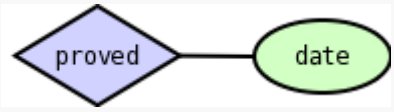
The first stage of information system design uses these models during the requirements analysis to describe information needs or the type of information that is to be stored in a database. The datamodelingtechnique can be used to describe any ontology (i.e. an overview and classifications of used terms and their relationships) for a certain area of interest. In the case of the design of an information system that is based on a database, the conceptual data model is, at a later stage (usually called logical design), mapped to a logical datamodel, such as the relational model; this in turn is mapped to a physical model during physical design. Note that sometimes, both of these phases are referred to as "physical design". It is also used in database managementsystem.

Entity–relationship modeling

Two related entities



An entity with an attribute



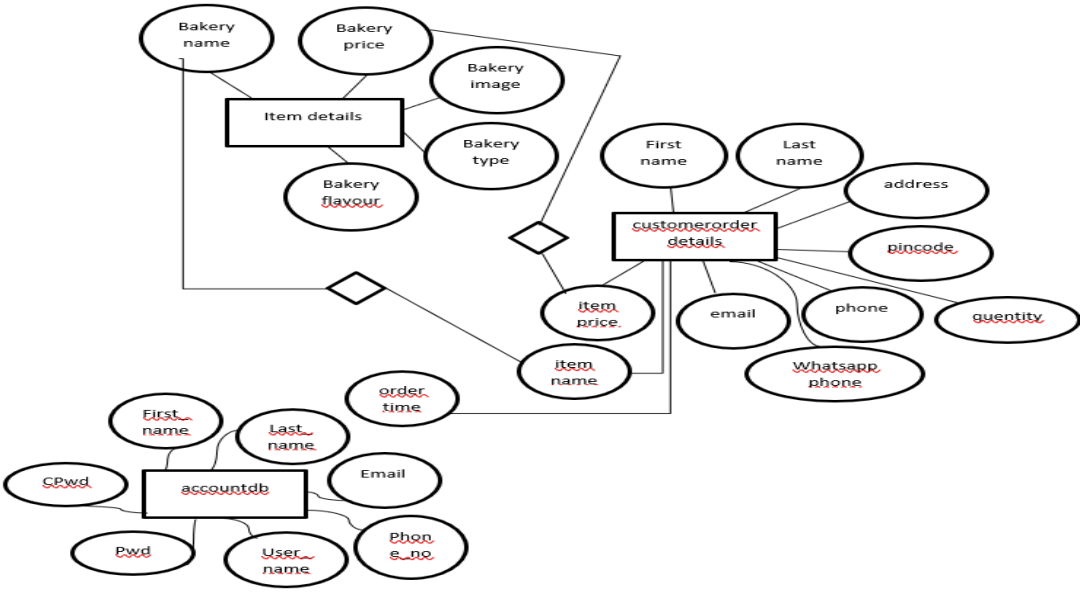
A relationship with an attribute

Primary key

Cardinality constraints are expressed as follows:

- A double line indicates a *participation constraint*, totalityorsubjectivity: all entities in the entity set must participate in *at least one* relationship in the relationshipset;
- an arrow from entity set to relationship set indicates a key constraint, i.e.injectivity: each entity of the entity set can participate in *at most one* relationship in the relationshipset;
- A thick line indicates both, i.e. bijectivity: each entity in the entity set is involved in *exactly one* relationship.
- An underlined name of an attribute indicates that it is a key: two different entities or relationships with this attribute always have different values for thisattribute.

ER-DIAGRAM


Bakery name
Bakery price
Bakery image
Item details
Bakery type
Bakery flavour
First name
Last name
address
customerorder details
pincode
item price
email
phone
quentity
item name
Whatsapp phone
order time
First name
Last name
CPwd
accountdb
Email
Pwd
User name
Phone_no

4d. USE CASE DIAGRAM

A **use case diagram** at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use casesin which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams aswell.

So only static behavior is not sufficient to model a system rather dynamic behavior is more important than static behavior. In UML there are five diagrams available to model dynamic nature and use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. So use case diagrams are consists of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of asystem.

So to model the entire system numbers of use case diagrams are used.

The purpose of use case diagram is to capture the dynamic aspect of a system. But this definition is too generic to describe the purpose.

Because other four diagrams (activity, sequence, collaboration and State chart) are also having the same purpose. So we will look into some specific purpose which will distinguish it from other four diagrams.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So when a system is analyzed to gather its functionalities use cases are prepared and actors areidentified.

Now when the initial task is complete use case diagrams are modelled to present the outside view.

So in brief, the purposes of use case diagrams can be as follows:

- Used to gather requirements of asystem.
- Used to get an outside view of asystem.
- Identify external and internal factors influencing thesystem.
- Show the interacting among the requirements

areactors.HowtodrawUseCaseDiagram?

Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases.

So we can say that uses cases are nothing but the system functionalities written in an organized manner. Now the second things which are relevant to the use cases are the actors. Actors can be defined as something that interacts with the system.

The actors can be human user, some internal applications or may be some external applications. So in a brief when we are planning

to draw a use case diagram we should have the following items identified.

- Functionalities to be represented as an usecase
- Actors
- Relationships among the use cases andactors.

Use case diagrams are drawn to capture the functional requirements of a system. So after identifying the above items we have to follow the following guidelines to draw an efficient use case diagram.

- The name of a use case is very important. So the name should be chosen in such a way so that it can identify the functionalitiesperformed.
- Give a suitable name foractors.
- Show relationships and dependencies clearly in thediagram.
- Do not try to include all types of relationships. Because the main purpose of the diagram is to identifyrequirements.
- Use note whenever required to clarify some important points.

4.e MODULARIZATION DETAILS

As Modularization has gained increasing focus from companies outside its traditional industries of aircraft and automotive, more and more companies turn to it as strategy and product development tool. I intend to explain the importance aspects of modularization and how it should be initiated within a company. After determining the theoretical steps of modularization success described in literature, I intend to conduct a multiple case study of companies who have implemented modularization in order to find how real world modularization was initiated and used to improve the company's competitiveness. By combining theory and practical approach to modularization I will derive at convergence and divergence between theoretical implementation to modularization and real world implementation to modularization. This gives a valuable input for both implantations in companies as well as new aspects to befurther.

DATA INTEGRITY AND CONSTRAINTS

Data integrity is normally enforced in a database systemby a series of integrityconstraintsor rules. Three types of integrity constraints are an inherent part of the relational data model: entity integrity, referential integrity and domain integrity:

- *Entity integrity*concerns the concept of aprimary key. Entity integrity is an integrity rule which states that every table must have a primary key and that the column or columns chosen to be the primary key should be unique and notnull.

- Concerns the concept of a foreign key. The referential integrity rule states that any foreign-key value can only be in one of two states. The usual state of affairs is that theforeign-keyvaluereferstoaprimarykeyvalueofsometableinthedatabase.

Occasionally, and this will depend on the rules of the data owner, a foreign-key value can be null. In this case we are explicitly saying that either there is no relationship between the objects represented in the database or that this relationship is unknown.

- *Domain integrity* specifies that all columns in a relational database must be declared upon a defined domain. The primary unit of data in the relational data model is the data item. Such data items are said to be non-decomposable or atomic. A domain is a set of values of the same type.

4f. DATABASE DESIGN

A database is an organized mechanism that has capability of storing information through which a user can retrieve stored information in an effective and efficient manner. The data is the purpose of any database and must be protected.

The database design is two level processes. In the first step, user requirements are gathered together and a database is designed which will meet these requirements as clearly as possible. This step is called information Level design and it is taken independent of any individual DBMS.

In the following snapshots we display the way we have used SQL Server as the back-end RDBMS for our project and the various entities that have been used along with their table definition and table data.

DATA DICTIONARY

```
from django.db import models

# Create your models here.

class itemdetails(models.Model):
    bakeryname=models.CharField(max_length=50)
    bakeryprice=models.CharField(max_length=10)
    bakeryimage=models.CharField(max_length=500)
    bakerytype=models.CharField(max_length=50)
    bakeryflavour=models.CharField(max_length=50)
    class Meta:
        db_table="itemdetails"
class accountdb(models.Model):
    First_name=models.CharField(max_length=50)
    Last_name=models.CharField(max_length=20)
    Email=models.EmailField()
    Phone_no=models.IntegerField(max_length=15)
    User_name=models.CharField(max_length=50)
    Pwd=models.CharField(max_length=20)
    Cpwd=models.CharField(max_length=20)
    class Meta:
        db_table="accountdb"
class customerorderdetails(models.Model):
    firstname=models.CharField(max_length=50)
    lastname=models.CharField(max_length=50)
    address=models.CharField(max_length=300)
    pincode=models.CharField(max_length=10)
    phone=models.CharField(max_length=15)
    email=models.EmailField()
    altphone=models.CharField(max_length=15)
```

```
quentity=models.CharField(max_length=20)
itemname=models.CharField(max_length=50)
itemprice=models.CharField(max_length=10)
ordertime=models.CharField(max_length=100)
class Meta:
    db_table="customerorderdetails"
```

