JAWAHAR PUBLIC SCHOOL

A CMI EDUCATIONAL INSTITUTION

(Affiliated to CBSE NO. 930167, Delhi)

Edava,Varkala,Thiruvananthapuram Dist - 695311

INVESTIGATORY PROJECT

IN COMPUTER SCIENCE

on the topic

DIGITAL MENU SYSTEM

Name : **Hanith Harid**

Standard : **XII A**

CBSE Register No :

Year : **2024-2025**

JAWAHAR PUBLIC SCHOOL

A CMI EDUCATIONAL INSTITUTION

(Affiliated to CBSE NO. 930167, Delhi)

Edava,Varkala,Thiruvananthapuram Dist - 695311

Bonafide Certificate

This to certify that the project done in this journal is the bonafide record of ..... of standard XII Reg. No. ..... has done the project in school laboratory during the academic year 2024-2025 for partial fulfilment of practical syllabus of Senior School Certificate Examination conducted by CBSE.

....

Date *Teacher in charge*

School Seal

....

External Examiner *Principal*

ACKNOWLEDGEMENT

It would be my utmost pleasure to express my sincere thanks to my Computer Science teacher Miss Priyadarsini S in providing a helping hand in this project. Her unflagging patience, creativity and immense knowledge that she shared with us have proved highly beneficial to us and have made our project both possible and successful.

I would also like to thank our Principal Father Joshy Mayamparambil CMI for his guidance, motivation and giving all the support.

This is to certify that Anu P of class 12th has prepared the report on the project entitled “Digital Menu System”

Table of Content

INTRODUCTION

This program is a comprehensive restaurant management software designed to streamline various operations within a restaurant. It ensures smooth menu management, order processing, user administration, and sales tracking, enhancing overall efficiency and service quality.

Admin Module:

Allows administrators to:

- Sign in securely.
- Add, update, and remove menu items.
- View and manage orders.
- Calculate total sales.
- Handle user administration (add/remove users).

Chef Module:

Enables kitchen staff to:

- Sign in.
- View and update order statuses.
- Access past orders.
- View the current menu.

Customer Module:

Facilitates the ordering process by allowing customers to:

- Place orders from the available menu.
- View the menu and check item availability.

The software's dynamic Menu Display function shows real-time updates of item availability, ensuring that both staff and customers are always informed about the current state of the menu. This comprehensive solution enhances the efficiency of restaurant operations, improves coordination among staff, and provides a better dining experience for customers.

OBJECTIVES OF THE PROJECT

The objective of this project is to let the students apply the programming knowledge into a real- world situation/problem and exposed the students how programming skills helps in developing a good software.

- Write programs utilizing modern software tools.
- Apply object oriented programming principles effectively when developing small to medium sized projects.
- Write effective procedural code to solve small to medium sized problems.
- Students will demonstrate a breadth of knowledge in computer science, as exemplified in the areas of systems, theory and software development.
- Students will demonstrate ability to conduct a research or applied Computer Science project, requiring writing and presentation skills which exemplify scholarly style in computer science.

PROPOSED SYSTEM

Today one cannot afford to rely on the fallible human beings of be really wants to stand against today's merciless competition where not to wise saying "to err is human" no longer valid, it's outdated to rationalize your mistake. So, to keep pace with time, to bring about the best result without malfunctioning and greater efficiency so to replace the unending heaps of flies with a much sophisticated hard disk of the computer.

One has to use the data management software. Software has been an ascent in atomization various organisations. Many software products working are now in markets, which have helped in making the organizations work easier and efficiently. Data management initially had to maintain a lot of ledgers and a lot of paper work has to be done but now software product on this organization has made their work faster and easier. Now only this software has to be loaded on the computer and work can be done.

This prevents a lot of time and money. The work becomes fully automated and any information regarding the organization can be obtained by clicking the button. Moreover, now it's an age of computers of and automating such an organization gives the better look.

SYSTEM DEVELOPMENT LIFE CYCLE (SDLC)

The systems development life cycle is a project management technique that divides complex projects into smaller, more easily managed segments or phases. Segmenting projects allows managers to verify the successful completion of project phases before allocating resources to subsequent phases.

Software development projects typically include initiation, planning, design, development, testing, implementation, and maintenance phases. However, the phases may be divided differently depending on the organization involved.

For example, initial project activities might be designated as request, requirements-definition, and planning phases, or initiation, conceptdevelopment, and planning phases. End users of the system under development should be involved in reviewing the output of each phase to ensure the system is being built to deliver the needed functionality.

PHASES OF SYSTEM DEVELOPMENT LIFE CYCLE

INITIATION PHASE

The Initiation Phase begins when a business sponsor identifies a need or an opportunity.
The purpose of the Initiation Phase is to:

- Identify and validate an opportunity to improve business accomplishments of the organization or a deficiency related to a business need.

- Identify significant assumptions and constraints on solutions to that need.

- Recommend the exploration of alternative concepts and methods to satisfy the need including questioning the need for technology, i.e., will a change in the business process offer a solution?

- Assure executive business and executive technical sponsorship. The Sponsor designates a Project Manager and the business need is documented in a Concept Proposal. The Concept Proposal includes information about the business process and the relationship to the Agency/Organization.

- Infrastructure and the Strategic Plan. A successful Concept Proposal results in a Project Management

- Charter which outlines the authority of the project manager to begin the project

Careful oversight is required to ensure projects support strategic business objectives and resources are effectively implemented into an organization's enterprise architecture. The initiation phase begins when an opportunity to add, improve, or correct a system is identified and formally requested through the presentation of a business case. The business case should, at a minimum, describe a proposal's purpose, identify expected benefits, and explain how the proposed system supports one of the organization's business strategies. The business case should also identify alternative solutions and detail as many informational, functional, and network requirements as possible.

