Date:10.4.2019

ON CAMPUS FOOD ORDERING SYSTEM

Shelly Choudhary
Pratigya Shakya Pradhan
Vyshampa Maringanti

Table of Contents

1. Vision Statement

1.1. Vision

The goal of this project is to develop an online food ordering system for the students of University of New Haven. This software should help students and university staff to order from any location, pay online and pick up food from different café available around the campus. The admin and café employee should be able to manage the incoming orders through the online portal.

1.2. Project

Online On campus food ordering system

1.3. Target Customers

Students, staff and anyone near the premises of University of New Haven

1.4. Project Objectives

The functionalities of this application are discussed below:

a. Roles and access management

i) Administrator (Owner of the system)

- The administrator can monitor the activities of all cafes (view menu, change menu or price, view order history, view customer feedback)
- The admin can add and delete account for an administrator or an employee

ii) Café employee

- Café employee have access to the part of database specific to their cafe

iii) Customer

- The customer can only navigate through the given menu of the available café and make online purchase
- The customer can view different café options available in the university like Bartels, and library café.
- The user should be able to select one of the cafes and view the menu of that café.
- The user should be able to view and select food from the menu
- The user should be able to add/remove/edit the food selection and make changes to their personal information before checking out
- The user should be able to checkout securely
- They should get an order confirmation number for pick up purpose

b. Menu Management

- The employee should be able to manage menu displayed on the user end by adding, deleting items from the menu
- The employees should be able to update the price of a given item (Example: Item name: Mashed potato bacon pizza, Description: Mashed potato, bacon, cheddar and mozzarella

c. Order Management
    - Café employees should be able to check the notification for order detail
    - The orders should follow queue data structure to store the incoming order
    - The employee should be able to view the confirmation number and food items
    - The order should be removed from the queue once it is ready for pick up

d. Notification
    - The café employee should get notified once the order is placed by the customer

e. Online Payment
    - The user should provide name as shown on the card, card number, expiry date and CVV number to make the payment

f. Feedback
    - The application will allow user to write feedbacks by clicking on the feedback menu to select the desired café
    - Each café employee can access their feedback table from the database

1.5. Future Enhancements

- Artificial Intelligent bot: User can chat with a bot to make an order

2. Product Backlog

ID	User stories	Priority	Estimate
5	As an end user, navigate the website to view information	1	7
6	As a user, search the food item	2	4
1	As an admin/employee, login	5	5
2	As an admin/employee, logout	6	3
s	As an admin, add or remove employee to the system	7	6
3	As a customer, register and login	8	4
4	As a customer, logout	9	3
7	As a user, select items on menu	10	4
11	As an admin, add, delete and edit menu	11	5
15	As an employee, add/delete/ modify item or change price of items in the menu	12	3
8	As a user, add payment detail such as card number, expiry date, card holder's name, and CVV	13	6
9	As a user, modify order and payment detail before payment	14	5
10	As a user, make the payment and receive order confirmation number	15	8
16	The system should notify the employee when the order is placed by the user	16	4
17	Café employee should be able to check the notification	17	10
18	The orders should remain in the system in a first in first out order until they are ready for pickup	18	4
19	As an employee, view order (confirmation number, food item)	19	6
22	As a user, write feedback	20	3
23	As an admin, view feedbacks of all the café	21	3
24	As an employee, view feedbacks of the designated cafe	22	3
14	As an admin, monitor activities of all café	23	12
12	As an admin, add or remove admin	24	5

3. UML

3.1. Fully Dressed Use Cases

1. Use Case PB2: Admin/ Employee Logout

Use Case	Comments
Scope	On Campus Food Ordering System
Level	Employee Goal
Primary Actor	Employee and system
Stakeholders and Interests	Employee: Wants to make changes in the menu. Change the price of an existing item, add/remove/modify item from the menu. Changes the status of the order "Open" to "ready for pick up" and from "Ready for pick up" to "pick up successful" and removes it from the queue.
Preconditions	1. Café employee must be an authorized user 2. Employee must be logged into the system
Success Guarantee (or Postconditions)	Cafe employee should be able to logout of the application successfully after completing the intended task.
Main Success Scenario	1. Logs out successfully after completing the intended work by clicking on the logout button.
Extensions	1a. Internet connection is lost and web page unable to open. 2a. Employee tries to login with wrong user id/password. 4a. Unable to logout from the session.