OUTPUT SCREEN

5a. USER INTERFACE DESIGN

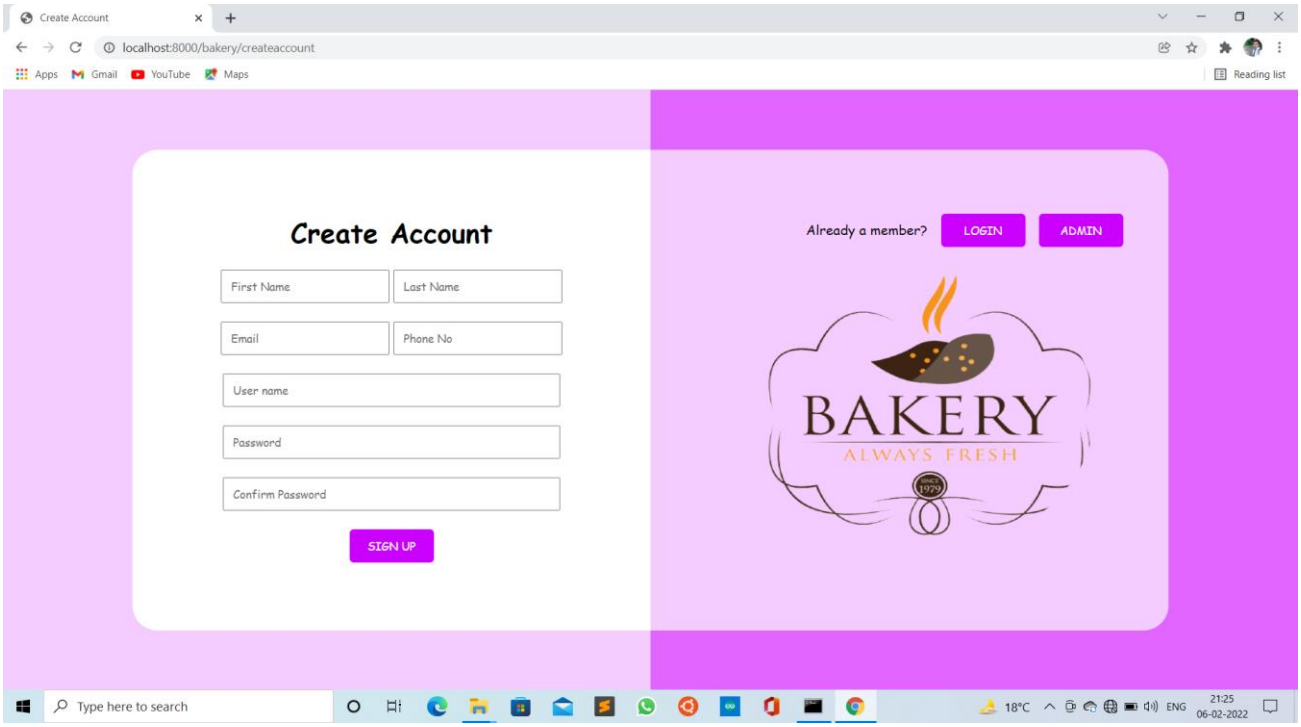
User interface design (UID) or **user interface engineering** is the designof user interfacesfor machinesand software, such as computers, home appliances, mobile devices, and other electronicdevices, with the focus on maximizing the user experience. The goal of user interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals (user-centereddesign).

Good user interface design facilitates finishing the task at hand without drawing unnecessary attention to it. Graphic designand typography are utilized to support its usability, influencing how the user performs certain interactions and improving the aesthetic appeal of the design; design aesthetics may enhance or detract from the ability of users to use the functions of the interface. The design process must balance technical functionality and visual elements (e.g., mental model) to create a system that is not only operational but also usable and adaptable to changing user needs.

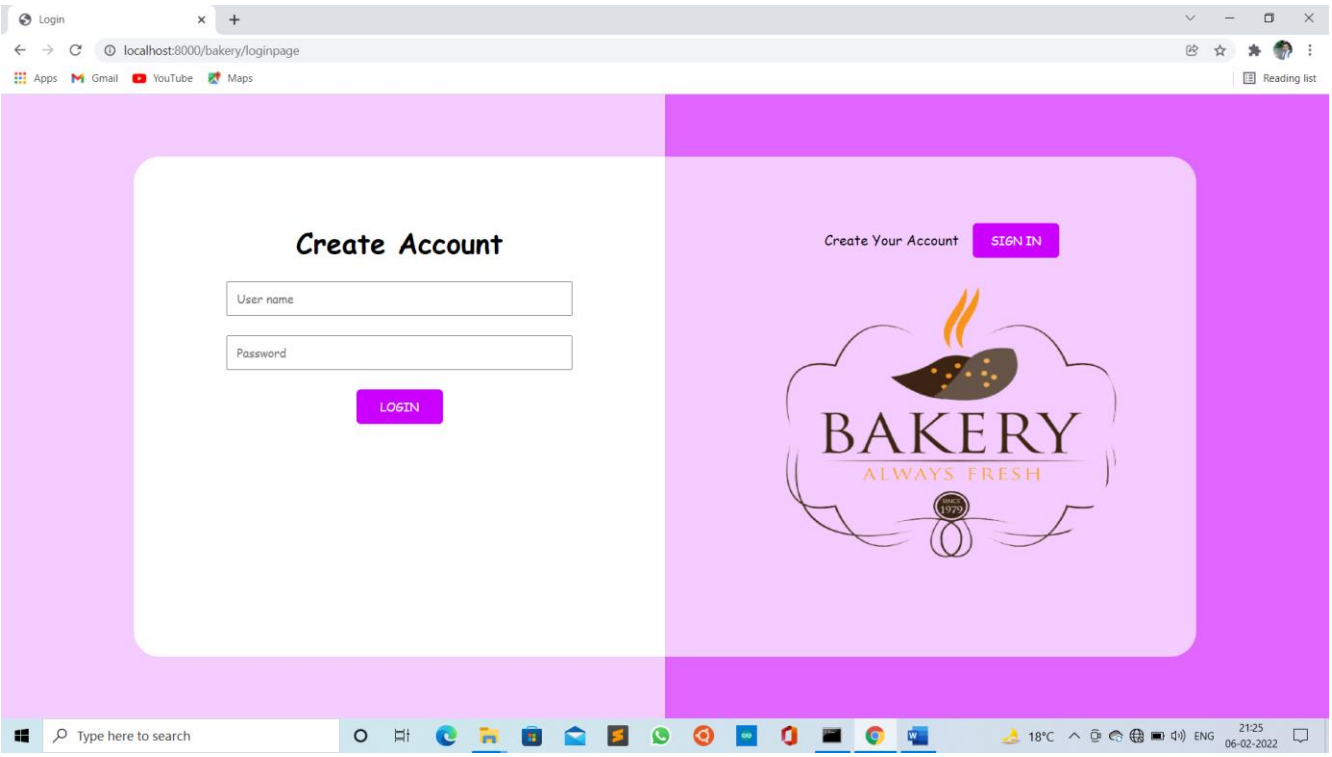
Interface design is involved in a wide range of projects from computer systems, to cars, to commercial planes; all of these projects involve much of the same basic human interactions yet also require some unique skills and knowledge. As a result, designers tend to specialize in certain types of projects and have skills centered on their expertise, whether that be software design, user research, web design, or industrialdesign.