SYSTEM CONCEPT DEVELOPMENT PHASE

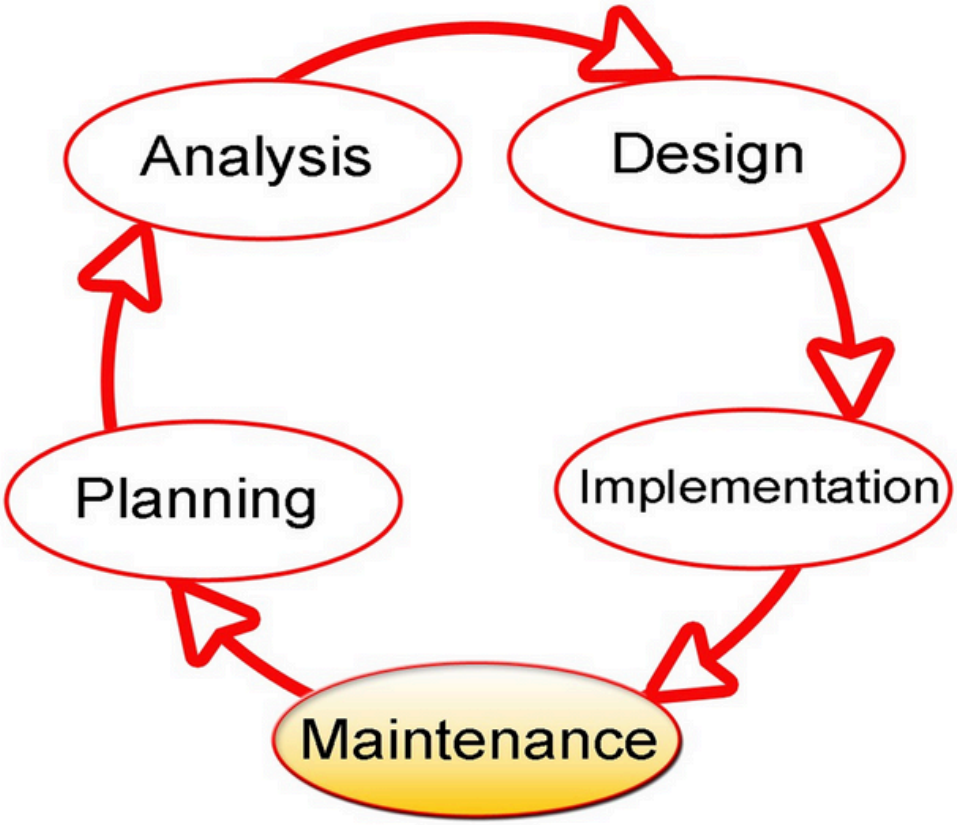
The System Concept Development Phase begins after a business need or opportunity is validated by the Agency/Organization Program Leadership and the Agency/Organization CIO.

The purpose of the System Concept Development Phase is to:

- Determine the feasibility and appropriateness of the alternatives.
- Identify system interfaces.
- Identify basic functional and data requirements to satisfy the business need
- Establish system boundaries; identify goals, objectives, critical success factors, and performance measures.
- Evaluate costs and benefits of alternative approaches to satisfy the basic functional requirements
- Assess project risks
- Identify and initiate risk mitigation actions, and Develop high-level technical architecture, process models, data models, and a concept of operations. This phase explores potential technical solutions within the context of the business need.
- It may include several trade-off decisions such as the decision to use COTS software products as opposed to developing custom software or reusing software components, or the decision to use an incremental delivery versus a complete, onetime deployment.

- Construction of executable prototypes is encouraged to evaluate technology to support the business process.The System Boundary Document serves as an important reference document to support the Information Technology Project Request (ITPR) process.
- The ITPR must be approved by the State CIO before the project can move forward.

PICTORIAL REPRESENTATION OF SDLC



PLANNING PHASE

The planning phase is the most critical step in completing development, acquisition, and maintenance projects. Careful planning, particularly in the early stages of a project, is necessary to coordinate activities and manage project risks effectively. The depth and formality of project plans should be commensurate with the characteristics and risks of a given project. Project plans refine the information gathered during the initiation phase by further identifying the specific activities and resources required to complete a project.

A critical part of a project manager's job is to coordinate discussions between user, audit, security, design, development, and network personnel to identify and document as many functional, security, and network requirements as possible. During this phase, a plan is developed that documents the approach to be used and includes a discussion of methods, tools, tasks, resources, project schedules, and user input. Personnel assignments, costs, project schedule, and target dates are established.

A Project Management Plan is created with components related to acquisition planning, configuration management planning, quality assurance planning, concept of operations, system security, verification and validation, and systems engineering management planning.

REQUIREMENTS ANALYSIS PHASE

This phase formally defines the detailed functional user requirements using high-level requirements identified in the Initiation, System Concept, and Planning phases. It also delineates the requirements in terms of data, system performance, security, and maintainability requirements for the system. The requirements are defined in this phase to a level of detail sufficient for systems design to proceed. They need to be measurable, testable, and relate to the business need or opportunity identified in the Initiation Phase. The requirements that will be used to determine acceptance of the system are captured in the Test and Evaluation Master Plan.

The purposes of this phase are to:

- Further define and refine the functional and data requirements and document them in the Requirements Document.

- Complete business process reengineering of the functions to be supported (i.e., verify what information drives the business process, what information is generated, who generates it, where does the information go, and who processes it).

- Develop detailed data and process models (system inputs, outputs, and the process.

- Develop the test and evaluation requirements that will be used to determine acceptable system performance.

DESIGN PHASE

The design phase involves converting the informational, functional, and network requirements identified during the initiation and planning phases into unified design specifications that developers use to scriptprograms during the development phase. Program designs are constructed in various ways. Using a top-down approach, designers first identify and link majorprogram components and interfaces, then expand design layouts as they identify and link smaller subsystems and connections. Using a bottom-up approach, designers first identify and link minor program components and interfaces, then expand design layouts as they identify and link larger systems and connections. Contemporary design techniques often use prototyping tools that build mock-up designs of items such as application screens, database layouts, and system architectures. End users, designers, developers, database managers, and network administrators should review and refine the prototyped designs in an iterative process until they agree on an acceptable design. Audit, security, and quality assurance personnel should be involved in the review and approval process. During this phase, the system is designed to satisfy the functional requirements identified in the previous phase. Since problems in the design phase could be very expensive to solve in the later stage of the software development, a variety of elements are considered in the design to mitigate risk.

These include:

- Identifying potential risks and defining mitigating design features.
- Performing a security risk assessment.
- Developing a conversion plan to migrate current data to the new system.
- Determining the operating environment.
- Defining major subsystems and their inputs and outputs.
- Allocating processes to resources.
- Preparing detailed logic specifications for each software module. The result is a draft System Design Document which captures the preliminary design for the system

- Everything requiring user input or approval is documented and reviewed by the user. Once these documents have been approved by the Agency CIO and Business Sponsor, the final System Design Document is created to serve as the Critical/Detailed Design for the system.

- This document receives a rigorous review byAgency technical and functional representatives to ensure that it satisfies the business requirements. Concurrent with the development of the system design, the Agency Project Manager begins development of the Implementation Plan, Operations and Maintenance Manual, and the Training Plan.

