

**JOMO KENYATTA UNIVERSITY OF AGRICULTURE AND TECHNOLOGY (JKUAT)**

**DEPARTMENT OF ELECTRICAL & ELECTRONIC ENGINEERING**

**B.Sc. ELECTRICAL AND ELECTRONIC ENGINEERING**

**ICS 2276 COMPUTER PROGRAMMING II**

**PROJECT WRITEUP**

**JKUAT DIGITAL MEAL TICKETING SYSTEM**

|  |  |
| --- | --- |
| **GROUP MEMBERS**: | **REGISTRATION NUMBERS** |
| 1. DENNIS MUENDO - | ENE211-0025/2020 |
| 2. JAMES WAIGWA- | ENE211-0047/2020 |
| 3. REAGAN SIMIYU- | ENE211-0279/2020 |
| 4. ELLERTONE ONGOKI - | ENE211-0055/2020 |

**PROBLEM STATEMENT**

The JKUAT mess has two service points from which payment of food is done. These two or sometimes three serving points serve each individual for around four minutes. The time is spent on the selection of the food the individual wants to eat, the payment by the individual and the authentication of the payment before the receipt is given.

The mess also has peak times when the line longest and individuals will take longer to get their food. During these peak times one can expect to wait for thirty minutes to get their receipts and this leads to losses as when students go for other options. This system is here to streamline this process.

**OBJECTIVES**

1. To reduce the time taken to get receipts therefore increasing efficiency.
2. To collect data on the preference of the students to inform sales in the mess.
3. To reduce congestion in the mess by showing peak times when less people should go the mess

**PROJECT OVERVIEW**

To meet the above objectives, our project will entail the following features

1. Sign up and login features for users.
2. A list of all the offerings in the school mess.
3. A payment system once the user selects the food of their wish
4. An automatically generated receipt which is used to get food from the mess.

# CODE

#import the json library

import json

class User:

def \_\_init\_\_(self, username, password):

# Initialize a User object with a username and password

self.username = username

self.password = password

class MealTicket:

def \_\_init\_\_(self, meal\_code, meal\_name, meal\_price, quantity):

# Initialize a MealTicket object with a meal code, meal name, meal price, and quantity

self.meal\_code = meal\_code

self.meal\_name = meal\_name

self.meal\_price = meal\_price

self.quantity = quantity

self.total\_cost = meal\_price \* quantity

class Student(User):

def \_\_init\_\_(self, username, password, student\_id, balance):

# Initialize a Student object with a username, password, student ID, balance, and an empty list of meal tickets

super().\_\_init\_\_(username, password)

self.student\_id = student\_id

self.meal\_tickets = []

self.balance = balance

self.sales = Sales()

def place\_order(self, meal\_code, quantity):

# Allow the student to place an order by creating a MealTicket object, adding it to the student's list of meal tickets,

# and recording the sale in the system's Sales object

if meal\_code not in MEAL\_OPTIONS:

print("Invalid meal code, please try again")

return

meal\_name, meal\_price = MEAL\_OPTIONS[meal\_code]

meal\_ticket = MealTicket(meal\_code, meal\_name, meal\_price, quantity)

self.meal\_tickets.append(meal\_ticket)

self.balance -= meal\_ticket.total\_cost

self.sales.record\_sale(self.student\_id, meal\_ticket)

def cancel\_order(self, meal\_ticket):

# Allow the student to cancel an order by removing the corresponding MealTicket object from their list of meal tickets

# and adding the cost back to their balance

self.meal\_tickets.remove(meal\_ticket)

self.balance += meal\_ticket.total\_cost

class Admin(User):

def \_\_init\_\_(self, username, password, admin\_id):

# Initialize an Admin object with a username, password, and admin ID

super().\_\_init\_\_(username, password)

self.admin\_id = admin\_id

def view\_sales(self):