SNAPSHOTS

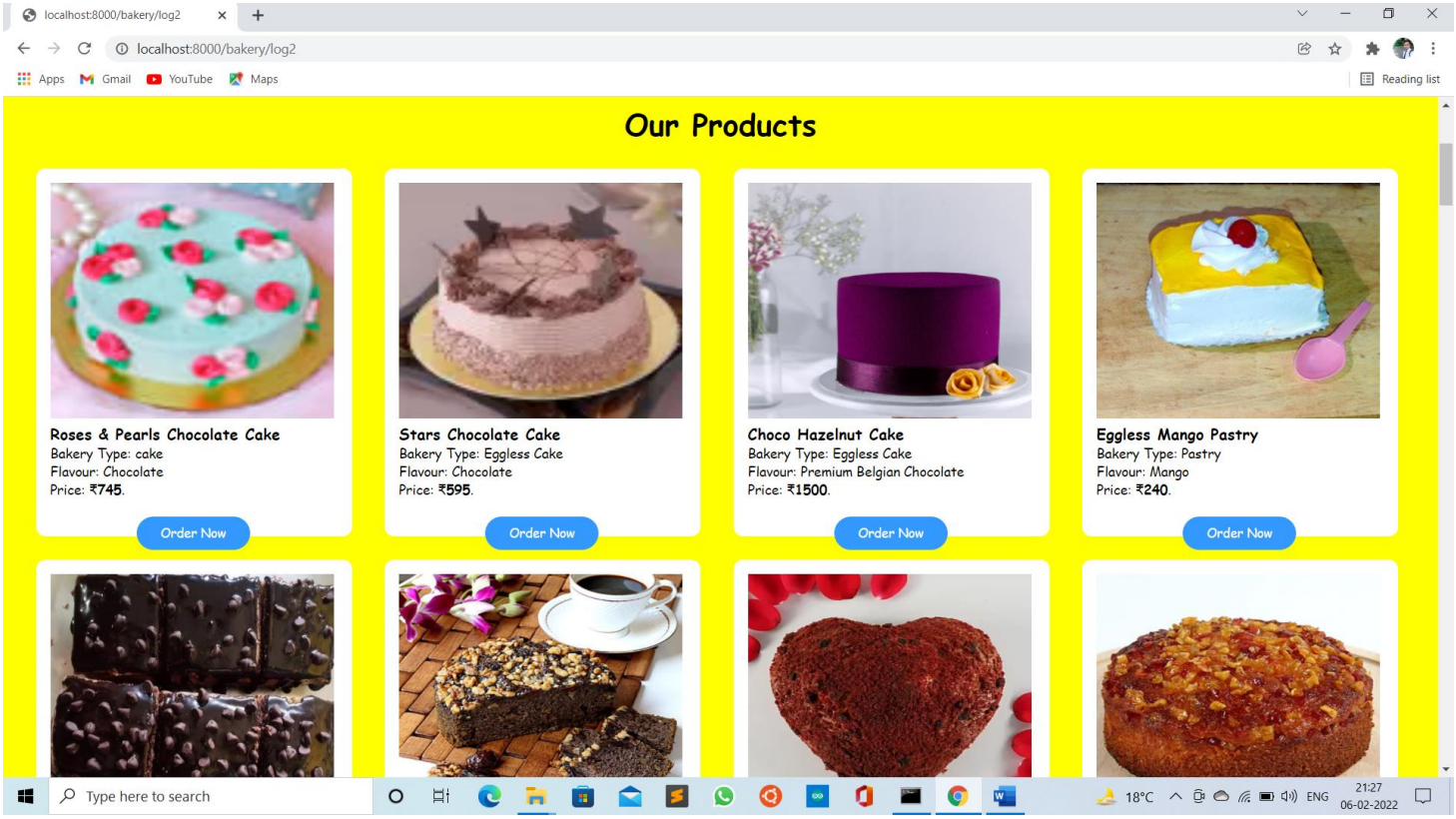
Master Home page



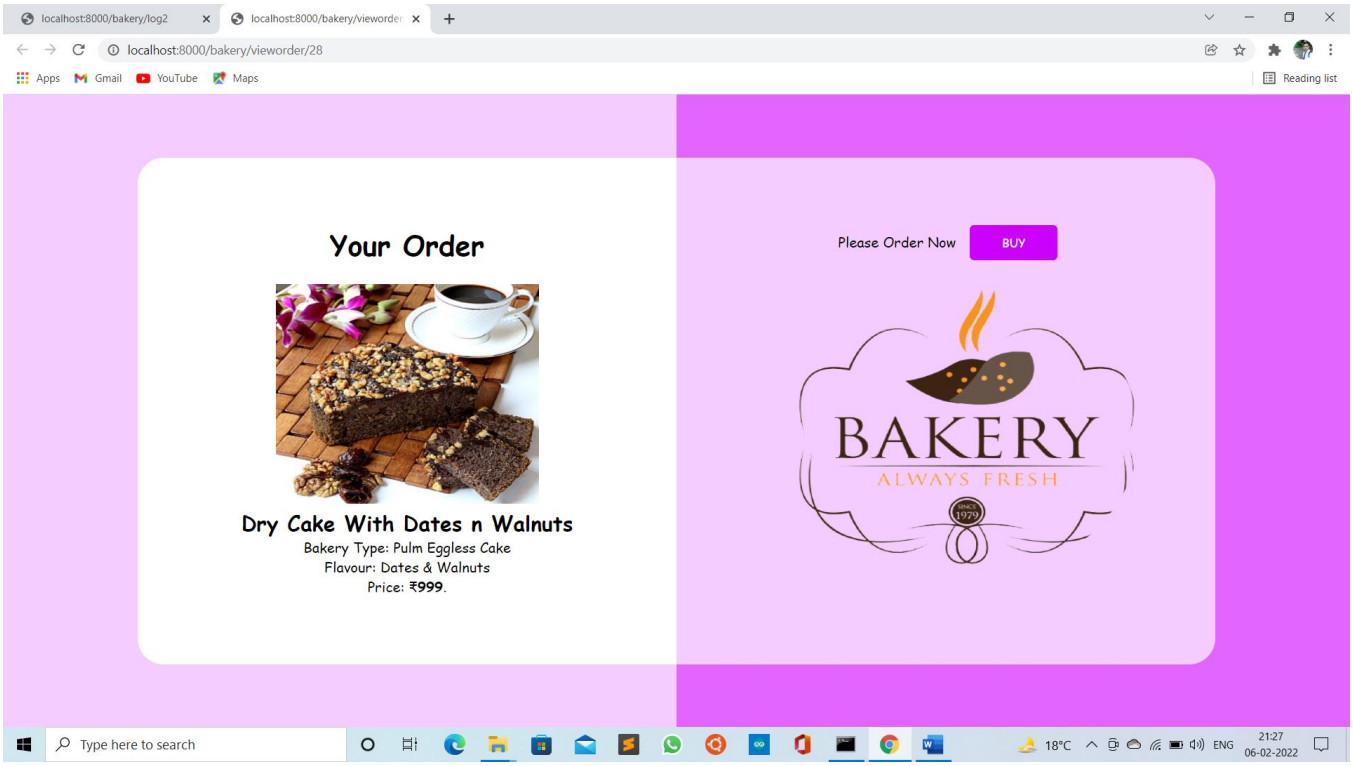
Login Page


Login
localhost:8000/bakery/loginpage
Apps
Gmail
YouTube
Maps
Reading list
Create Account
User name
Password
LOGIN
Create Your Account
SIGN IN
BAKERY
ALWAYS FRESH
Type here to search
18°C
ENG
21:25
06-02-2022