DEVELOPMENT PHASE

The development phase involves converting design specifications into executable programs. Effective development standards include requirements that programmers and other project participants discuss design specifications before programming begins. The procedures help ensure programmers clearly understand program designs and functional requirements. Programmers use various techniques to develop computer programs. The large transaction oriented programs associated with financial institutions have traditionally been developed using procedural programming techniques. Procedural programming involves the line-by-line scripting of logical instructions that are combined to form a program.Effective completion of the previous stages is a key factor in the success of the Development phase. The Development phase consists of:

- Translating the detailed requirements and design into system components.
- Testing individual elements (units) for usability.
- Preparing for integration and testing of the IT system.

INTEGRATION AND TEST PHASE

Subsystem integration, system, security, and user acceptance testing is conducted during the integration and test phase. The user, with those responsible for quality assurance, validates that the functional requirements, as defined in the functional requirements document, are satisfied by the developed or modified system. OIT Security staff assess the system security and issue a security certification and accreditation prior to installation/implementation.

Multiple levels of testing are performed, including:

- Testing at the development facility by the contractor and possibly supported by end users.

- Testing as a deployed system with end users working together with contract personnel.

- Operational testing by the end user alone performing all functions. Requirements are traced throughout testing,a final Independent Verification & Validation evaluation is performed and all documentation is reviewedand accepted prior to acceptance of the system.

IMPLEMENTATION PHASE

This phase is initiated after the system has been tested and accepted by the user. In this phase, the system is installed to support the intended business functions. System performance is compared to performance objectives established during the planning phase. Implementation includes user notification, user training, installation of hardware, installation of software onto production computers, and integration of the system into daily work processes. This phase continues until the system is operating in production in accordance with the defined userrequirements.

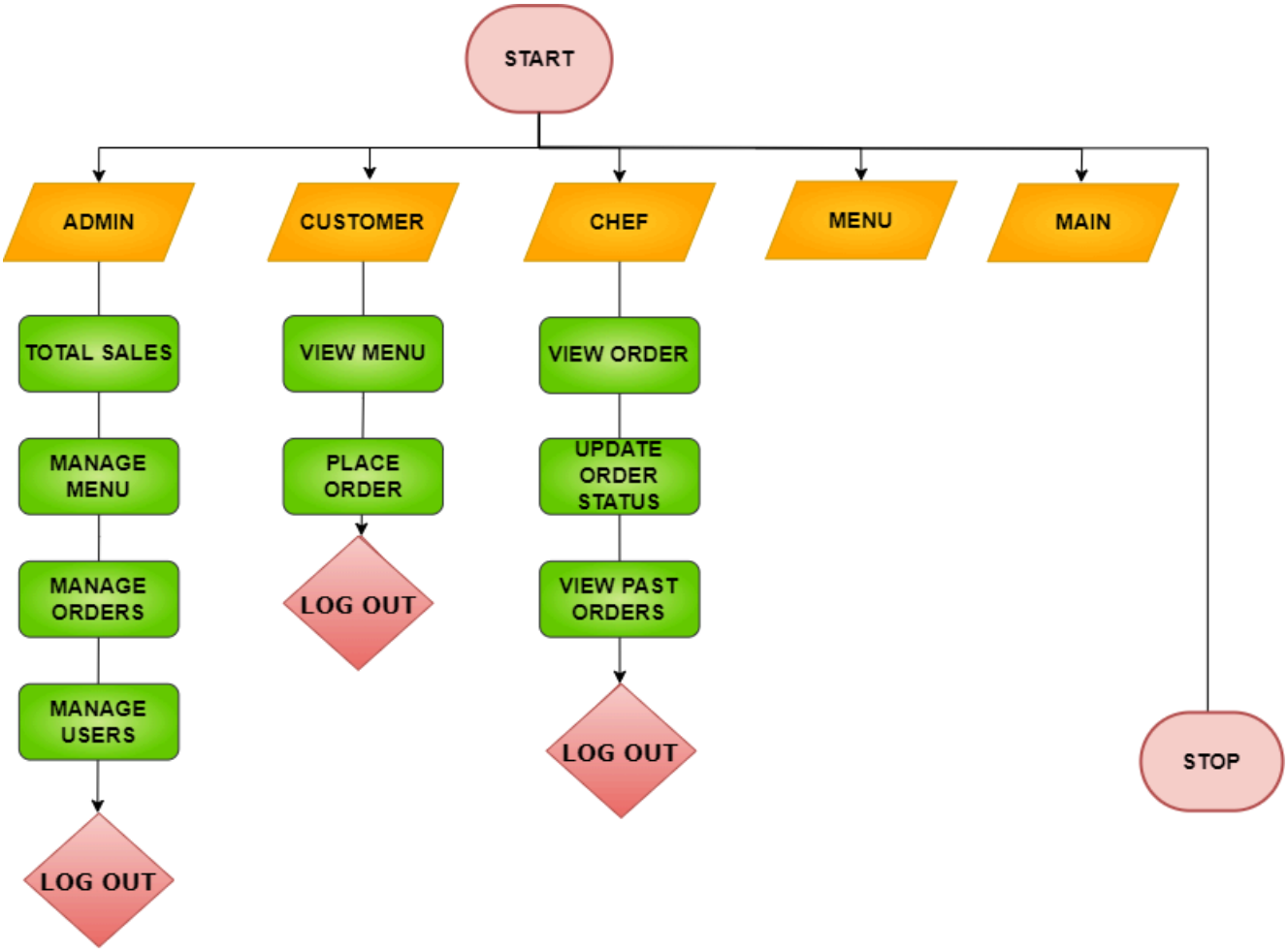
OPERATIONS AND MAINTENANCE PHASE

The system operation is ongoing. The system is monitored for continued performance in accordance with user requirements and needed system modifications are incorporated. Operations continue as long as the system can be effectively adapted to respond to the organization's needs. When modifications or changes are identified, the system may reenter the planning phase.