2. Use Case PB2: Admin/Employee Login

Use Case	Comments
Scope	On Campus Food Ordering System
Level	Employee Goal
Primary Actor	Employee and system
Stakeholders and Interests	Employee: Wants to make changes in the menu. Change the price of an existing item, add/remove/modify item from the menu. Changes the status of the order "Open" to "ready for pick up" and from "Ready for pick

	up" to "pick up successful" and removes it from the queue.
Preconditions	1. Café employee must be an authorized user. 2. Employee must be logged into the system.
Success Guarantee (or Postconditions)	1. The employee should login by providing user Id and password 2. Clicks on the Login button.
Main Success Scenario	Employee should be able to login to the system with valid credentials successfully.
Extensions	1a. Employee forgets the credentials. 1b. Internet connection is lost and web page unable to open.

3. Use Case PB6: Select café

-	Comments
Scope	On Campus Food Ordering System
Level	User Goal
Primary Actor	Customer and the System
Stakeholders and Interests	Customer: Should be able to login to the system and select the cafe option.
Preconditions	1. The customer views cafe option. 2. System allows the customer to select cafe option.
Success Guarantee (or Postconditions)	Customer successfully selects the cafe.
Main Success Scenario	1. Customer login to the system. 2. Check the system is connected to the internet. 3. Go to the web page. 4. Select the cafe option.
Extensions	1a. internet is down b. not able to login 4a. Cafe option not available

4. Use Case PB7: Select items on menu

Use Case	Comments

Scope	On Campus Food Ordering System
Level	User Goal
Primary Actor	Customer
Stakeholders and Interests	Customer: Wants to select items from the menu
Preconditions	1. The customer views menu 2. System allows the customer to select one or more items.
Success Guarantee (or Postconditions)	1. Customer successfully selects the items from the menu.
Main Success Scenario	1. Customer login to the system. 2. Check the system is connected to the internet. 3. Go to the web page. 4. Select a specific cafe. 5. View the items in the menu 6. Selects one or more items from the menu.
Extensions	1a. Internet is down 6a. Unable to select any item from the menu. 6b. Unable to select one or more items

5. Use Case PB11: Addition, deletion and editing food menu by the admin

Use Case	Comments
Scope	On Campus Food Ordering System
Level	Admin Goal
Primary Actor	admin and system
Stakeholders and Interests	Admin: Wants to edit the menu, i.e. add food item, remove food item, change price, change the description of the food item
Preconditions	The admin must be logged in and authorized by the system.
Success Guarantee (or Postconditions)	The admin can successfully add, remove or edit food menu in the system.
Main Success Scenario	1. The admin would be able to add food items in the menu by keying in the details. 2. The admin would be able to remove food items in the menu. 3. The admin would be able to edit food items in the menu.

	4. The system shows a success message to the admin after completing the required task.
Extensions	1a. The admin couldn't add a food item, 1b. The food item already exists in the menu. 2a. The admin couldn't remove a food item. 2b. The food item doesn't exist in the list. 3a. The system shows an erroneous message instead of successful message.

6. Use Case PB8: Payment information and verification

Use Case	Comments
Scope	On Campus Food Ordering System
Level	User Goal
Primary Actor	Customer and third-party payment authorization service.
Stakeholders and Interests	Customer: wants to select a café, look through a menu and order food and beverages Payment authorization service: Proper authorization of payment is done by third party. Wants to receive digital authorization requests in the correct protocol and format
Preconditions	1. The customer must be logged into the system. 2. The customer must select the items from the menu and items are added to the cart. 3. The customer should be present on the payment screen
Success Guarantee (or Postconditions)	The customer successfully pays for the online order
Main Success Scenario	1. Customer enters their credit card information like credit card number, expiry date, the name of the card holder, CVV number. 2. System displays their payment information for verification. 3. System sends payment authorization request to an external Payment Authorization Service System, and requests for payment approval.