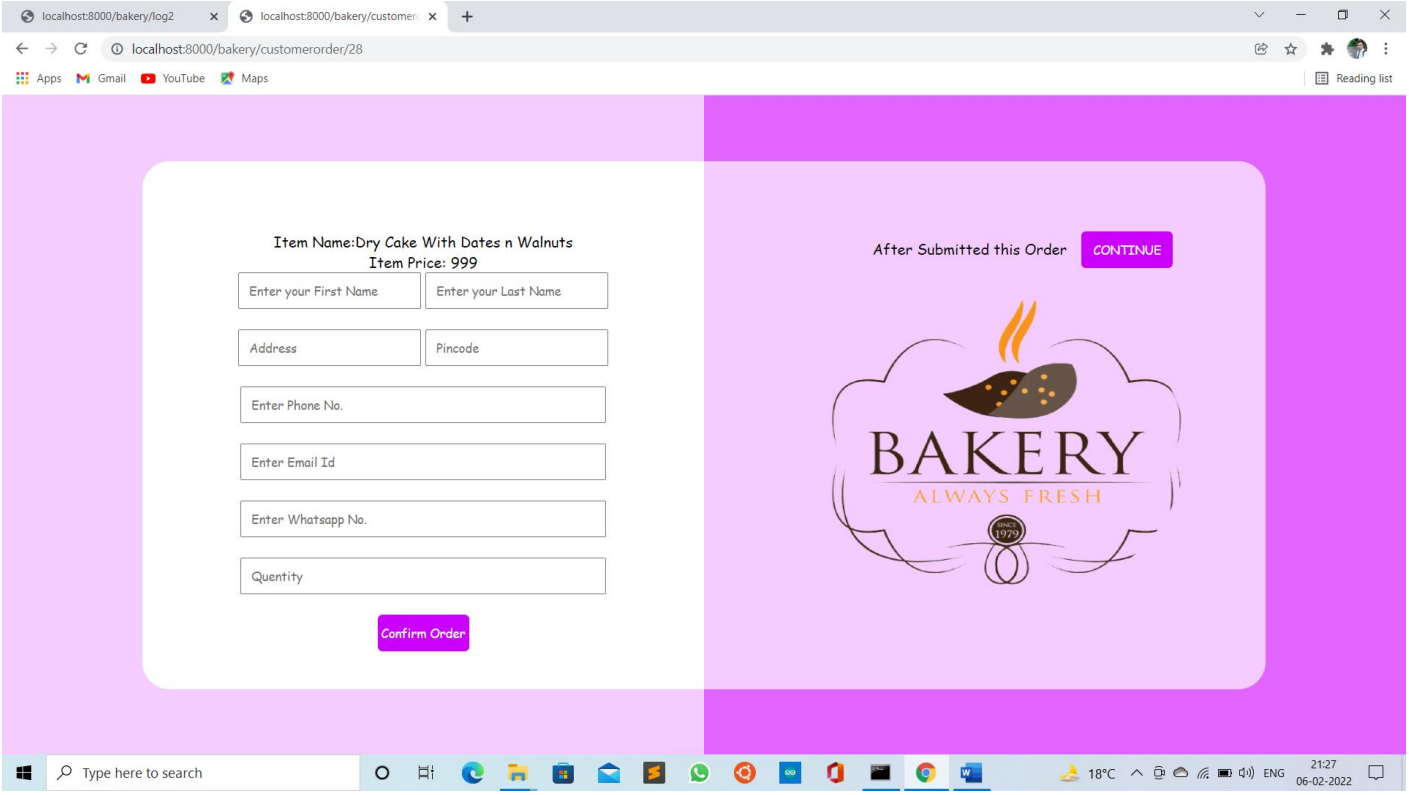
Customer Home Page



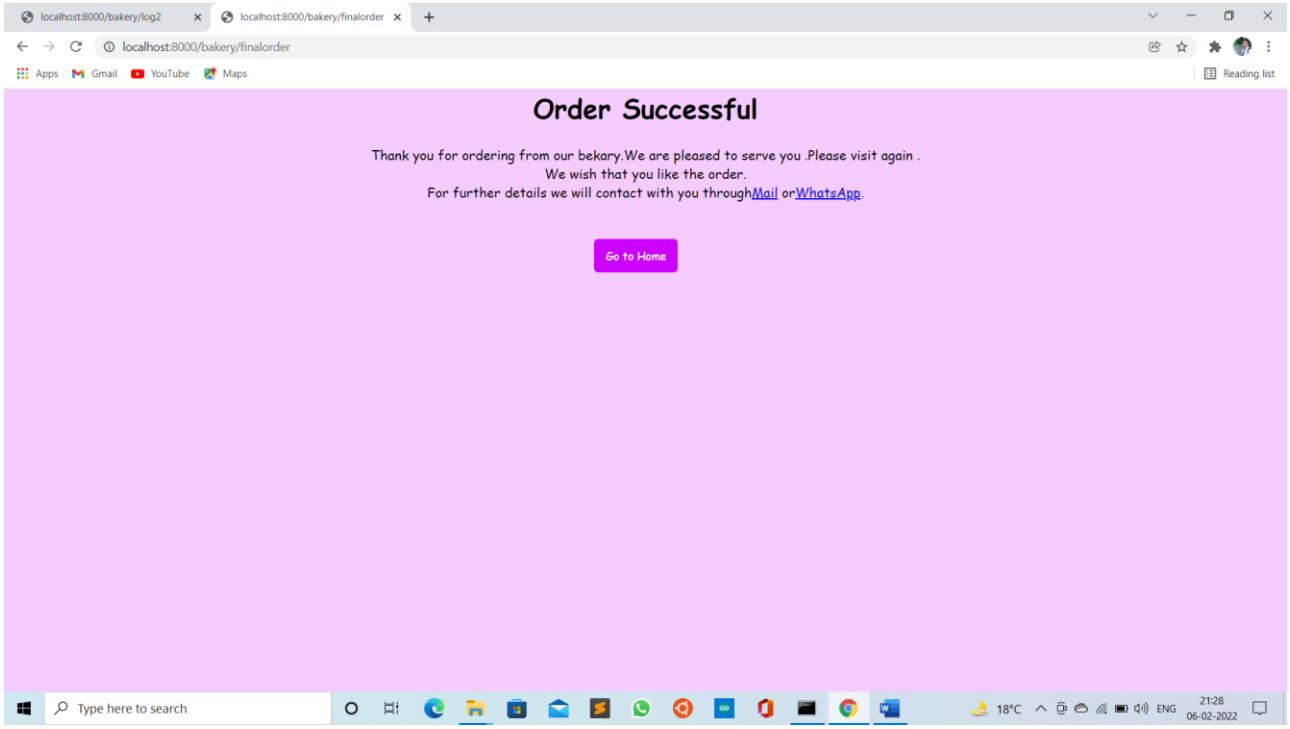
Customer Order Page



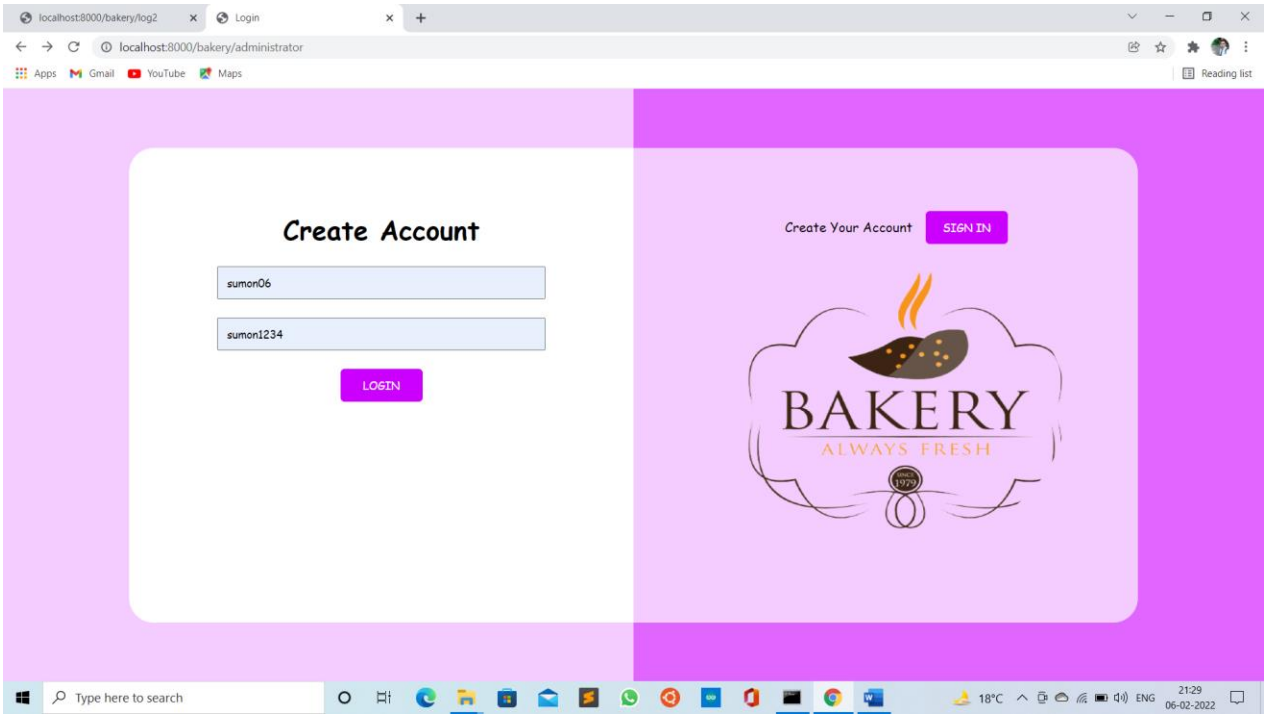
Customer Order Placeing Details



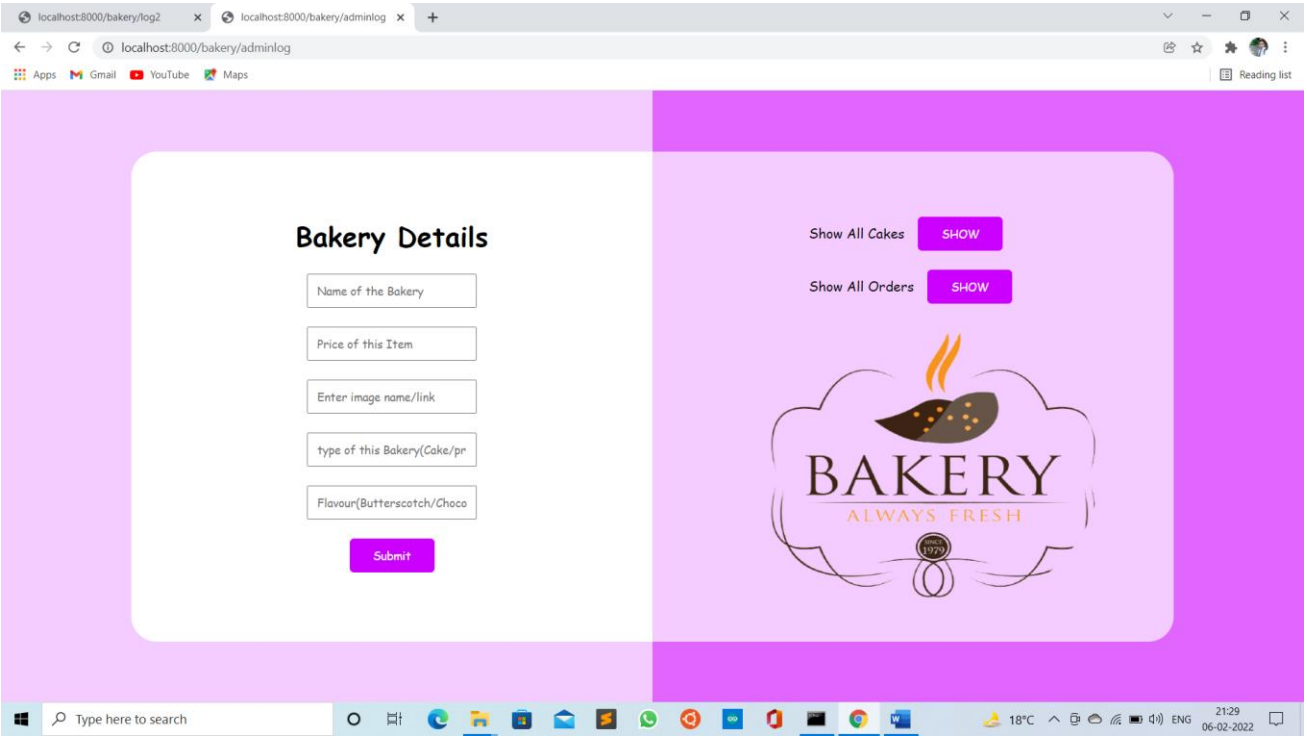
Order Confirmation Page



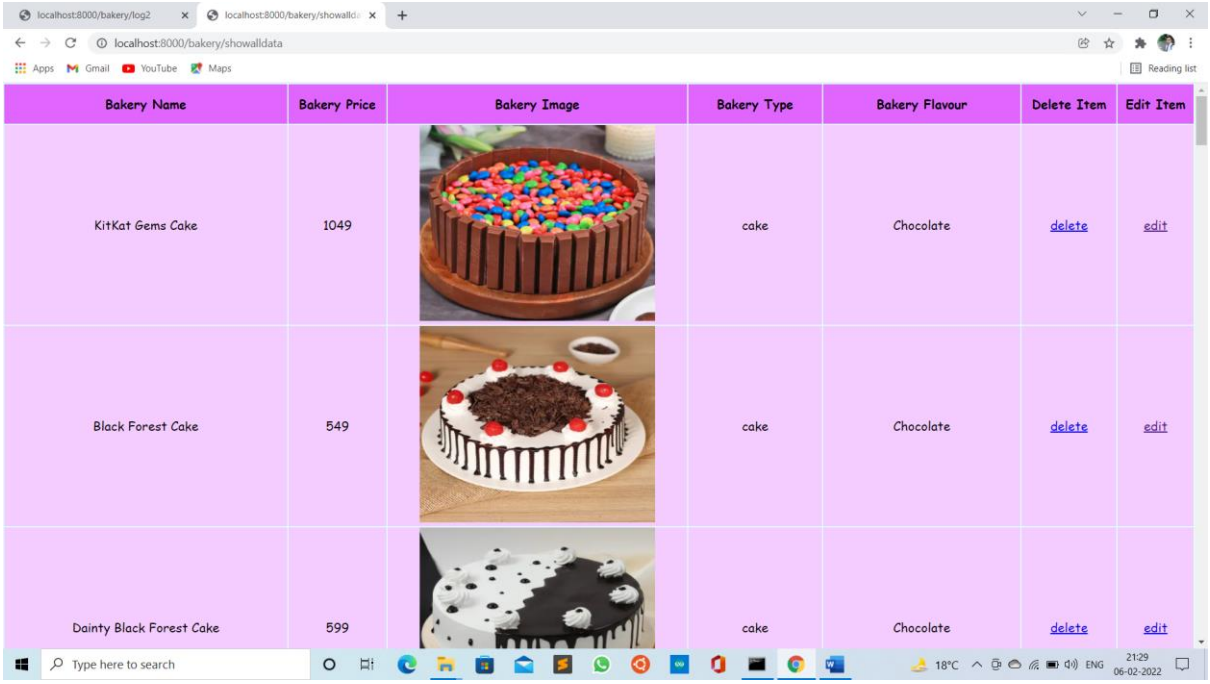
Admin Login Page



Adding New Items page

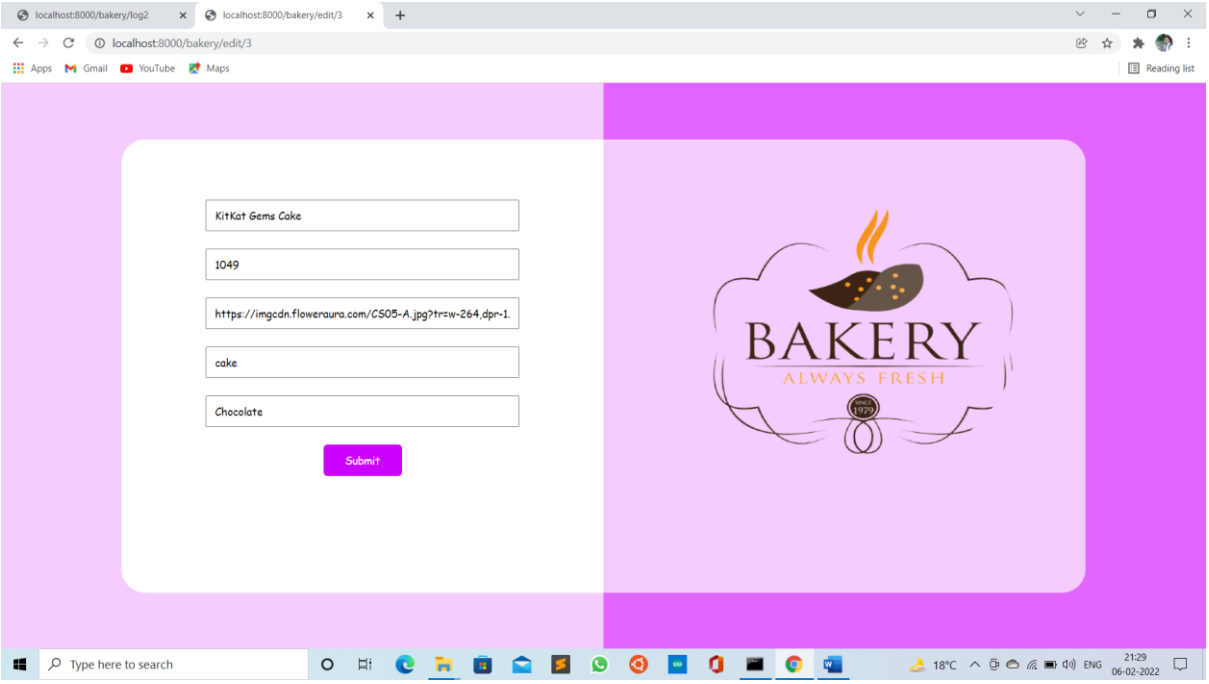


Items Showing & Editing page

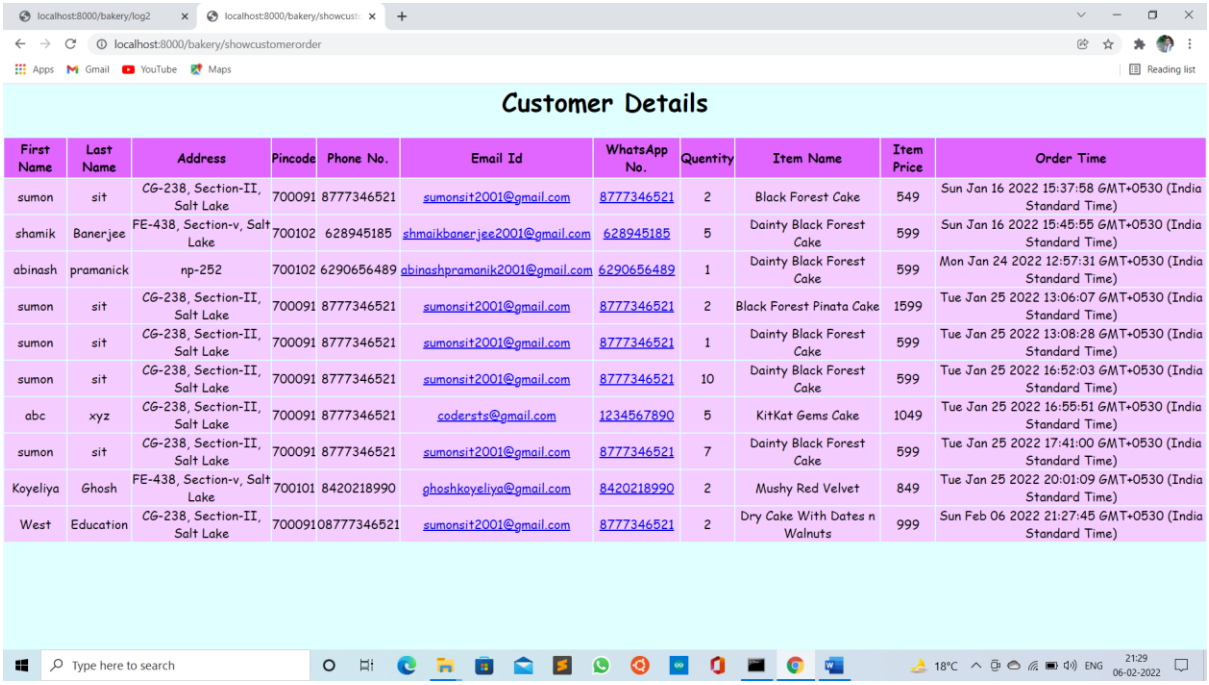


Bakery Name	Bakery Price	Bakery Image	Bakery Type	Bakery Flavour	Delete Item	Edit Item
KitKat Gems Cake	1049		cake	Chocolate	delete	edit
Black Forest Cake	549		cake	Chocolate	delete	edit
Dainty Black Forest Cake	599		cake	Chocolate	delete	edit

Adding Items page



Customer Details Page



Customer Details

First Name	Last Name	Address	Pincode	Phone No.	Email Id	WhatsApp No.	Quentity	Item Name	Item Price	Order Time
sumon	sit	CG-238, Section-II, Salt Lake	700091	8777346521	sumonsit2001@gmail.com	8777346521	2	Black Forest Cake	549	Sun Jan 16 2022 15:37:58 GMT+0530 (India Standard Time)
shamik	Banerjee	FE-438, Section-v, Salt Lake	700102	628945185	shmaikbanerjee2001@gmail.com	628945185	5	Dainty Black Forest Cake	599	Sun Jan 16 2022 15:45:55 GMT+0530 (India Standard Time)
abinash	pramanick	np-252	700102	6290656489	abinashpramanik2001@gmail.com	6290656489	1	Dainty Black Forest Cake	599	Mon Jan 24 2022 12:57:31 GMT+0530 (India Standard Time)
sumon	sit	CG-238, Section-II, Salt Lake	700091	8777346521	sumonsit2001@gmail.com	8777346521	2	Black Forest Pinata Cake	1599	Tue Jan 25 2022 13:06:07 GMT+0530 (India Standard Time)
sumon	sit	CG-238, Section-II, Salt Lake	700091	8777346521	sumonsit2001@gmail.com	8777346521	1	Dainty Black Forest Cake	599	Tue Jan 25 2022 13:08:28 GMT+0530 (India Standard Time)
sumon	sit	CG-238, Section-II, Salt Lake	700091	8777346521	sumonsit2001@gmail.com	8777346521	10	Dainty Black Forest Cake	599	Tue Jan 25 2022 16:52:03 GMT+0530 (India Standard Time)
abc	xyz	CG-238, Section-II, Salt Lake	700091	8777346521	codersts@gmail.com	1234567890	5	KitKat Gems Cake	1049	Tue Jan 25 2022 16:55:51 GMT+0530 (India Standard Time)
sumon	sit	CG-238, Section-II, Salt Lake	700091	8777346521	sumonsit2001@gmail.com	8777346521	7	Dainty Black Forest Cake	599	Tue Jan 25 2022 17:41:00 GMT+0530 (India Standard Time)
Koyeliya	Ghosh	FE-438, Section-v, Salt Lake	700101	8420218990	ghoshkoyeliya@gmail.com	8420218990	2	Mushy Red Velvet	849	Tue Jan 25 2022 20:01:09 GMT+0530 (India Standard Time)
West	Education	CG-238, Section-II, Salt Lake	700091	08777346521	sumonsit2001@gmail.com	8777346521	2	Dry Cake With Dates n Walnuts	999	Sun Feb 06 2022 21:27:45 GMT+0530 (India Standard Time)

CODES

HTML FILES

administrator login

```
<!DOCTYPE html>
<html>
{% load static %}
<head>
<title>Login</title>
</head>
<link rel="stylesheet" type="text/css" href="{% static 'css/style2.css' %}">
<body>
                <div class="Big">
                  <div class="BigLeft">
                        <div class="MiniLeft">
                              <h1>Create Account</h1><br>
                              <form method="post" action="adminlog">
                                    {% csrf_token %}
                                    <input type="text" placeholder="User name" name="User_name"
class="textbox2"><br><br>
                                    <input type="text" placeholder="Password" name="Pwd"
class="textbox2"><br><br>
                                    <input type="submit" value="LOGIN" class="signup">
                              </form>
                        </div>
                  </div>
                  <div class="BigRight">
                        <div class="MiniRight">
                         <p>Create Your Account  <a href="createaccount"><button
class="login">SIGN IN</button></a></p><br>
                              <img src="{% static 'img/Crlogo.png' %}">
                        </div>
                  </div>
                </div>
</body>
</html>
```

Createaccountuser

```
<!DOCTYPE html>
<html>
{% load static %}
<head>
		<title></title>
		<link rel="stylesheet" type="text/css" href="{% static 'css/styleaccountcreate.css' %}">
</head>
<body>
		<div class="Big">
		  <div class="BigLeft">