# Display the total sales and a breakdown of sales by meal for all recorded sales in the system's Sales object

with open('sales.json', "r") as f:

sales\_data = json.load(f)

total\_sales = sum(sale['total\_cost'] for sale in sales\_data)

print(f"Total sales: {total\_sales}")

print("Sales breakdown by meal:")

for meal\_code in MEAL\_OPTIONS.keys():

meal\_name, meal\_price = MEAL\_OPTIONS[meal\_code]

quantity\_sold = sum(sale['quantity'] for sale in sales\_data if sale['meal\_code'] == meal\_code)

total\_sales\_for\_meal = sum(sale['total\_cost'] for sale in sales\_data if sale['meal\_code'] == meal\_code)

print(f"{meal\_name}: {quantity\_sold} sold for a total of {total\_sales\_for\_meal}")

def add\_meal(self, meal\_code, meal\_name, meal\_price):

# Add a new meal option to the system's MEAL\_OPTIONS dictionary

MEAL\_OPTIONS[meal\_code] = (meal\_name, meal\_price)

def remove\_meal(self, meal\_code):

# Remove a meal option from the system's MEAL\_OPTIONS dictionary

if meal\_code in MEAL\_OPTIONS:

del MEAL\_OPTIONS[meal\_code]

def create\_student\_account(self):

# Prompt the admin to create a new student account and add it to the system's STUDENTS dictionary and students.json file

username = input("Enter a username for the student: ")

password = input("Enter a password for the student: ")

student\_id = input("Enter a student ID: ")

balance = int(input("Enter a starting balance: "))

with open('students.json', 'r+') as f:

students\_data = json.load(f)

students\_data.append({'username': username, 'password': password, 'student\_id': student\_id, 'balance': balance})

f.seek(0)

json.dump(students\_data, f, indent=4)

if username in system.students:

print("A user with that username already exists.")

else:

student = Student(username, password, student\_id, balance)

system.students[username] = student

print(f"Student account created with username: {username}")

def view\_student\_accounts(self):

# Display a list of all the students in the system's STUDENTS dictionary

for username, student in system.students.items():

print(f"Username: {username}, Student ID: {student.student\_id}, Balance: {student.balance}")

# Define a Sales class to manage the sales record

class Sales:

# Initialize an empty sales record list

def \_\_init\_\_(self):

self.sales\_record = []

# Record a sale by adding the sale details to the sales record list and updating the sales data file

def record\_sale(self, student\_id, meal\_ticket):

# Open the sales data file and load the existing sales data into a list

with open('sales.json', 'r') as f:

sales\_data = json.load(f)

# Create a new sales record dictionary with the provided sale details

new\_sale = {'student\_id': student\_id,

'meal\_code': meal\_ticket.meal\_code,

'meal\_name': meal\_ticket.meal\_name,

'meal\_price': meal\_ticket.meal\_price,

'quantity': meal\_ticket.quantity,

'total\_cost': meal\_ticket.total\_cost}

# Add the new sales record to the sales data list

sales\_data.append(new\_sale)

# Open the sales data file in write mode and update it with the updated sales data list

with open('sales.json', 'w') as f:

json.dump(sales\_data, f, indent=4)

class System:

def \_\_init\_\_(self):

# Load students, admins and sales data from files

self.students = self.load\_students\_from\_file()

self.admins = self.load\_admins\_from\_file()

self.sales = Sales()

self.login\_attempts = 0

def load\_admins\_from\_file(self):

# Load data from admins file

with open('admins.json', 'r') as f:

admins\_data = json.load(f)

admins = {}

# Create an Admin object for each admin data and add it to the dictionary

for admin\_data in admins\_data:

admin = Admin(admin\_data['username'], admin\_data['password'], admin\_data['admin\_id'])

admins[admin\_data['username']] = admin

return admins

def load\_students\_from\_file(self):

# Load data from students file

with open('students.json', 'r') as f:

students\_data = json.load(f)

students = {}

# Create a Student object for each student data and add it to the dictionary

for student\_data in students\_data:

student = Student(student\_data['username'], student\_data['password'], student\_data['student\_id'], student\_data['balance'])

students[student\_data['username']] = student

return students

def login(self):

print("\nWELCOME TO THE JKUAT MEAL SERVICE.\n")

print("Please enter your details to log in.")

for attempt in range(3):

username = input("Enter your username: ")

password = input("Enter your password: ")

if username in self.admins:

if self.admins[username].password == password:

print("\nLogin successful as admin.")

return self.admins[username]

else:

print("Incorrect password.")