	4. System receives payment approval, signals approval to application, and releases payment confirmation receipt. 5. System records the credit payment, which includes the payment approval.
Extensions	1a. System signals timeout for current session 2a. System asks Customer for alternate payment 3a. System detects failure to collaborate with external system 3b. System signals error to payment verification page 4a. System receives payment denial

7. Use Case PB18: Order ready for pickup

Use Case	Comments
Scope	On Campus Food Ordering System
Level	User goal
Primary Actor	Customer
Stakeholders and Interests	Customer: Wants to pick up the ordered item within the system generated ETA, estimated time of arrival
Preconditions	1. Cafe employee has accepted the order 2. Customer has received confirmation number and estimated time for order to get ready
Success Guarantee (or Postconditions)	1. The customer picks up the correct order and is satisfied with the quality of the product 2. The order is removed from the queue in the system
Main Success Scenario	1. The customer arrives at the cafe with the order confirmation detail 2. The employee checks the order confirmation details and hands the order if the confirmation number with the customer and matches the detail in the cafe's system 3. The customer receives the order

	4. The order gets removed from the queue in the system
Extensions	1a. The customer successfully paid the bill but received no confirmation number 1b. The session expired as soon as the customer paid for the order 2a. The customer's order did not get recorded in the employee's system 3a. The customer gets the wrong order

8. Use Case PB15: Add/Delete/Modify item or change their price on the menu

Use Case	Comments
Scope	On Campus Food Ordering System
Level	User goal
Primary Actor	Employee
Stakeholders and Interests	Employee: Wants to make changes in the menu. Change the price of an existing item, add/remove/modify item from the menu
Preconditions	1. The employee should be logged into the system with the credential.
Success Guarantee (or Postconditions)	1. The employee is able to make the required changes in menu.
Main Success Scenario	1. The employee logins in the system. 2. If required, the employee is able to make the required changes in the system.
Extensions	1a. The employee is unable to login 2a. The system does not allow the employee to make the required changes

9. Use CasePB9: Modify order or payment details before payment

Use Case	Comments
Scope	On Campus Food Ordering System
Level	User goal
Primary Actor	Customer
Stakeholders and Interests	Customer: Wants to make final changes to the selected order from the menu or make changes to the payment details (correct errors, use another card for payment)

Preconditions	1. The customer must have selected the order 2. The customer is ready to make the payment
Success Guarantee (or Postconditions)	1. The customer is able to modify order or make changes in payment details
Main Success Scenario	1. Customer selects the items from menu 2. Customer should enter the details of payment 3. Customer is able to make changes to the payment details and order before submitting the payment.
Extensions	3a. The session expired 3b. System does not allow the customer to make changes to the order or payment detail

10. Use CasePB10: Make the payment and receive order confirmation number

Use Case	Comments
Scope	On Campus Food Ordering System
Level	User level
Primary Actor	System and customer
Stakeholders and Interests	Customer: Make the payment for the order
Preconditions	1. The customer must select the items from the menu and items are added to the cart 2. The customer should be successful in making the payment
Success Guarantee (or Postconditions)	1. The order will be placed in the system. 2. The customer will receive a confirmation ID 3. The customer will receive an estimated pickup time for the order
Main Success Scenario	1. The system will indicate that the customer has been charged for the order 2. System successfully records the transaction and generates confirmation ID 3. Customer can view the order detail 4. Cafe employee will receive the order

	5. Cafe employee will check the order and estimated pickup time 6. Confirmation ID is sent to the customer
Extensions	4a. Due to system error, the cafe employee fails to receive the order 6a. The customer does not receive an order confirmation number

11. Use Case PB17, 18,19: Notify employee of the order placed. Check notification and view order (As an Employee)

Use Case	Comments
Scope	On Campus Food Ordering System
Level	Employee Goal
Primary Actor	Employee
Stakeholders and Interests	Employee: Wants to check the system when notification is received and view the order
Preconditions	1. Café employee must be an authorized user. 2. Employee must be logged into the system. 3. Employee should confirm the order from his web page.
Success Guarantee (or Postconditions)	1. Café employee hold the rights to view the number orders of that specific café along with the Order status like “Completed”, “In progress”, “Cancelled”.
Main Success Scenario	1. Café employee logs into the application. 2. Navigates to the orders tab in the menu bar to view notification. 3. Clicks on the orders tab to view the queue of recently placed orders. 4. Employee can also view the details and status of the order.
Extensions	1a. Employee forgets the credentials to login. 2a. Employee tries to view the orders of different café which is not associated with the employee.