			<div class="MiniLeft">
				<h1 class="h1">Create Account</h1>
				<form method="post" action="cap">
					{% csrf_token %}
					<input class="t1" type="text" placeholder="Enter the First Name"
name="firstname">
					<input class="t1" type="text" placeholder="Enter the Last Name"
name="lastname"><br><br>
					<input class="t1" type="email" placeholder="Enter the Email id"
name="email">
					<input class="t1" type="tel" placeholder="Phone No"
name="phoneno"><br><br>
					<input class="t2" type="text" placeholder="User name"
name="username"><br><br>
					<input class="t2" type="text" placeholder="Create Password"
name="password"><br><br>
					<input class="t2" type="password" placeholder="Confirm
Password" name="cpassword"><br>

					<input class="btn1" type="submit" value="SIGN UP">
				</form>
			</div>
		  </div>
		  <div class="BigRight">
			<div class="MiniRight">
```

```
                                <span class="span1">Already a member?</span> 
                                <input class="LoginButton" type="submit" value="LOGIN"
name=""><br>
                                <img class="img1" src="{% static 'img/bakery_logo.png' %}">
                        </div>
                    </div>
                </div>
</body>
</html>
```

customerorder

```
<!DOCTYPE html>
<html>
{% load static %}
<head>
                <title></title>
</head>
<link rel="stylesheet" type="text/css" href="{% static 'css/style2.css' %}">
<body>
                <div class="Big">
                  <div class="BigLeft">
                        <div class="MiniLeft">
                                <form method="post" action="../cto/{{ x.id }}">
                                {% csrf_token %}
                                <span>Item Name:</span><span>{{ x.bakeryname }}</span><br>
                                <span>Item Price: </span><span>{{ x.bakeryprice }}</span><br>
                                </form>
                                <form method="post" action="../ceo">
                                {% csrf_token %}
                                <input type="text" placeholder="Enter your First Name"
name="firstname" class="textbox">
                                <input type="text" placeholder="Enter your Last Name"
name="lastname"  class="textbox"><br><br>
                                <input type="text" placeholder="Address" name="address"
class="textbox">
                                <input type="tel" placeholder="Pincode" name="pincode"
class="textbox"><br><br>
                                <input type="tel" placeholder="Enter Phone No." name="phone"
```

```
class="textbox2"><br><br>
                                <input type="email" placeholder="Enter Email Id" name="email"
class="textbox2"><br><br>
                                <input type="tel" placeholder="Enter Whatsapp No." name="altphone"
class="textbox2"><br><br>
                                <input type="number" placeholder="Quentity" name="quentity"
class="textbox2"><br><br>
                                <input type="submit" value="Confirm Order" class="login">
                                <input style="display: none;" type="text" name="itemname" value="{{
x.bakeryname }}"><br><br>
                                <input style="display: none;" type="tel" name="itemprice" value="{{
x.bakeryprice }}"><br><br>
                                <input style="display: none;" type="text" id="time1" value=""
name="ordertime"><br><br>
                                </form>
                        <script type="text/javascript">
                                time= new Date();
                                document.getElementById('time1').value = time;
                        </script>
                                </div>
                        </div>
                        <div class="BigRight">
                                <div class="MiniRight">
                                 <p>After Submitted this Order  <a
href="../bakery/finalorder"><button class="login">CONTINUE</button></a></p><br>
                                <img src="{% static 'img/Crlogo.png' %}">
                                </div>
                        </div>
                    </div>
</body>
</html>
```