elif username in self.students:

if self.students[username].password == password:

print("\nLogin successful as student.")

return self.students[username]

else:

print("Incorrect password.")

print(f"Invalid username or password. Attempts remaining: {2-attempt}")

print("Maximum number of login attempts reached.")

exit()

MEAL\_OPTIONS = {

"1": ("Tea", 15),

"3": ("Bread Slice", 3),

"7": ("Ndazi", 10),

"9": ("Coffee", 15),

"13": ("Chapati", 20),

"14": ("Bun", 20),

"15": ("Sukuma", 13),

"16": ("Beef", 120),

"17": ("Ugali", 15),

"18": ("Bean", 20),

"20": ("Rice", 23),

"21": ("African Stew", 30),

"22": ("Omelette", 20),

"24": ("Githeri", 20),

"47": ("Boiled Egg", 20)

}

# Call the login method of the system object to authenticate the user

sales = Sales()

system = System()

user = system.login()

# Check if the user is a student

if isinstance(user, Student):

# Display welcome message and current balance

print(f"Welcome {user.username}. Your balance is {user.balance}.\n")

# Loop until the user logs out

while True:

# Display options for the student

print("Please select an option:")

print("1. Place order")

print("2. Cancel order")

print("3. View order history")

print("4. Logout")

# Get user input for selected option

option = input("\nEnter option number: ")

# If option 1 is selected

if option == "1":

# Display available meals and their codes and prices

print("\nHere are the available meals:\n")

print("Code | Name | Price")

for code, (name, price) in MEAL\_OPTIONS.items():

print(f"{code} | {name} | {price}")

# Get user input for meal code and quantity

meal\_code = input("\nEnter meal code: ")

quantity = int(input("Enter quantity: "))

meal\_price = MEAL\_OPTIONS[meal\_code][1]

total\_cost = meal\_price \* quantity

# Check if user has enough balance to place the order

if user.balance >= total\_cost:

# Place the order and display message with updated balance

user.place\_order(meal\_code, quantity)

print("Order placed successfully!Your balance is", user.balance)

else:

# Display message if user has insufficient balance

print("Sorry, you do not have enough balance to place this order.Your balance is",user.balance)

# If option 2 is selected

elif option == "2":

# Display current orders of the user

print("Here are your current orders:\n")

print("Code | Name | Price | Quantity | Total Cost")

for meal\_ticket in user.meal\_tickets:

print(f"{meal\_ticket.meal\_code} | {meal\_ticket.meal\_name} | {meal\_ticket.meal\_price} | {meal\_ticket.quantity} | {meal\_ticket.total\_cost}")

# Get user input for the ticket code of the order to be cancelled

ticket\_code = input("\nEnter ticket code to cancel: ")

# Loop through the user's meal tickets to find the order with the ticket code

for meal\_ticket in user.meal\_tickets:

if meal\_ticket.meal\_code == ticket\_code:

# Cancel the order and display success message

user.cancel\_order(meal\_ticket)

print("\nOrder cancelled successfully!\n")

break

else:

# Display message if invalid ticket code is entered

print("Invalid ticket code, please try again")

# If option 3 is selected

elif option == "3":

# Display order history of the user

print("Here are your order history:")

print("Code | Name | Price | Quantity | Total Cost")

for meal\_ticket in user.meal\_tickets:

print(f"{meal\_ticket.meal\_code} | {meal\_ticket.meal\_name} | {meal\_ticket.meal\_price} | {meal\_ticket.quantity} | {meal\_ticket.total\_cost}")

elif option == "4":

# Update student's balance in students.json before logging out

with open('students.json', 'r') as f:

students = json.load(f)

for student in students:

if student['username'] == user.username:

student['balance'] = user.balance

break

with open('students.json', 'w') as f:

json.dump(students, f, indent=4)

print("Thank you for using JKUAT Meal Service!")

break

else:

print("Invalid option, please try again.")