12. Use Case PB22: Write Feedback

Use Case	Comments
Scope	On Campus Food Ordering System
Level	User Goal
Primary Actor	Customer and the System
Stakeholders and Interests	Customer: Wants to pick up the ordered item within the system generated ETA, estimated time of arrival and wants to write a review/ feedback on the application
Preconditions	1. Customer has received one or multiple order pickup's successfully. 2. Customer must have some experience with OnCampus Food Ordering System. 3. Customer must have ordered at least once from the specific café.
Success Guarantee (or Postconditions)	1. Customer can complete a feedback form and submit from the system successfully. 2. Feedback should be saved to the system and records in the database successfully.
Main Success Scenario	1. Customer login to the system. 2. Check the system is connected to the internet. 3. Go to the web page. 4. Selects the café for which he/she wants to provide feedback. 5. Selects feedback form from the Menu bar of the café. 6. System should allow customers to view and edit the feedback form. 7. Customer can write detailed feedback in "Enter Feedback" text field. 8. Clicks on submit button.
Extensions	1a. Customer is unable to login to the system 6a. The system does not allow the customer to edit the feedback form or make required changes in the form.

13. Use CasePB24: View Feedback (of designated café as an employee)

Use Case	Comments
Scope	OnCampus Food Ordering System

Level	Employee Goal
Primary Actor	Employee and System
Stakeholders and Interests	Employee: Wants to make changes in the menu. Change the price of an existing item, add/remove/modify item from the menu.
Preconditions	1. Café Employee must be an authorized user. 2. Café Employee must be logged into the system. 3. Customer should confirm the order from his web page. 4. At least one feedback is received from the customer. 5. At least one feedback form must be recorded by the system.
Success Guarantee (or Postconditions)	Employee should be able to view the feedback forms submitted by the users to the designated cafe.
Main Success Scenario	2. Employee should login to the system. Employee should be able to view the web page of his/her designated café with specific access level. 3. Selects feedback form from the Menu bar of the café. 4. Employee should be able to see the list of feedback provided by the customers (displayed in queue data structure). 5. System should allow employee to view the feedback form submitted by the customer of specific cafe.
Extensions	1a. Employee is unable to login to the system. 5a. The system does not allow any employee to view the feedback form from any of the cafes. 5b. The system does not allow the employee to view the feedback form from one specific café.

14. Use Case PB14: Monitor Activities (As an admin, Tracking order and view feedback from all cafe's)

Use Case	Comments
Scope	On Campus Food Ordering System

Level	Admin Goal
Primary Actor	Admin and the System
Stakeholders and Interests	Admin: Wants to track the status of the order and view the feedback from the customers.
Preconditions	1. Admin must be an authorized user. 2. Admin must be logged into the system. 3. Customer should order food from one of the café. 4. At least one feedback is received from the customer. 5. At least one feedback form must be recorded by the system
Success Guarantee (or Postconditions)	Admin should be able to track/monitor the orders, sales and feedbacks of all café successfully.
Main Success Scenario	1. Admin should login to the system. 2. Selects the café for which he/she wants to view the feedback. 3. Selects feedback form from the Menu bar of the café. 4. Admin should be able to see the list of feedback provided by the customers (displayed in queue data structure). 5. System should allow admin to view the feedback form submitted by the customer of specific cafe. 6. System should allow admin to track the orders from orders tab. 7. System should allow admin to view the current orders from all café's. 8. System should allow admin to view the completed orders from the orders. 9. System should allow admin to track total payments from the database.
Extensions	1a. Admin is unable to login to the system. 5a. The system does not allow the admin to view the feedback form from all the cafes. 5b. The system does not allow the admin to view the feedback form from one specific café.

	7a. System does not allow admin to view the current orders which are in progress. 8a. System does not allow admin to view the status=completed orders. 9a. System does not allow admin to track total payments from the database.