editalldata

```
<!DOCTYPE html>
<html>
{% load static %}
<head>
                    <title></title>
```

```
                <link rel="stylesheet" type="text/css" href="{% static 'css/style2.css' %}">
</head>
<body>
                <div class="BigLeft">
                  <div class="MiniLeft">
                        <form method="post" action="../edc/{{ x.id }}">
                              {% csrf_token %}
                              <input type="text" placeholder="Name of the Bakery"
name="bakeryname" value="{{ x.bakeryname }}" class="textbox2"><br><br>
                              <input type="tel" placeholder="Price of this Item" name="bakeryprice"
value="{{ x.bakeryprice }}" class="textbox2"><br><br>
                              <input type="text" name="bakeryimage" placeholder="Enter image
name/link" value="{{ x.bakeryimage }}" class="textbox2"><br><br>
                              <input type="text" placeholder="type of this Bakery(Cake/presti)"
name="bakerytype" value="{{ x.bakerytype }}" class="textbox2"><br><br>
                              <input type="text" placeholder="Flavour(Butterscotch/Chocolate)"
name="bakeryflavour" value="{{ x.bakeryflavour }}" class="textbox2"><br><br>
                              <input type="submit" name="" class="login">
                        </form>
                  </div>
                </div>
                <div class="BigRight">
                  <div class="MiniRight">
                              <img src="{% static 'img/Crlogo.png' %}">
                        </div>
                </div>
</body>
</html>
```

finalorder

```
<!DOCTYPE html>
<html>
{% load static %}
<head>
                <title></title>
                <link rel="stylesheet" type="text/css" href="{% static 'css/style2.css' %}">
</head>
<body bgcolor="#f5ccff">
```

```
                <h1 align="center">Order Successful</h1><br>
                    <p align="center" style="padding-left: 0;" class="data">
                        Thank you for ordering from our bekary.We are pleased to serve
you .Please visit again .<br>We wish that you like the order.<br> For further details we will contact with
you through<a href="mailto:" title="" target="blank">Mail</a> or<a href="https://wa.me/+91" title=""
target="blank">WhatsApp</a>.
                    </p><br><br>
                <a href="loginpage"><button class="login" style="margin-left: 46%;"
align="center">Go to Home </button></a>
</body>
</html>
```

Login Page

```
<!DOCTYPE html>
<html>
{% load static %}
<head>
        <title>Create Account</title>
</head>
<link rel="stylesheet" type="text/css" href="{% static 'css/style2.css' %}">
<body>
        <div class="Big">
          <div class="BigLeft">
                <div class="MiniLeft">
                    <h1>Create Account</h1><br>
                    <form method="post" action="cre">
                        {% csrf_token %}
                        <input type="text" placeholder="First Name" name="First_name"
class="textbox">
                        <input type="text" placeholder="Last Name" name="Last_name"
class="textbox"><br><br>
                        <input type="text" placeholder="Email" name="Email"
class="textbox">
                        <input type="text" placeholder="Phone No" name="Phone_no"
class="textbox"><br><br>
                        <input type="text" placeholder="User name" name="User_name"
class="textbox2"><br><br>
                        <input type="text" placeholder="Password" name="Pwd"
class="textbox2"><br><br>
```

```
                                        <input type="text" placeholder="Confirm Password"
name="Cpwd" class="textbox2"><br><br>
                                        <input type="submit" value="SIGN UP" class="signup">
                                </form>
                        </div>
                </div>
                <div class="BigRight">
                        <div class="MiniRight">
                                <p>Already a member?  <a href="loginpage"><button
class="login">LOGIN</button></a>  <a href="administrator"><button
class="login">ADMIN</button></a></p><br>
                                <img src="{% static 'img/Crlogo.png' %}">
                        </div>
                </div>
            </div>
</body>
</html>
```

Login

```
<!DOCTYPE html>
<html>
<head>
<title>Login</title>
</head>
<body>
            <div class="Big">
              <div class="BigLeft">
                        <div class="MiniLeft">
                                <h1>Create Account</h1><br>
                                <form method="post" action="log">
                                        {% csrf_token %}
                                        <input type="text" placeholder="First Name" name="First_name"
class="textbox">
                                        <input type="text" placeholder="Last Name" name="Last_name"
class="textbox"><br><br>
                                        <input type="text" placeholder="Email" name="Email"
class="textbox">
                                        <input type="text" placeholder="Phone No" name="Phone_no"
class="textbox"><br><br>
```

```
                                        <input type="text" placeholder="User name" name="User_name"
class="textbox2"><br><br>
                                        <input type="Password" placeholder="Password" name="Pwd"
class="textbox2"><br><br>
                                        <input type="Password" placeholder="Confirm Password"
name="Cpwd" class="textbox2"><br><br>
                                        <input type="submit" value="SIGN UP" class="signup">
                                </form>
                        </div>
                </div>
                <div class="BigRight">
                        <div class="MiniRight">
                                <p>Create Your Account  <button class="login">SIGN
IN</button></p><br>
                                <img src="{% static 'img/Crlogo.png' %}">
                        </div>
                </div>
        </div>
</body>
</html>
```

Login2

```
<!DOCTYPE html>
<html>
{% load static %}
<head>
<title>Login</title>
</head>
<link rel="stylesheet" type="text/css" href="{% static 'css/style2.css' %}">
<body>
        <div class="Big">
                <div class="BigLeft">
                        <div class="MiniLeft">
                                <h1>Create Account</h1><br>
                                <form method="post" action="log2">
                                        {% csrf_token %}
                                        <input type="text" placeholder="User name" name="User_name"
class="textbox2"><br><br>
```

```
                                        <input type="password" placeholder="Password" name="Pwd"
class="textbox2"><br><br>
                                        <input type="submit" value="LOGIN" class="signup">
                                </form>
                        </div>
                </div>
                <div class="BigRight">
                        <div class="MiniRight">
                                <p>Create Your Account  <a href="createaccount"><button
class="login">SIGN IN</button></a></p><br>
                                <img src="{% static 'img/Crlogo.png' %}">
                        </div>
                </div>
            </div>
</body>
</html>
```

showalldata

```
<!DOCTYPE html>
<html>
{% load static %}
<head>
            <title></title>
            <link rel="stylesheet" type="text/css" href="{% static 'css/style5.css' %}">
</head>
<body>
            <div>
              <table>
                        <tr align="center">
                                <th>Bakery Name</th><th>Bakery Price</th><th>Bakery
Image</th><th>Bakery Type</th><th>Bakery Flavour</th><th>Delete Item</th><th>Edit Item</th>
                        </tr>
                        {% for i in x %}
                                <tr align="center"><td>{{ i.bakeryname }}</td><td>{{ i.bakeryprice
}}</td><td><img height="250" width="300" src="{{ i.bakeryimage }}"></td><td>{{ i.bakerytype
}}</td><td>{{ i.bakeryflavour }}</td><td><a href="delete/{{ i.id }}">delete</a></td><td><a
href="edit/{{ i.id }}">edit</a></td></tr>
                        {% endfor %}
              </table>
```

```
                    </div>
</body>
</html>
```

Showcustomerorder

```
<!DOCTYPE html>
<html>
{% load static %}
<head>
                    <title></title>
                    <link rel="stylesheet" type="text/css" href="{% static 'css/style5.css' %}">
</head>
<body>
                    <h1 align="center">Customer Details</h1><br>
                    <table>
                              <tr align="center">
                                        <th>First Name</th><th>Last
Name</th><th>Address</th><th>Pincode</th><th>Phone No.</th><th>Email Id</th><th>WhatsApp
No.</th><th>Quentity</th><th>Item Name</th><th>Item Price</th><th>Order Time</th>
                              </tr>
                              {% for i in x %}
                                        <tr align="center">
                                                  <td>{{ i.firstname }}</td><td>{{ i.lastname }}</td><td>{{
i.address }}</td><td>{{ i.pincode }}</td><td>{{ i.phone }}</td><td><a title="{{ i.email }}"
href="mailto:{{i.email}}">{{ i.email }}</a></td><td><a href="https://wa.me/+91{{ i.altphone }}"
title="{{ i.altphone }}" target="blank">{{ i.altphone }}</a></td><td>{{ i.quentity }}</td><td>{{
i.itemname }}</td><td>{{ i.itemprice }}</td><td>{{ i.ordertime }}</td>
                                        </tr>
                              {% endfor %}
                    </table>
</body>
</html>
```

Showmaindata

```
<!DOCTYPE html>
<html>
{% load static %}
<head>
```

```
                <title></title>
                <link rel="stylesheet" type="text/css" href="{% static 'css/style3.css' %}">
</head>
<body>
                <div class="top">

                </div>
                <div class="display">
                  <h1>Our Products</h1>
                        {% for i in x %}
                                <div class="cake">
                                        <img height="250" width="300" src="{{ i.bakeryimage }}"><br>
                                        <h3>{{ i.bakeryname }}</h3>
                                        <span>Bakery Type: </span><span>{{ i.bakerytype
}}</span><br>
                                        <span>Flavour: </span><span>{{ i.bakeryflavour
}}</span><br>
                                        <span>Price: ₹<b>{{ i.bakeryprice }}</b>.</span><br><br>
                                        <a target="_blank" href="vieworder/{{ i.id }}"><button>Order
Now</button></a>
                                </div>
                        {% endfor %}
                </div>
</body>
</html>
```

uploaddata

```
<!DOCTYPE html>
<html>
{% load static %}
<head>
                <title></title>
                <link rel="stylesheet" type="text/css" href="{% static 'css/style2.css' %}">
</head>
<body>
                <div class="BigLeft">
                  <div class="MiniLeft">
                        <h1>Bakery Details</h1>
                        <form method="post" action="uld">
```

```
                                        {% csrf_token %}
                                        <br><input type="text" placeholder="Name of the Bakery"
name="bakeryname" class="textbox"><br><br>
                                        <input type="tel" placeholder="Price of this Item" name="bakeryprice"
class="textbox"><br><br>
                                        <input type="text" placeholder="Enter image name/link"
name="bakeryimage" class="textbox"><br><br>
                                        <input type="text" placeholder="type of this Bakery(Cake/presti)"
name="bakerytype" class="textbox"><br><br>
                                        <input type="text" placeholder="Flavour(Butterscotch/Chocolate)"
name="bakeryflavour" class="textbox"><br><br>
                                        <input type="submit" name="" class="login">
                                </form>
                        </div>
                </div>
                <div class="BigRight">
                        <div class="MiniRight">
                                <p>Show All Cakes  <a href="showalldata"><button
class="login">SHOW</button></a></p><br>
                                <p>Show All Orders  <a href="showcustomerorder"><button
class="login">SHOW</button></a></p><br>
                                <img src="{% static 'img/Crlogo.png' %}">
                        </div>
                </div>
</body>
</html>
```