# If the logged in user is an admin

elif isinstance(user, Admin):

# Print a welcome message for the admin

print(f"Welcome {user.username}.\n")

# Keep looping until the admin logs out

while True:

# Display the available options for the admin

print("\n Please select an option:")

print("1. View sales")

print("2. Add meal")

print("3. Remove meal")

print("4. Create student account")

print("5. View student accounts")

print("6. Logout \n")

# Get the admin's option choice

option = input("Enter option number: ")

# If option 1 is selected

if option == "1":

# View the total sales made by the system

sales = user.view\_sales()

# If option 2 is selected

elif option == "2":

# Add a new meal to the available options

meal\_code = input("Enter meal code: ")

meal\_name = input("Enter meal name: ")

meal\_price = int(input("Enter meal price: "))

user.add\_meal(meal\_code, meal\_name, meal\_price)

print("Meal added successfully!")

# If option 3 is selected

elif option == "3":

# Remove a meal from the available options

meal\_code = input("Enter meal code to remove: ")

user.remove\_meal(meal\_code)

print("Meal removed successfully!")

# If option 4 is selected

elif option == "4":

# Create a new student account

user.create\_student\_account()

# If option 5 is selected

elif option == "5":

# Display all existing student accounts

user.view\_student\_accounts()

# If option 6 is selected

elif option == "6":

# Print a logout message for the admin

print("Thank you for using JKUAT Meal Service!")

break

# If an invalid option is selected

else:

print("Invalid option, please try again")

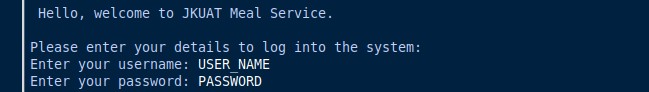
else:

print("User not found, exiting system")

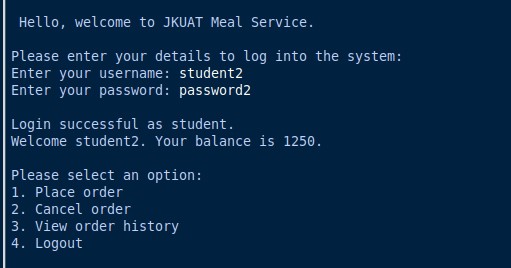
exit()