15. Use CasePB12(i): Add admin (As an admin)

Use Case	Comments
Scope	OnCampus Food Ordering System
Level	Admin Goal
Primary Actor	Admin and the System
Stakeholders and Interests	Admin: Wants to add another admin for the application to monitor orders and sales of the cafe.
Preconditions	1. Admin must be an authorized user. 2. Admin must be logged into the system.
Success Guarantee (or Postconditions)	1. Admin can add or remove another admin by providing the same access privileges. 2. System should allow new admin to access the system with administrative privileges. 3. System should delete the record when admin performs delete operation on specific admin.
Main Success Scenario	1. The administrator selects ‘add admin’. 2. The system displays blank admin form. 3. The administrator enters the new administrator information. This includes: 1. Name: 2. Employee Type: 3. Mailing address: 4. Phone Number: 4. The Administrator now selects “add admin” once all the information has been entered. 5. The system generates and assigns a unique employee Id number to the admin with admin

	level of access privilege 6. New Admin is added to the system.
Extensions	1a. Administrator is not able to add a new admin form by clicking on ‘add admin’ activity. 3a. System does not allow administrator to enter the new admin information in the form. 5a. System does not generate unique employee Id even after adding all the information into the system.

16. Use CasePB13(i): Add admin or employee (As an admin)

Use Case	Comments
Scope	On Campus Food Ordering System
Level	Admin Goal
Primary Actor	Admin and the System
Stakeholders and Interests	Admin: Wants to add another admin for the application to monitor orders and sales of the cafe.
Preconditions	1. Admin must be an authorized user. 2. Admin must be logged into the system.
Success Guarantee (or Postconditions)	1. Admin can add or remove another admin by providing the same access privileges. 2. System should allow new admin to access the system with administrative privileges. 3. System should delete the record when admin performs delete operation on specific admin.
Main Success Scenario	1. The administrator selects ‘add admin’. 2. The system displays blank admin form. 3. The administrator enters the new administrator information. This includes: 1. Name: 2. Employee Type: 3. Mailing address: 4. Phone Number:

	4. The Administrator now selects “add admin” once all the information has been entered. 5. The system generates and assigns a unique employee Id number to the admin with admin level of access privilege 6. New Admin is added to the system.
Extensions	1a. Administrator is not able to add a new admin form by clicking on ‘add admin’ activity. 3a. System does not allow administrator to enter the new admin information in the form. 5a. System does not generate unique employee Id even after adding all the information into the system.
Alternate scenario	The administrator will be able to add the cafe employee to the system.

17. Use CasePB13(ii): Delete admin (As an admin)

Use Case	Comments
Scope	On Campus Food Ordering System
Level	Admin Goal
Primary Actor	Admin and the System
Stakeholders and Interests	Admin: Wants to delete existing admin by revoking their access privilege.
Preconditions	1. Admin must be an authorized user. 2. Admin must be logged into the system.
Success Guarantee (or Postconditions)	1. The administrator selects ‘delete admin’ activity from administrators view of the application. 2. Administrator enters Employee Id. 3. System locates the Admin with Employee Id 4. The system prompts to verify the details. 5. Administrator verifies the deletion. 6. The system saves this operation of the record deletion.
Extensions	1a. Administrator is not able to delete the admin by clicking ‘delete admin’ activity.

	2a. System does not allow administrator to enter the employee id of the admin. 6a. Admin is able to view the deleted record information even though it is deleted from the front end.
Alternate Scenario	The admin will be able to remove the café employee from the system

18. Use casePB3,4: Login/Register for customer

Use Case	Comments
Scope	On Campus Food Ordering System
Level	User Goal
Primary Actor	User and the System
Stakeholders and Interests	User: Wants to register and login to the system to make an online order
Preconditions	1. The user should be on the landing page
Success Guarantee (or Postconditions)	1. The user registers with valid credentials
Main Success Scenario	The first-time user can register and login successfully.
Extensions	1a. The details provided by the user is not accepted by the system 1b. The login credentials provided by the user is not correct