vieworder

```
<!DOCTYPE html>
<html>
{% load static %}
<head>
                <title></title>
</head>
<link rel="stylesheet" type="text/css" href="{% static 'css/style4.css' %}">
<body>
                <div class="Big">
                        <div class="BigLeft">
                                <div class="MiniLeft">
```

```
                                <h1>Your Order</h1><br>
                                <form method="post" action="../vld/{{ x.id }}">
                                {% csrf_token %}
                                <img height="250" width="300" src="{{ x.bakeryimage }}"><br>
                                        <h2>{{ x.bakeryname }}</h2>
                                        <span>Bakery Type: </span><span>{{ x.bakerytype
}}</span><br>
                                        <span>Flavour: </span><span>{{ x.bakeryflavour
}}</span><br>
                                        <span>Price: ₹<b>{{ x.bakeryprice }}</b>.</span><br><br>
                        </form>
                        </div>
                </div>
                <div class="BigRight">
                        <div class="MiniRight">
                         <p>Please Order Now  <a
href="../customerorder/{{x.id}}"><button class="login">BUY</button></a></p><br>
                                <img src="{% static 'img/Crlogo.png' %}" class="logo">
                        </div>
                </div>
        </div>
</body>
</html>
```

welcome2

```
<!DOCTYPE html>
<html>
<head>
        <title>welcome</title>
</head>
<body>
        <h1>Successfully Loged in.</h1>
</body>
</html
```

CSS FLES

style2

```
*
{
  margin:0;
  padding:0;
  box-sizing: border-box;
  font-family: 'Brush Script MT', cursive;
}
.Big
{
  position: relative;
  width: 100%;
  height: 100vh;
  background-color: pink;
}
.BigLeft
{
  position: relative;
  float: left;
  width: 50%;
  height: 100vh;
  background-color: #f5ccff;
}
.BigRight
{
  position: relative;
  float: left;
  width: 50%;
  height: 100vh;
  background-color: #e066ff;
}
.MiniLeft
{
```

```
  position: relative;
  top: 10%;
  left: 20%;
  float: left;
  width: 80%;
  height: 80vh;
  background-color: white;
  border-radius: 30px 0px 0px 30px;
  text-align: center;
  padding-top: 10%;
}
.MiniRight
{
  position: relative;
  top: 10%;
  float: left;
  width: 80%;
  height: 80vh;
  background-color: #f5ccff;
  border-radius: 0px 30px 30px 0px;
  padding-top: 10%;
  padding-left: 10%;
}
.login
{
  width: 100px;
  height: 40px;
  color: white;
  background-color: #cc00ff;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}
.signup
{
  width: 100px;
  height: 40px;
  color: white;
  background-color: #cc00ff;
  border-radius: 5px;
```

```
  border: none;
  cursor: pointer;
}
.textbox
{
  width: 200px;
  height: 40px;
  padding: 10px;
}
.textbox2
{
  width: 400px;
  height: 40px;
  padding: 10px;
}
.data
{
  align-items: center;
  text-align: center;
}
img
{
  width: 450px;
  height: 350px;
  border-radius: 50%;
}
p
{
  padding-left: 20%;
}
```

style3

```
*
{
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: 'Brush Script MT', cursive;
}
.top
```

```
{
  width: 100%;
  height: 50vh;
  background-color: green;
}
.display
{
  width: 100%;
  height: 1000vh;
  background-color: yellow;
  padding-top: 10px;
}
.cake
{
  width: 22%;
  height: 54vh;
  float: left;
  margin-left: 35px;
  margin-top: 25px;
  background-color: white;
  border-radius: 10px;
  padding: 15px;
  font-size: 14px;
}
h1
{
  text-align: center;
}
button
{
  width: 120px;
  height: 35px;
  border-radius: 30px;
  border: none;
  color: white;
  background-color: #3399ff;
  cursor: pointer;
  margin-left: 30%;
}
button:hover
```

```
{
  background-color: #ff4d88;
}
.cakeimg
{
  width: 320px;
  height: 250px;
  border-radius: 10px;
}
```

style4

```
*
{
  margin:0;
  padding:0;
  box-sizing: border-box;
  font-family: 'Brush Script MT', cursive;
}
.Big
{
  position: relative;
  width: 100%;
  height: 100vh;
  background-color: pink;
}
.BigLeft
{
  position: relative;
  float: left;
  width: 50%;
  height: 100vh;
  background-color: #f5ccff;
}
.BigRight
{
  position: relative;
  float: left;
  width: 50%;
```

```
  height: 100vh;
  background-color: #e066ff;
}
.MiniLeft
{
  position: relative;
  top: 10%;
  left: 20%;
  float: left;
  width: 80%;
  height: 80vh;
  background-color: white;
  border-radius: 30px 0px 0px 30px;
  text-align: center;
  padding-top: 10%;
}
.MiniRight
{
  position: relative;
  top: 10%;
  float: left;
  width: 80%;
  height: 80vh;
  background-color: #f5ccff;
  border-radius: 0px 30px 30px 0px;
  padding-top: 10%;
  padding-left: 10%;
}
.login
{
  width: 100px;
  height: 40px;
  color: white;
  background-color: #cc00ff;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}
.signup
{
```

```
  width: 100px;
  height: 40px;
  color: white;
  background-color: #cc00ff;
  border-radius: 5px;
  border: none;
  cursor: pointer;
}
.textbox
{
  width: 200px;
  height: 40px;
  padding: 10px;
}
.textbox2
{
  width: 400px;
  height: 40px;
  padding: 10px;
}
p
{
  padding-left: 20%;
}
.logo
{
  width: 450px;
  height: 350px;
  border-radius: 50%;
}
```

style5

```
*
{
        margin: 0;
        padding: 0;
        box-sizing: border-box;
        font-family: 'Brush Script MT', cursive;
```

```
}
body
{
		background-color: lightcyan;
}
table
{
		border: none;
		width: 100%;
}
th
{
		background-color: #e066ff;
		height: 50px;
}
td
{
		background-color: #f5ccff;
}
```

PYTHON FILES

Forms

```
from django import forms
from bakery.models import itemdetails
from bakery.models import accountdb
from bakery.models import customerorderdetails
class userForm(forms.ModelForm):
                class Meta:
                    model=itemdetails
                    fields="__all__"
class cForm(forms.ModelForm):
                class Meta:
                    model=accountdb
                    fields="__all__"
class userForm2(forms.ModelForm):
                class Meta:
                    model=customerorderdetails
                    fields="__all__"
```

Urls

```
from django.urls import path
from.import views
urlpatterns = [
                path('uploaddata',views.uploaddata),
                path('uld',views.uld),
                path('showalldata',views.showalldata),
                path('delete/<int:id>',views.delete),
                path('edit/<int:id>',views.edit),
                path('edc/<int:id>',views.edc),
                path('vieworder/<int:id>',views.vieworder),
                path('vld/<int:id>',views.vld),
                path('administrator',views.administrator),
                path('adminlog',views.adminlog),
                path('createaccount',views.createaccount),
                path('cre',views.cre),
                path('loginpage',views.loginpage),
                path('log2',views.log2),
                path('customerorder/<int:id>',views.customerorder),
                path('cto/<int:id>',views.cto),
                path('ceo',views.ceo),
                path('showcustomerorder',views.showcustomerorder),
                path('finalorder',views.finalorder),
]
```

views

```
from django.shortcuts import render,redirect
from bakery.forms import userForm
from bakery.forms import cForm
from bakery.models import itemdetails
from bakery.models import accountdb
from bakery.models import customerorderdetails
from bakery.forms import userForm2
# Create your views here.
def uploaddata(request):
                return render(request,'uploaddata.html')
def uld(request):
```

```
                    if request.method=="POST":
                        t=userForm(request.POST)
                        if t.is_valid():
                                try:
                                        t.save()
                                        return render(request,"uploaddata.html")
                                except:
                                        pass
                        return render(request,"uploaddata.html")
def showalldata(request):
                    t=itemdetails.objects.all()
                    return render(request,'showalldata.html',{'x':t})
def delete(request,id):
                    t=itemdetails.objects.get(id=id)
                    t.delete()
                    return redirect("../showalldata")
def edit(request,id):
                    t=itemdetails.objects.get(id=id)
                    return render(request,'editalldata.html',{'x':t})
def edc(request,id):
                    t=itemdetails.objects.get(id=id)
                    f=userForm(request.POST,instance=t)
                    if f.is_valid():
                        f.save()
                        return redirect("../showalldata")
                    return render(request,"editalldata.html",{'x':t})


def vieworder(request,id):
                    t=itemdetails.objects.get(id=id)
                    return render(request,'vieworder.html',{'x':t})
def vld(request,id):
                    t=itemdetails.objects.get(id=id)
                    f=userForm(request.POST,instance=t)
                    if f.is_valid():
                        f.save()
                        return redirect("../vieworder")
                    return render(request,"vieworder.html",{'x':t})
```

```
def usercreateacc(request):
                    return render(request,'createaccountuser.html')

def administrator(request):
                    return render(request,'administrator login.html')
def adminlog(request):
                    if request.method=='POST':
                        User_name=request.POST['User_name']
                        Pwd=request.POST['Pwd']
                        try:
                                if User_name=='sumon06' and Pwd=='sumon1234':
                                        return render(request,'uploaddata.html')
                                else:
                                        return render(request,'administrator login.html')
                        except:
                                return render(request,'administrator login.html')

def createaccount(request):
                    return render(request,'Login Page.html')
def cre(request):
                    if request.method=="POST":
                        t=cForm(request.POST)
                        if t.is_valid():
                                try:
                                        t.save()
                                        return render(request,'Login Page.html')
                                except:
                                        pass
                        return render(request,'Login Page.html')

def loginpage(request):
                    return render(request,'Login2.html')
def log2(request):
                    if request.method=='POST':
                        User_name=request.POST['User_name']
                        Pwd=request.POST['Pwd']
                        try:
                                p=accountdb.objects.get(User_name=User_name)
                                t=accountdb.objects.get(Pwd=Pwd)
                                if p and t is not None:
```

```
                        t=itemdetails.objects.all()
                        return render(request,'showmaindata.html',{'x':t})
                    else:
                        return render(request,'login2.html')
                except:
                    return render(request,'login2.html')

def customerorder(request,id):
                t=itemdetails.objects.get(id=id)
                return render(request,'customerorder.html',{'x':t})
def cto(request,id):
                t=itemdetails.objects.get(id=id)
                f=userForm(request.POST,instance=t)
                if f.is_valid():
                    f.save()
                    return redirect("../customerorder")
                return render(request,"customerorder.html",{'x':t})
def ceo(request):
                if request.method=="POST":
                    t=userForm2(request.POST)
                    if t.is_valid():
                        try:
                            t.save()
                            return render(request,"customerorder.html")
                        except:
                            pass
                    return render(request,"customerorder.html")
def showcustomerorder(request):
                t=customerorderdetails.objects.all()
                return render(request,'showcustomerorder.html',{'x':t})
def finalorder(request):
                return render(request,'finalorder.html')
```

IMPLEMENTATION AND TESTING

A software system test plan is a document that describes the objectives, scope, approach and focus of software testing effort. The process of preparing a test plan is a usual way to think the efforts needed to validate the acceptability of a software product. The complete document will help people outside the test group understand the "WHY" and "HOW" product validation. It should be through enough to be useful but not so through that no one outside the test group will readit.