The purpose of this phase is to:

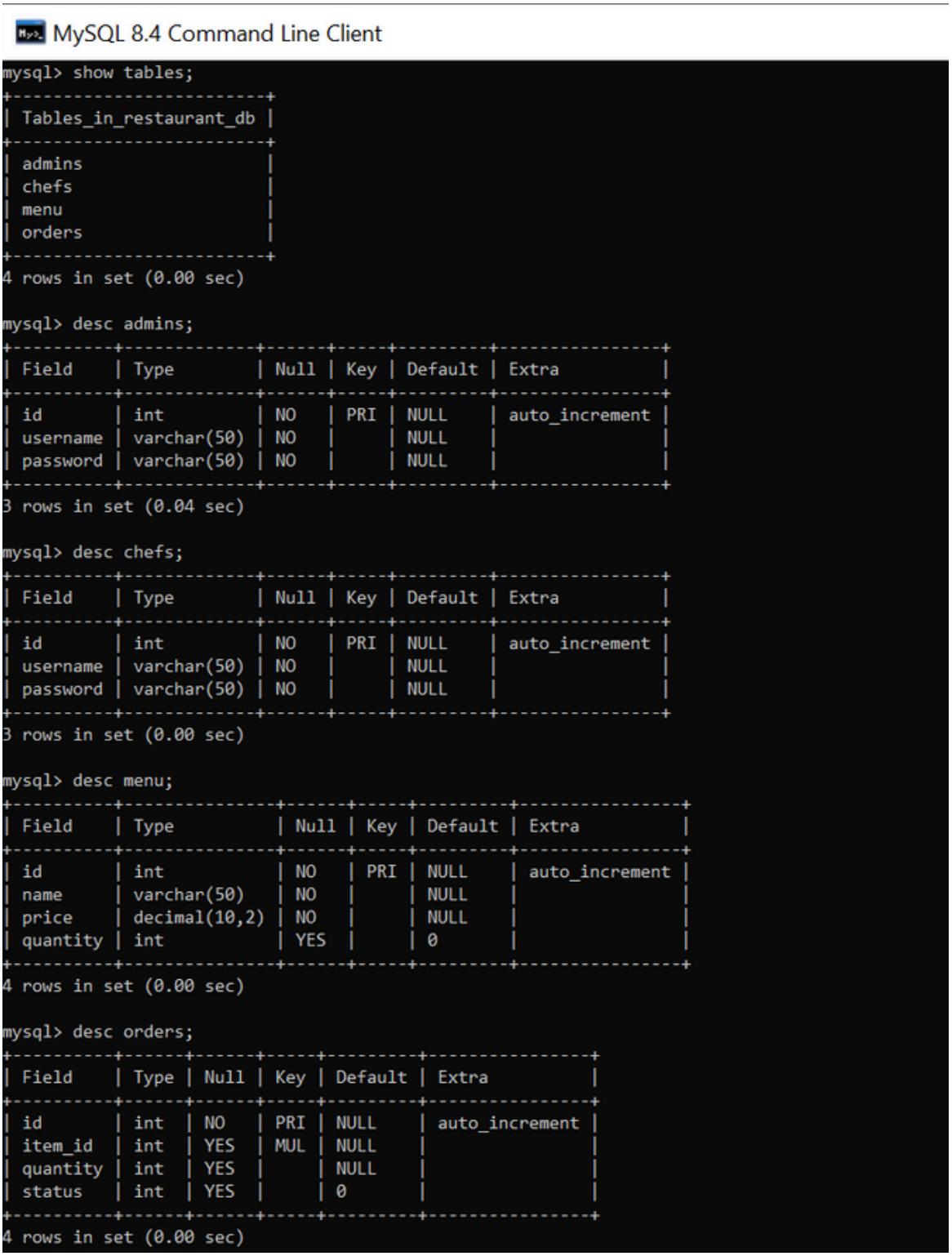
- Operate, maintain, and enhance the system.
- Certify that the system can process sensitive information.
- Conduct periodic assessments of the system to ensure the functional requirements continue to be satisfied.
- Determine when the system needs to be modernized, replaced, or retired

FLOW CHART


START
ADMIN
CUSTOMER
CHEF
MENU
MAIN
TOTAL SALES
MANAGE MENU
MANAGE ORDERS
MANAGE USERS
LOG OUT
VIEW MENU
PLACE ORDER
LOG OUT
VIEW ORDER
UPDATE ORDER STATUS
VIEW PAST ORDERS
LOG OUT
STOP

SOURCE CODE SCREEN

DBMS: MYSQL
Host: local host
User: root
Password: root
Database: Food Table Structure:



PYTHON CODE

1-ADMIN MODULE

```
import mysql.connector from
menu import view_menu

# Connection to sql
conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="tiger",
    database="restaurant_db")
cursor = conn.cursor()

# Function for admin login
def admin_login():
    username = input("Enter admin username: ")
    password = input("Enter admin password: ")
    cursor.execute("SELECT * FROM admins WHERE
username = %s AND password = %s", (username,
password))
    admin = cursor.fetchone()
    if admin:
        print("Login successful!")
        return True
    else:
        print("Invalid credentials. Please try
again.")
        return False
```

```
# Function to calculate total sales
def total_sales():
    cursor.execute("""
        SELECT SUM(m.price * o.quantity)
        FROM orders o JOIN menu m ON
        o.item_id = m.id WHERE o.status =
        2
    """) total = cursor.fetchone()[0] or
    0 print(f'Total sales: ₹
    {total:.2f}')

# Function to manage the menu
def manage_menu():
    while True:
        print("\n>>> Manage Menu <<<")
        print("1. Add item to menu")
        print("2. Remove item from menu")
        print("3. Update item quantity")
        print("4. View menu")
        print("5. Back")
        choice = input("Enter your choice: ")
        if choice == '1':
          add_item_to_menu()
              view_menu()
         elif choice == '2':
              view_menu()
```

```
        remove_item_from_menu()
        view_menu() elif choice ==
        '3':
           update_item_quantity()
         view_menu() elif choice ==
            '4': view_menu() elif
               choice == '5':


            break
        else:
            print("Invalid choice. Please try again.")

# Function to add an item to the menu
def add_item_to_menu():
    name = input("Enter item name: ")
    price = float(input("Enter item price: "))
    cursor.execute("INSERT INTO menu (name, price,
quantity) VALUES (%s, %s, %s)", (name, price, 0))
    conn.commit()

# Function to remove an item from the menu
def remove_item_from_menu():
    item_id = int(input("Enter item ID to remove: "))
    cursor.execute("DELETE FROM orders WHERE item_id =
%s", (item_id,))
    conn.commit()
```

```
    cursor.execute("DELETE FROM menu WHERE id = %s",
(item_id,))
    conn.commit()
    print(f"Item {item_id} and its related orders have
been deleted successfully.")


# Function to update the quantity of an item in the
menu
def update_item_quantity():
    item_id = int(input("Enter item ID to update
quantity: "))
    new_quantity = int(input("Enter new quantity: "))
    cursor.execute("UPDATE menu SET quantity = %s
WHERE id = %s", (new_quantity, item_id))
    conn.commit()
    print(f"Updated item {item_id} quantity to
{new_quantity}.")


# Function to manage orders
def manage_orders():
    while True:
        print("1. View all orders")
        print("2. Update order status")
        print("3. Back")
        choice = input("Enter your choice: ")
        if choice == '1':
            view_all_orders()
        elif choice == '2':
```

```
            order_id = int(input("Enter order ID to
update: "))
            status = int(input("Enter new status (1
for being prepared, 2 for ready): "))
            update_order_status(order_id, status)
        elif choice == '3':
            break
        else:
            print("Invalid choice. Please try again.")

# Function to view all orders
def view_all_orders():
cursor.execute("""
        SELECT o.id, m.name, o.quantity, o.status
        FROM orders o
        JOIN menu m ON o.item_id = m.id
        ORDER BY o.id
    """)
    orders = cursor.fetchall()
    print("\n--- All Orders ---")
    print(f"{'Order ID':<10} {'Item Name':<20}
{'Quantity':<10} {'Status':<10}")
    print("-" * 60)
    for order in orders:
        print(f"{order[0]:<10} {order[1]:<20}
{order[2]:<10} {order[3]:<10}")
```

```
# Function to update the status of an order
def update_order_status(order_id, status):
    cursor.execute("UPDATE orders SET status=%s WHERE
id=%s", (status, order_id))
    conn.commit()

# Function to manage users
def manage_users():
while True:
        print("\n--- Manage Users ---")
        print("1. Add User")
        print("2. View Usernames")
        print("3. Delete User")
        print("4. Back")
        choice = input("Enter your choice: ")
        if choice == '1':
            add_user()
            view_usernames()
        elif choice == '2':
            view_usernames()
        elif choice == '3':
            delete_user()
            view_usernames()
        elif choice == '4':
            break
        else:
```

```
            print("Invalid choice. Please try again.")

# Function to add a new user
def add_user():
while True:
        print("\n--- Add User ---")
        print("1. Add Admin")
        print("2. Add Chef")
        print("3. Back")
        choice = input("Enter your choice: ")
        if choice == '1':
            add_admin()
        elif choice == '2':
            add_chef()
        elif choice == '3':
            break
        else:
            print("Invalid choice. Please try again.")

# Function to add a new admin
def add_admin():
username = input("Enter new admin username: ")
password = input("Enter new admin password: ")
cursor.execute("INSERT INTO admins (username,
password) VALUES (%s, %s)", (username, password))
    conn.commit()
```

```
    print(f"Admin {username} added successfully.")

# Function to add a new chef
def add_chef():
    username = input("Enter new chef username: ")
    password = input("Enter new chef password: ")
    cursor.execute("INSERT INTO chefs (username,
password) VALUES (%s, %s)", (username, password))
    conn.commit()
    print(f"Chef {username} added successfully.")

# Function to view usernames of admins and chefs
def view_usernames():
    print("\n--- Admins ---")
    cursor.execute("SELECT username FROM admins")
    admins = cursor.fetchall()
    for admin in admins:
        print(admin[0])
    print("\n--- Chefs ---")
    cursor.execute("SELECT username FROM chefs")
    chefs = cursor.fetchall()
    for chef in chefs:
    print(chef[0])

# Function to delete a user
def delete_user():
```

```
    while True:
        print("\n--- Delete User ---")
        print("1. Delete Admin") print("2.
        Delete Chef") print("3. Back") choice
        = input("Enter your choice: ") if
        choice == '1':


            delete_admin()
        elif choice == '2':
            delete_chef()
        elif choice == '3':
            break
        else:
            print("Invalid choice. Please try again.")


# Function to delete an admin
def delete_admin():
    username = input("Enter admin username to delete:
")
    cursor.execute("DELETE FROM admins WHERE username
= %s", (username,))
    conn.commit()
    print(f"Admin {username} deleted successfully.")


# Function to delete a chef
def delete_chef():
```

```
    username = input("Enter chef username to delete: ")
    cursor.execute("DELETE FROM chefs WHERE username = %s", (username,))
conn.commit()
print(f"Chef {username} deleted successfully.")


# Function to close the connection to the database
def close():
    conn.close()
```

2-CUSTOMER MODULE

```
import mysql.connector from menu import view_menu #
Import view_menu from
menu.py


# Connect to the database
conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="tiger",
    database="restaurant_db")
cursor = conn.cursor()


# Function to place an order
def place_order():
    order_items = []
    total_price = 0.0
    while True:
        item_id = input("Enter item ID to order (or
press [ENTER] to finish): ")
        if item_id.lower() == '':
            break
        quantity = int(input("Enter quantity: "))
        cursor.execute("SELECT price FROM menu WHERE
id = %s", (item_id,))
        price = float(cursor.fetchone()[0])
        order_items.append((item_id, quantity, price *
quantity))
```

```
        total_price += price * quantity print(f"Added
        {quantity} x item ID {item_id}
 to order. Current total: ₹{total_price:.2f}")
     confirm = input("Confirm order? (yes/no):
               ").lower()
    if confirm == 'yes':
        for item in order_items:
            cursor.execute("INSERT INTO orders
(item_id, quantity, status) VALUES (%s, %s, %s)",
(item[0], item[1], 0))
        conn.commit()
        print("Order placed successfully.")
    else:
        print("Order cancelled.")


# Function to close the connection to the database
def close():
    conn.close()
```