**OUTPUTS**

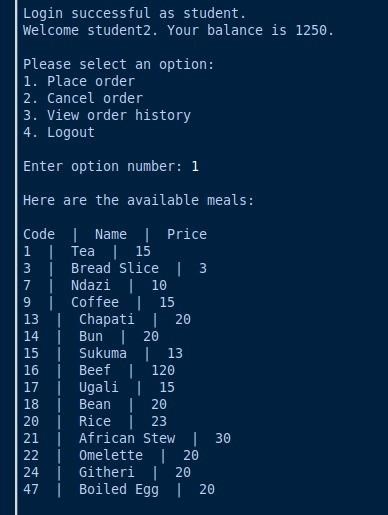
# Welcome Screen



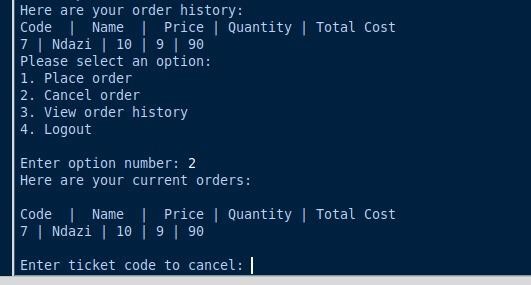
# Student login



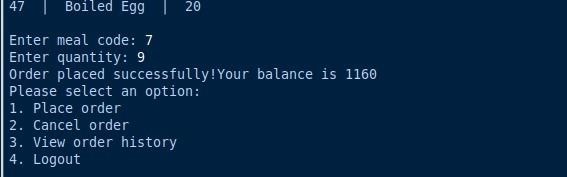
# Menu



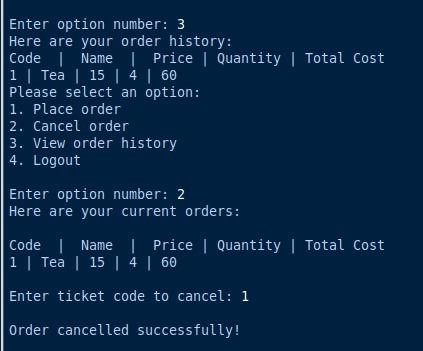
# Placing an order



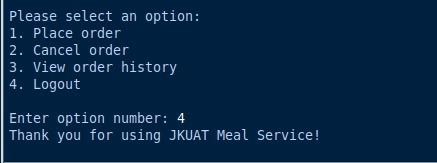
# Successful order confirmation



# Cancelling an order



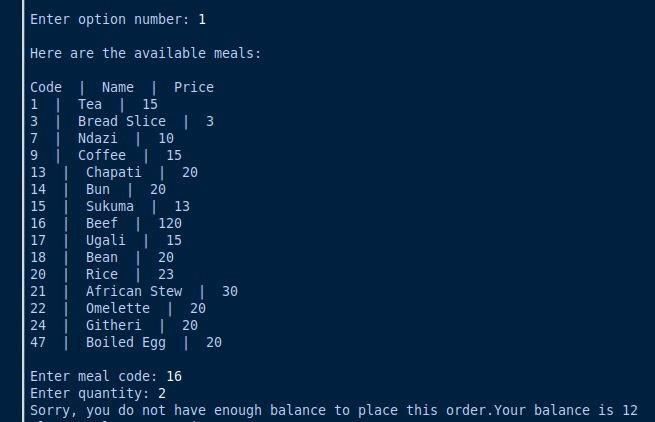
# Logging out of the system



# User with insufficient balance case Logging in



# Order unsuccessful due to insufficient funds



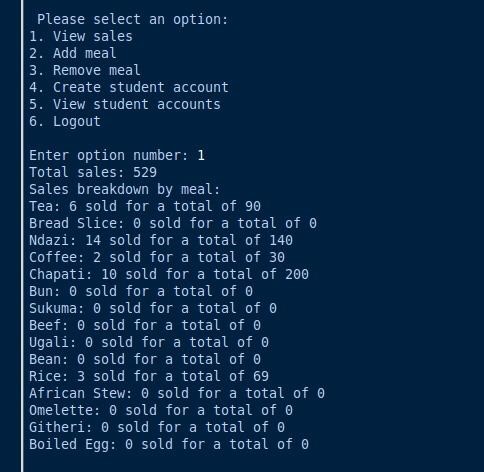
# User not in the system trying to login



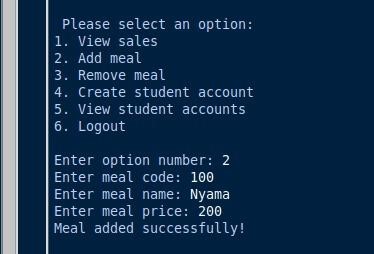
# ADMIN LOGIN



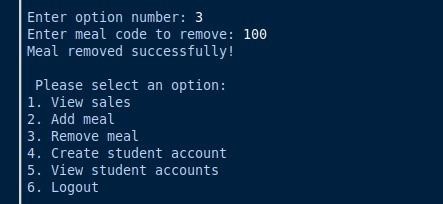
# VIEWING SALES



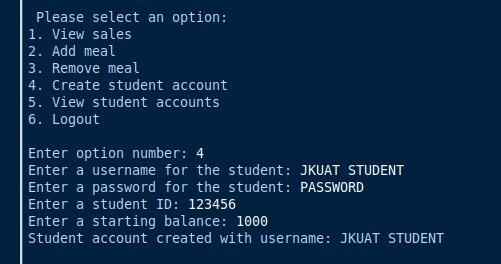
# ADDING A NEW MEAL



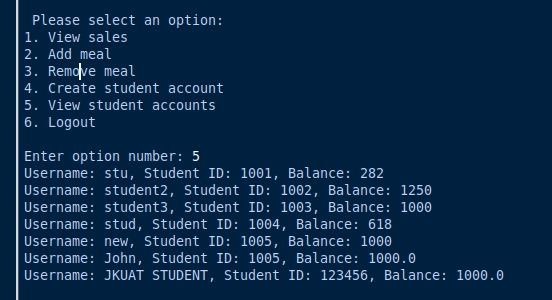
# REMOVING A MEAL



# CREATING A NEW STUDENT ACCOUNT



# VIEWING STUDENT ACCOUNTS IN THE SYSTEM



# ADMIN LOGOUT