3.2. Class Diagram

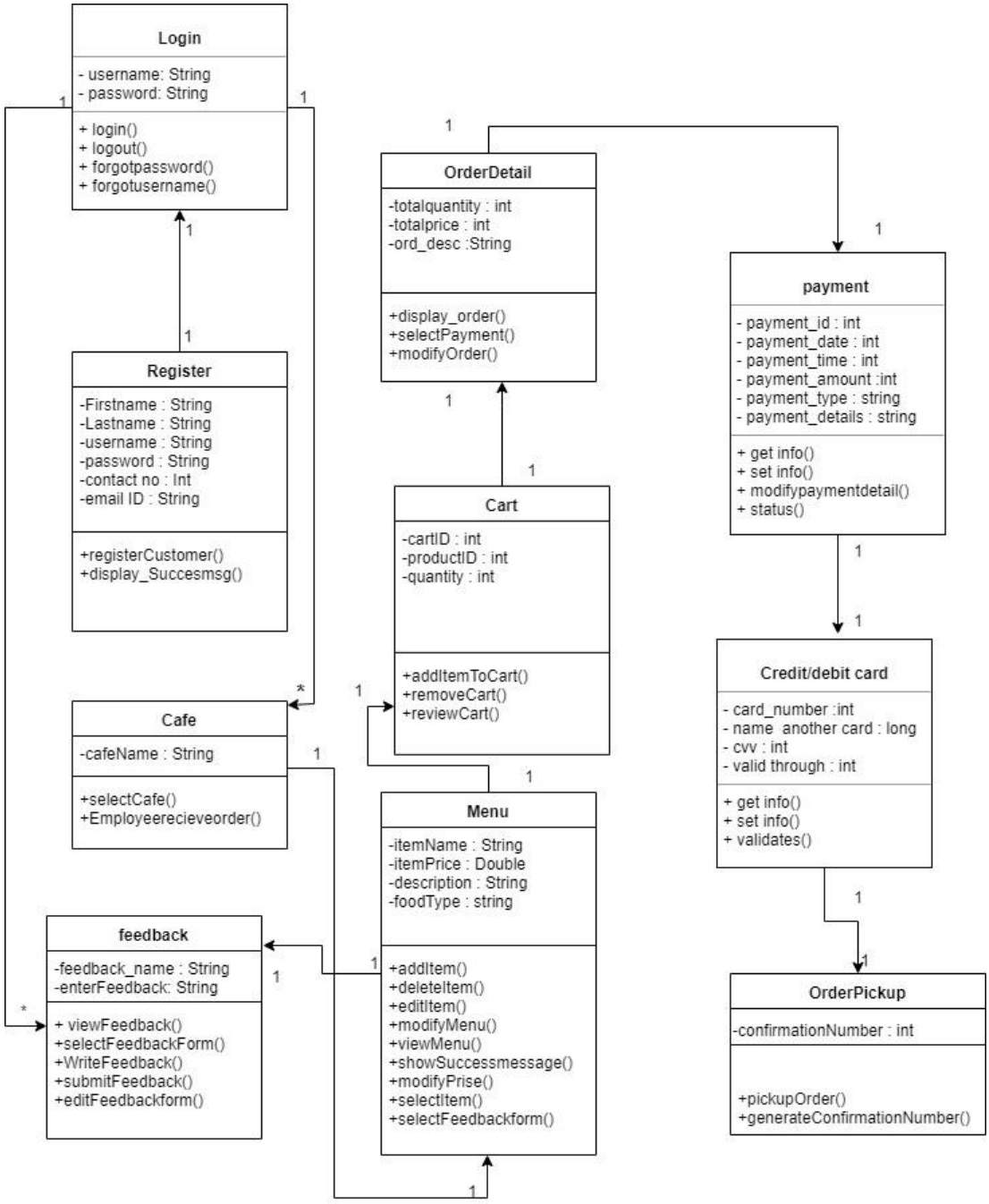


Fig 1: Class diagram for Customer