3-CHEF MODULE

```
import mysql.connector from
menu import view_menu

# Connect to the database
conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="tiger",
    database="restaurant_db")
cursor = conn.cursor()
# Function for chef login
def chef_login():
    username = input("Enter chef username: ")
    password = input("Enter chef password: ")
    cursor.execute("SELECT * FROM chefs WHERE username
= %s AND password = %s", (username, password))
    chef = cursor.fetchone()
    if chef:
        print("Login successful!")
        return True
    else:
        print("Invalid credentials. Please try
again.")
        return False
# Function to display the chef menu
def chef_menu():
```

```
    while True:
        print("\nChef Menu:") print("1. View
        Orders") print("2. Update Order
        Status") print("3. View Menu")
        print("4. View Past Orders") print("5.
        Logout") choice = input("Enter your
        choice: ") if choice == '1':
        view_orders()



        elif choice == '2':
            view_all_orders() order_id =
            int(input("Enter order ID to
update: "))
            status = int(input("Enter new status (1
for being prepared, 2 for ready): "))
            update_order_status(order_id, status)
        elif choice == '3':
            view_menu()
        elif choice == '4':
            view_past_orders()
        elif choice == '5':
            break
        else:
            print("Invalid choice. Please try again.")
```

```
# Function to view orders that are not ready
def view_orders():
    cursor.execute("""
        SELECT o.id, m.name, o.quantity, o.status
        FROM orders o JOIN menu m ON o.item_id =
        m.id WHERE o.status != 2 ORDER BY o.id



    """) orders = cursor.fetchall() print("\n<<<
    Orders >>>") print("{:<10} {:<20} {:<10} {:
    <10}".format("Order

ID", "Item Name", "Quantity", "Status"))
    print("-" * 60)
    for order in orders:
        print("{:<10} {:<20} {:<10}
{:<10}".format(order[0], order[1], order[2],
order[3]))


# Function to view all orders
def view_all_orders():
cursor.execute("""
        SELECT o.id, m.name, o.quantity, o.status
        FROM orders o
        JOIN menu m ON o.item_id = m.id
        ORDER BY o.id
    """)
```

```
    orders = cursor.fetchall() print("\n<<< All Orders
    >>>") print("{:<10} {:<20} {:<10} {:
    <10}".format("Order
ID", "Item Name", "Quantity", "Status"))
    print("-" * 60)
    for order in orders:
        print("{:<10} {:<20} {:<10}
{:<10}".format(order[0], order[1], order[2],
order[3]))


# Function to view past orders that are ready
def view_past_orders():
    cursor.execute("""
        SELECT o.id, m.name, o.quantity, o.status
        FROM orders o
        JOIN menu m ON o.item_id = m.id
        WHERE o.status = 2
        ORDER BY o.id
    """)
    past_orders = cursor.fetchall()
    print("\n<<< Past Orders >>>")
    print("{:<10} {:<20} {:<10} {:<10}".format("Order
ID", "Item Name", "Quantity", "Status"))
    print("-" * 60)
    for order in past_orders:
        print("{:<10} {:<20} {:<10}
{:<10}".format(order[0], order[1], order[2],
order[3]))
```

```
# Function to update order status def
update_order_status(order_id, status):
    cursor.execute("UPDATE orders SET status = %s
WHERE id = %s", (status, order_id))
    conn.commit()

# Function to close the connection to the database
def close():
    conn.close()
```

4-MAIN MODULE

```
import mysql.connector from admin import admin_login,
total_sales, manage_menu,
manage_orders, manage_users, close as admin_close
from chef import chef_login, chef_menu, close as
chef_close
from customer import place_order, close as
customer_close
from menu import view_menu
# Connect to the database
conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="tiger",
    database="restaurant_db")
cursor = conn.cursor()
def main_menu():
    while True:
        print("\n1. Place Order")
        print("2. View Menu")
        print("3. Chef Login")
        print("4. Admin Login")
        print("5. Exit")
        choice = input("Enter your choice: ")
        if choice == '1':
            view_menu()
            place_order()
        elif choice == '2':
```

```
view_menu() elif
 choice == '3':
  if chef_login():
   chef_menu() elif
    choice == '4':
  if admin_login():
      while True:
          print("\nAdmin     Menu:")
          print("1.   Total   Sales")
          print("2.   Manage   Menu")
          print("3.  Manage  Orders")
          print("4.   Manage  Users")
          print("5.          Logout")
          admin_choice = input("Enter choice:")
          if admin_choice == '1':
              total_sales()
          elif admin_choice == '2':
              manage_menu()
          elif admin_choice == '3':
              manage_orders()
          elif admin_choice == '4':
              manage_users()
          elif admin_choice == '5':
              break
          else:
             print("Invalid choice. Please try
                    again.")
```

```
            else:
                print("Access Denied")
        elif choice == '4':
            if chef_login():
                chef_menu()
            else:
                print("Access Denied")
        elif choice == '5':
            print('Thankyou !')
            break
        else:
            print("Invalid choice. Please try again.")
    admin_close()
    chef_close()
    customer_close()
print("\n" + "*" * 60) print(f"*{'WELCOME TO THE
PROJECT OF DIGITAL MENU
SYSTEM':^58}*")
print("*" * 60)
print(f"*{'Developed by:':^58}*")
print("*" + "-" * 58 + "*")
print(f"*{'Hanith Harid':^58}*")
print(f"*{'Anu P':^58}*")
print(f"*{'Akash Sudersh':^58}*")
print("*" * 60 + "\n")
main_menu()
```

5-MENU MODULE

```
import mysql.connector


def view_menu():
    conn = mysql.connector.connect(
        host="localhost",
        user="root",
        password="tiger",
        database="restaurant_db")
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM menu")
    menu = cursor.fetchall()
    conn.close()


    print("\n------------------------- Menu ----------
----------------")
    print(f"{'ID':<5} {'Name':<20} {'Price (₹)':>10}
{'Stock Status':<15}")
    print("-" * 60)
    for item in menu:
        stock_status = 'In Stock' if item[3] > 0 else
'Out of Stock'
        print(f"{item[0]:<5} {item[1]:<20}
{item[2]:>10.2f} {stock_status:<15}")
```