6a. INTRODUCTION

Testing is the process of running a system with the intention of finding errors. Testing enhances the integrity of a system by detecting deviations in design and errors in the system. Testing aims at detecting error-prone areas. This helps in the prevention of errors in a system. Testing also adds value to the product by conforming to the user requirements.

The main purpose of testing is to detect errors and error-prone areas in a system. Testing must be thorough and well-planned. A partially tested system is as bad as an untested system. And the price of an untested and under-tested system is high.

The implementation is the final and important phase. It involves user-training, system testing in order to ensure successful running of the proposed system. The user tests the system and changes are made according to their needs. The testing involves the testing of the developed system using various kinds of data. While testing, errors are noted and correctness is the mode.

6b. OBJECTIVES OF TESTING:

The objective our test plan is to find and report as many bugs as possible to improve the integrity of our program. Although exhaustive testing is not possible, we will exercise a broad range of tests to achieve our goal. Our user interface to utilize these functions is designed to be user-friendly and provide easy manipulation of the tree. The application will only be used as a demonstration tool, but we would like to ensure that it could be run from a variety of platforms with little impact on performance or usability.

Process Overview

The following represents the overall flow of the testing process:

1. Identify the requirements to be tested. All test cases shall be derived using the current ProgramSpecification.

2. Identify which particular test(s) will be used to test eachmodule.

3. Review the test data and test cases to ensure that the unit has been thoroughly verified and that the test data and test cases are adequate to verify proper operation of theunit.

4. Identify the expected results for eachtest.

82

5. Document the test case configuration, test data, and expected results.

6. Perform thetest(s).

7. Document the test data, test cases, and test configuration used during the testing process. This information shall be submitted via the Unit/System Test Report (STR).

8. Successful unit testing is required before the unit is eligible for component integration/systemtesting.

9. Unsuccessful testing requires a Bug Report Form to be generated. This document shall describe the test case, the problem encountered, its possible cause, and the sequence of events that led to the problem. It shall be used as a basis for later technicalanalysis.

10. Test documents and reports shall be submitted. Any specifications to be reviewed, revised, or updated shall be handledimmediately.

6c. TEST CASES

A test case is a document that describe an input, action, or event and expected response, to determine if a feature of an application is working correctly. A test case should contain particular such as test case identifier, test condition, inputdata
Requirement expected results. The process of developing test cases can help find problems in the requirement or design of an application, since it requires completely thinking through the operation of theapplication.

TESTING STEPS

Unit Testing:

Unit testing focuses efforts on the smallest unit of software design. This is known as module testing. The modules are tested separately. The test is carried out during programming stage itself. In this step, each module is found to be working satisfactory as regards to the expected output from themodule.

Integration Testing:

Data can be lost across an interface. One module can have an adverse effect on another, sub functions, when combined, may not be linked in desired manner in major functions. Integration testing is a systematic approach for constructing the program structure, while at the same time conducting test to uncover errors associated within the interface. The objective is to take unit tested modules and builds program structure. All the modules are combined and tested as a whole.

Validation:

At the culmination of the integration testing, Software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test begin in validation testing. Validation testing can be defined in many ways, but a simple definition is that the validation succeeds when the software functions in a manner that is expected by the customer. After validation test has been conducted, one of the three possible conditions exists.

a) The function or performance characteristics confirm to specification and areaccepted.

b) A deviation from specification is uncovered and a deficiency lists is created.

c) Proposed system under consideration has been tested by using validation test and found to be working satisfactory.

Tested By:	Abinash Pramanik
Test Type	Customer Testing
Test Case Number	1
Test Case Name	Customer Identification,Order Items
Test Case Description	The Customer should create his/her account.Then enter his/her id and password so that he/she can able to go for views the cakes and order. The test case will check the application for the same since a user can only login with the correct user id, password.After selecting the required cake the customer have to place all the necessary informationthen the order is successfully confirmed.

Item(s) to be tested	
1	Verification of the user id and password ,order and customer information with the record in the database .

Specifications	
Input	**Expected Output/Result**

1) Correct User id andpassword 2) Incorrect Id orPassword 3) Select wanted cake 4) Order placed 5) Confermation information send	1) Successfullogin 2) FailureMessage 3) Successfulselection 4) Orderplaced 5) Successful

ested By:	Mittu Saha
Test Type	Admin Testing
Test Case Number	2
Test Case Name	Cake details inserting and editing, and order placed details seeing and delivering .
Test Case Description	Admin will enter the details in the cake details form and the also can delete and edit it.and also can see the orders which given by the customer.

Item(s) to be tested	
1	Required fields in the form are not empty, new items can entered edit and also delete also can see the orders.

Specifications	
Input	**Expected Output/Result**
1) admin id ,password, 2) Empty field, Invalidentry 3) Editing items 4) Deleting items 5) Add items 6) Seeing order	1) Successfulregistration 2) FailureMessage 3) Successful 4) Successful 5) Successful 6) Successful

6 d. WHITE BOX TESTING

In white box testing, the UI is bypassed. Inputs and outputs are tested directly at the code level and the results are compared against specifications. This form of testing ignores the function of the program under test and will focus only on its code and the structure of that code. Test case designers shall generate cases that not only cause each condition to take on all possible values at least once, but that cause each such condition to be executed at least once. To ensure this happens, we will be applying Branch Testing. Because the functionality of the program is relatively simple, this method will be feasible to apply.

Each function of the binary tree repository is executed independently; therefore, a program flow for each function has been derived from the code.

6e. BLACK BOX TESTING

Black box testing typically involves running through every possible input to verify that it results in the right outputs using the software as an end-user would. We have decided to perform Equivalence Partitioning and Boundary Value Analysis testing on our application.

System Testing

The goals of system testing are to detect faults that can only be exposed by testing the entire integrated system or some major part of it. Generally, system testing is mainly concerned with areas such as performance, security, validation, load/stress, and configuration sensitivity. But in our case well focus only on function validation and performance. And in both cases we will use the black-box method of testing.

6f. OUTPUT TESTING

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

User Acceptance Testing:

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for the user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wheneverrequired.

This is done in regard to the following point:

a) Input ScreenDesign
b) Output Screen Design
c) Format of reports and otheroutputs.

6g. GOAL OF TESTING

"Program testing can be used to slow the presence of bug, but never to slow their absence." If the results delivered by the system are different from the excepted ones then the system is incorrect and these bugs should be fixed.

6h. INTEGRATION TEST REPORTS

Software testing is always used in association with verification and validation. In the testing phase of this project our aim is to find the answer to following two questions.

- Whether the software matches with the specification (i.e. process base) to verify theproduct.

- Whether this software in one client what wants (i.e. product base) to validate theproduct.

  Unit testing and integration testing has been carried out to find the answer to above questions. In unit testing each individual module was test to find any unexpected behaviour if exists. Later all the module was integrated and flat file wasgenerated.

FUNCTIONAL TESTING

These are the points concerned during the stress test:

- Nominal input: character is in putted in the place of digits and the system has to flash the message "Dataerror"
- Boundary value analysis: exhaustive test cases have designed to create an output report that produces the maximum (and minimum) allowable number of table entries.

Testing Method Used

We have adopted a testing method which is a mix of both (structural) and black box (functional) testing. For modules we have adopted white box testing. Then we integrated the module into sub - systems and further into the system. These we adopted black box testing for checking the correctness of thesystem.

Requirements Validated and Verified:

- The data is getting entered properly intodatabase.
- The Screens are being loadedcorrectly
- The Various functions specified are being performed completely.

SYSTEM SECURITY MEASURES

8.a DATABASE SECURITY

System security measure is meant to be provided to make your system reliable and secured from unauthorized user may create threats to the system. So you should follow some security measures. We have used security levels in database level at systemlevel.

8.b SYSTEM SECURITY

If we talk about the system security in our proposed system we have implemented with the help of maintain the session throughout the system's use. Once a user has logged out than he/she will not be able to perform any task before signing back again.

A high level of authentic login is given to the system so this is a very tedious task to enter without authorization and authentication.

8c. LIMITATIONS:

- ✓ Since it is an online project, customers need internet connection to use it.
- ✓ People who are not familiar with computers can't use this website.
- ✓ Customer needs to register once , but login every time to use.
- ✓ After order is successful you have to login again to view or order Once again.

9. CONCLUSION

This project has been appreciated by all the users in the organization. It is easy to use, since it uses the GUI provided in the user dialog. User friendly screens are provided. The usage of software increases the efficiency, decreases the effort. It has been efficiently employed as a Site management mechanism. It has been thoroughly tested and implemented.

10. FUTURE SCOPE AND FURTHER ENHANCEMENTS

In future we would like to keep working on this project and make new additions to provide users with more advanced features and more detailed information. We have set our sights on the following additions in future:-

1. We can also add some pages who wants to make customized cake.
2. we also try to add online transaction in the system for caseless and digital transaction.
3. Add search bar to search what type of cake you want to order .
4. Trying to give easy and user friendly outlook for administrater for editing and delete or modify the data.

11. BIBLIOGRAPHY

- https://www.w3schools.com
- https://www.youtube.com

THANK YOU