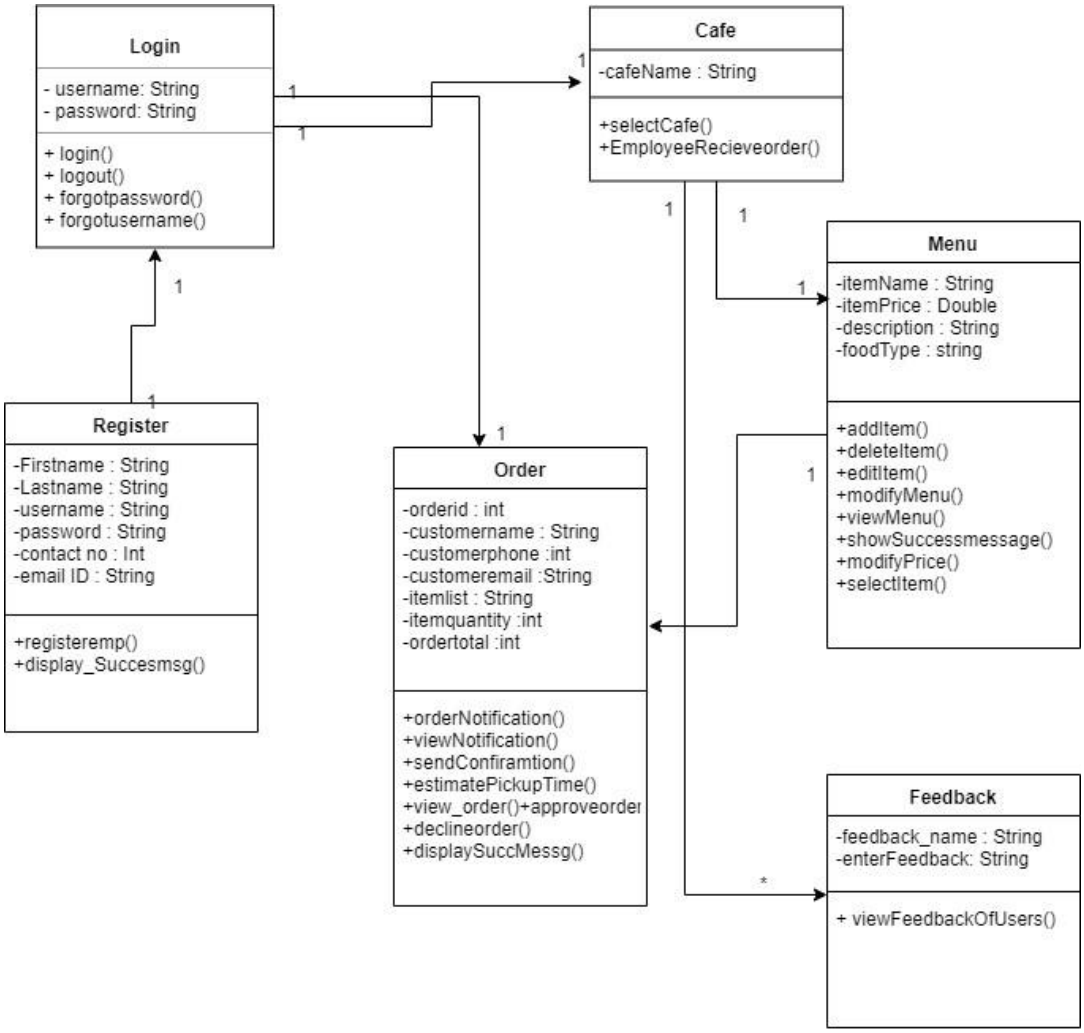


Fig_3:Class Diagram for Employee