OUTPUT

```
************************************************************
*      WELCOME TO THE PROJECT OF DIGITAL MENU SYSTEM       *
************************************************************
*                     Developed by:                        *
*----------------------------------------------------------*
*                      Hanith Harid                        *
*                         Anu P                            *
*                     Akash Sudersh                        *
************************************************************


1. Place Order
2. View Menu
3. Chef Login
4. Admin Login
5. Exit
Enter your choice: 1

------------------------ Menu --------------------------
ID    Name                 Price (₹) Stock Status
------------------------------------------------------------
1     Pizza                   180.00 In Stock
2     Burger                   80.00 In Stock
3     Water                    10.00 In Stock
Enter item ID to order (or press [ENTER] to finish): 1
Enter quantity: 1
Added 1 x item ID 1 to order. Current total: ₹180.00
Enter item ID to order (or press [ENTER] to finish): 3
Enter quantity: 1
Added 1 x item ID 1 to order. Current total: ₹180.00
Enter item ID to order (or press [ENTER] to finish): 3
Enter quantity: 1
Added 1 x item ID 3 to order. Current total: ₹190.00
Enter item ID to order (or press [ENTER] to finish):
Confirm order? (yes/no): yes
Order placed successfully.

1. Place Order
2. View Menu
3. Chef Login
4. Admin Login
5. Exit
Enter your choice: 1

------------------------ Menu --------------------------
ID    Name                 Price (₹) Stock Status
------------------------------------------------------------
1     Pizza                   180.00 In Stock
2     Burger                   80.00 In Stock
3     Water                    10.00 In Stock
Enter item ID to order (or press [ENTER] to finish): 3
Enter quantity: 1
Added 1 x item ID 3 to order. Current total: ₹10.00
Enter item ID to order (or press [ENTER] to finish):
Confirm order? (yes/no): no
Order cancelled.

1. Place Order
2. View Menu
3. Chef Login
4. Admin Login
5. Exit
Enter your choice: 2

------------------------ Menu --------------------------
ID    Name                 Price (₹) Stock Status
------------------------------------------------------------
1     Pizza                   180.00 In Stock
2     Burger                   80.00 In Stock
3     Water                    10.00 In Stock
```

```
1. Place Order
2. View Menu
3. Chef Login
4. Admin Login
5. Exit
Enter your choice: 3
Enter chef username: chef
Enter chef password: password
Login successful!

Chef Menu:
1. View Orders
2. Update Order Status
3. View Menu
4. View Past Orders
5. Logout
Enter your choice: 1

<<< Orders >>>
Order ID   Item Name            Quantity   Status
------------------------------------------------------------
2          Pizza                1          0
3          Pizza                1          0
4          Water                1          0

Chef Menu:
1. View Orders
2. Update Order Status
3. View Menu
4. View Past Orders
```

```
<<< All Orders >>>
Order ID   Item Name            Quantity   Status
------------------------------------------------------------
1          Burger               2          2
2          Pizza                1          0
3          Pizza                1          0
4          Water                1          0
Enter order ID to update: 2
Enter new status (1 for being prepared, 2 for ready): 1

Chef Menu:
1. View Orders
2. Update Order Status
3. View Menu
4. View Past Orders
5. Logout
Enter your choice: 2

<<< All Orders >>>
Order ID   Item Name            Quantity   Status
------------------------------------------------------------
1          Burger               2          2
2          Pizza                1          1
3          Pizza                1          0
4          Water                1          0
Enter order ID to update: 2
Enter new status (1 for being prepared, 2 for ready): 2

Chef Menu:
1. View Orders
2. Update Order Status
3. View Menu
4. View Past Orders
5. Logout
Enter your choice: 3

------------------------- Menu -------------------------
ID    Name                 Price (₹) Stock Status
------------------------------------------------------------
1     Pizza                180.00 In Stock
2     Burger                80.00 In Stock
3     Water                 10.00 In Stock

Chef Menu:
1. View Orders
2. Update Order Status
3. View Menu
4. View Past Orders
5. Logout
Enter your choice: 4

<<< Past Orders >>>
Order ID   Item Name            Quantity   Status
------------------------------------------------------------
1          Burger               2          2
2          Pizza                1          2
```

```
Chef Menu:
1. View Orders
2. Update Order Status
3. View Menu
4. View Past Orders
5. Logout
Enter your choice: 5

1. Place Order
2. View Menu
3. Chef Login
4. Admin Login
5. Exit
Enter your choice: 4
Enter admin username: admin
Enter admin password: password
Login successful!

Admin Menu:
1. Total Sales
2. Manage Menu
3. Manage Orders
4. Manage Users
5. Logout
Enter choice: 1
Total sales: ₹340.00

Admin Menu:
1. Total Sales
2. Manage Menu
3. Manage Orders
4. Manage Users
5. Logout
Enter choice: 2

>>> Manage Menu <<<
1. Add item to menu
2. Remove item from menu
3. Update item quantity
4. View menu
5. Back
Enter your choice: 1
Enter item name: Orange Juice
Enter item price: 20

------------------------- Menu -------------------------
ID    Name                  Price (₹) Stock Status
--------------------------------------------------------------
1     Pizza                   180.00 In Stock
2     Burger                   80.00 In Stock
3     Water                    10.00 In Stock
5     Orange Juice             20.00 Out of Stock

>>> Manage Menu <<<
1. Add item to menu
2. Remove item from menu
3. Update item quantity
4. View menu
5. Back
Enter your choice: 3
Enter item ID to update quantity: 5
Enter new quantity: 10
Updated item 5 quantity to 10.

------------------------- Menu -------------------------
ID    Name                  Price (₹) Stock Status
--------------------------------------------------------------
1     Pizza                   180.00 In Stock
2     Burger                   80.00 In Stock
3     Water                    10.00 In Stock
5     Orange Juice             20.00 In Stock

>>> Manage Menu <<<
1. Add item to menu
2. Remove item from menu
3. Update item quantity
4. View menu
5. Back
Enter your choice: 2

------------------------- Menu -------------------------
ID    Name                  Price (₹) Stock Status
--------------------------------------------------------------
1     Pizza                   180.00 In Stock
2     Burger                   80.00 In Stock
3     Water                    10.00 In Stock
5     Orange Juice             20.00 In Stock
Enter item ID to remove: 5
Item 5 and its related orders have been deleted successfully.

------------------------- Menu -------------------------
ID    Name                  Price (₹) Stock Status
--------------------------------------------------------------
1     Pizza                   180.00 In Stock
2     Burger                   80.00 In Stock
3     Water                    10.00 In Stock
```

```
>>> Manage Menu <<<
1. Add item to menu
2. Remove item from menu
3. Update item quantity
4. View menu
5. Back
Enter your choice: 4

------------------------ Menu -------------------------
ID    Name                 Price (₹) Stock Status
------------------------------------------------------------
1     Pizza                   180.00 In Stock
2     Burger                   80.00 In Stock
3     Water                    10.00 In Stock

>>> Manage Menu <<<
1. Add item to menu
2. Remove item from menu
3. Update item quantity
4. View menu
5. Back
Enter your choice: 5
```

```
Admin Menu:
1. Total Sales
2. Manage Menu
3. Manage Orders
4. Manage Users
5. Logout
Enter choice: 3
1. View all orders
2. Update order status
3. Back
Enter your choice: 1

--- All Orders ---
Order ID   Item Name            Quantity   Status
------------------------------------------------------------
1          Burger               2          2
2          Pizza                1          2
3          Pizza                1          0
4          Water                1          0
1. View all orders
2. Update order status
3. Back
Enter your choice: 2
Enter order ID to update: 4
Enter new status (1 for being prepared, 2 for ready): 2
1. View all orders
2. Update order status
3. Back
Enter your choice: 3
```

```
Admin Menu:
1. Total Sales
2. Manage Menu
3. Manage Orders
4. Manage Users
5. Logout
Enter choice: 4

--- Manage Users ---
1. Add User
2. View Usernames
3. Delete User
4. Back
Enter your choice: 1

--- Add User ---
1. Add Admin
2. Add Chef
3. Back
Enter your choice: 1
Enter new admin username: user
Enter new admin password: user
Admin user added successfully.

--- Add User ---
1. Add Admin
2. Add Chef
3. Back
```

```
Enter your choice: 2
Enter new chef username: chef1
Enter new chef password: chef1
Chef chef1 added successfully.

--- Add User ---
1. Add Admin
2. Add Chef
3. Back
Enter your choice: 3

--- Admins ---
admin
user

--- Chefs ---
chef
chef1

--- Manage Users ---
1. Add User
2. View Usernames
3. Delete User
4. Back
Enter your choice: 2
```

```
--- Admins ---
admin
user

--- Chefs ---
chef
chef1

--- Manage Users ---
1. Add User
2. View Usernames
3. Delete User
4. Back
Enter your choice: 3

--- Delete User ---
1. Delete Admin
2. Delete Chef
3. Back
Enter your choice: 1
Enter admin username to delete: user
Admin user deleted successfully.

--- Admins ---
admin
```

```
--- Chefs ---
chef
chef1

--- Manage Users ---
1. Add User
2. View Usernames
3. Delete User
4. Back
Enter your choice: 4

Admin Menu:
1. Total Sales
2. Manage Menu
3. Manage Orders
4. Manage Users
5. Logout
Enter choice: 5

1. Place Order
2. View Menu
3. Chef Login
4. Admin Login
5. Exit
```


Enter your choice: 9
Invalid choice. Please try again.
1. Place Order
2. View Menu
3. Chef Login
4. Admin Login
5. Exit
Enter your choice: 5
Thankyou !

TESTING

Software Testing is an empirical investigation conducted to provide stakeholders with information about the quality of the product or service under test[1] , with respect to the context in which it is intended to operate. Software Testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks at implementation of the software. Test techniques include, but are not limited to, the process of executing a program or application to find software bugs.

It can also be stated as the process of validating and verifying that a software program/application/product meets the business and technical requirements that guided its design and development, so that it works as expected and can be implemented with the same characteristics. Software Testing, depending on the testing method employed, can be implemented at any time in the development process, however the most test effort is employed after the requirements have been defined and coding process has been completed.

TESTING METHODS

Software testing methods are traditionally divided into black box testing and white box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

➢BLACK BOX TESTING

Black box testing treats the software as a "black box," without any knowledge of internal implementation. Black box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing and specification‐based testing.

➢SPECIFICATION-BASED TESTING

Specification-based testing aims to test the functionality of software according to the applicable requirements.[16] Thus, the tester inputs data into, and only sees the output from, the test object. This level of testing usually requires thorough test cases to be provided to the tester, who then can simply verify that for a given input, the output value (or behaviour), either "is" or "is not" the same as the expected value specified in the test case. Specification‐based testing is necessary, but it is insufficient to guard against certain risks

➢ADVANTAGES AND DISADVANTAGES

The black box tester has no "bonds" with the code, and a tester's perception is very simple: a code must have bugs. Using the principle, "Ask and you shall receive," black box testers find bugs where programmers don't. But, on the other hand, black box testing has been said to be "like a walk in a dark labyrinth without a flashlight," because the tester doesn't know how the software being tested was actually constructed.

That's why there are situations when (1) a black box tester writes many test cases to check something that can be tested by only one test case, and/or (2) some parts of the back end are not tested at all. Therefore, black box testing has the advantage of "an unaffiliated opinion," on the one hand, and the disadvantage of "blind exploring," on the other

➢**WHITE BOX TESTING**

White box testing, by contrast to black box testing, is when the tester has access to the internal data structures and algorithms (and the code that implement these)

Types of white box testing:-

The following types of white box testing exist:

- api testing - Testing of the application using Public and Private APIs.

- Code coverage - creating tests to satisfy some criteria of code coverage. For example, the test designer can create tests to cause all statements in the program to be executed at least once.
- fault injection methods.
- mutation testing methods.
- static testing - White box testing includes all static testing

➢CODE COMPLETENESS EVALUATION

White box testing methods can also be used to evaluate the completeness of a test suite that was created with black box testing methods. This allows the software team to examine parts of a system that are rarely tested and ensures that the most important function points have been tested. Two common forms of code coverage are: · Function Coverage: Which reports on functions executed and · Statement Coverage: Which reports on the number of lines executed to complete the test. They both return coverage metric, measured as a percentage

HARDWARE AND SOFTWARE REQUIREMENTS

- **OPERATING SYSTEM** : WINDOWS 7 AND ABOVE
- **PROCESSOR** : PENTIUM(ANY) OR AMD ATHALON(3800+- 4200+ DUAL CORE
- **MOTHERBOARD** : 1.845 OR 915,995 FOR PENTIUM OR MSI K9MM-V VIA K8M800+8237R PLUS CHIPSET FOR AMD ATHALON
- **RAM** : 512MB+
- **Hard disk** : SATA 120 GB OR ABOVE
- **MONITOR** 14.1 or 15 -17 inch
- **Key board and mouse**
- **Printer : required**

SOFTWARE REQUIREMENTS:

- **Windows OS**
- **Python**

INSTALLATION PROCEDURE

DIGITAL MENU SYSTEM:-

Pre-Requisites :-

---

1. You have to have the following softwares for the successful running of this software; which are

I) Python (Only for the First time), it is downloadable from 'www.python.org'.

II) MySQL (Only for the First time), it is downloadable from 'www.mysql.org'.

Installation :-

---

1. There will be two folders namely 'Python Files' and 'EXE files'.

2. The folder 'Python Files' will contain the source code of the software in python language. If you are running the software by the 3rd step mentioned below you have to pre install the following modules :-

I) mysql.connector

II) matplotlib.

1. Open the files in any python editors and run it to start and work on the software.

2. The folder 'EXE files' will contain two files namely 'main.exe' and 'Tables_in_mysql.exe'.

3. First run the 'Tables_in_mysql.exe' to create the tables in MySQL.

4. Then run the file 'main.exe' to start and work on the software.

CAUTION :-
=========

If you are running the software through running the python files or by running the .exe files ; first run the file named 'Tables_in_mysql'.

The .exe file will take a lot of time; so be PATIENT.

BIBLIOGRAPHY

- Computer science With Python - Class XI By : Sumita Arora
- Computer science With Python - Class XII By : Sumita Arora
- A Project Report On Space Invader Game and Gold Loan Shop Management (GLSM) By :ADWAITH SANTHOSH
- Website: https://www.youtube.com
  https://www.mysql.com
  https://www.google.com
  https://www.canva.com
  https://www.github.com