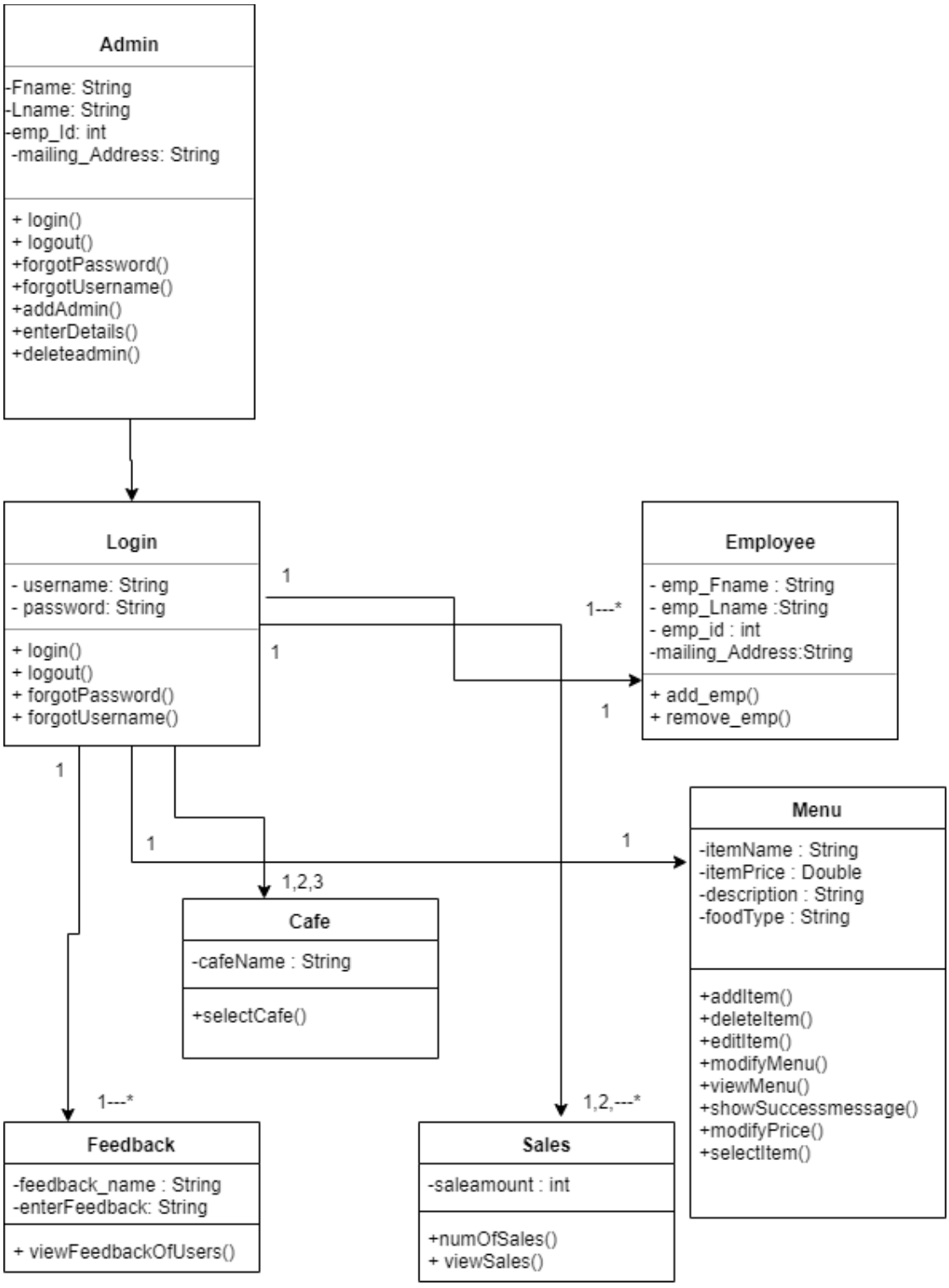
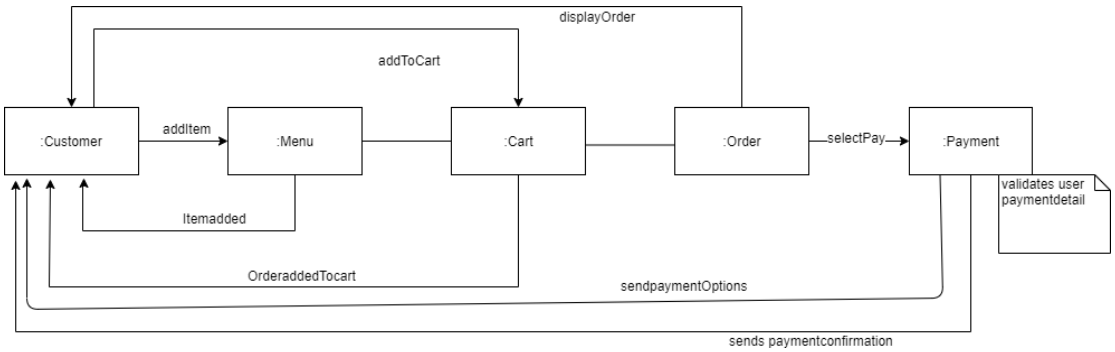


Fig 2: Class diagram for Admin

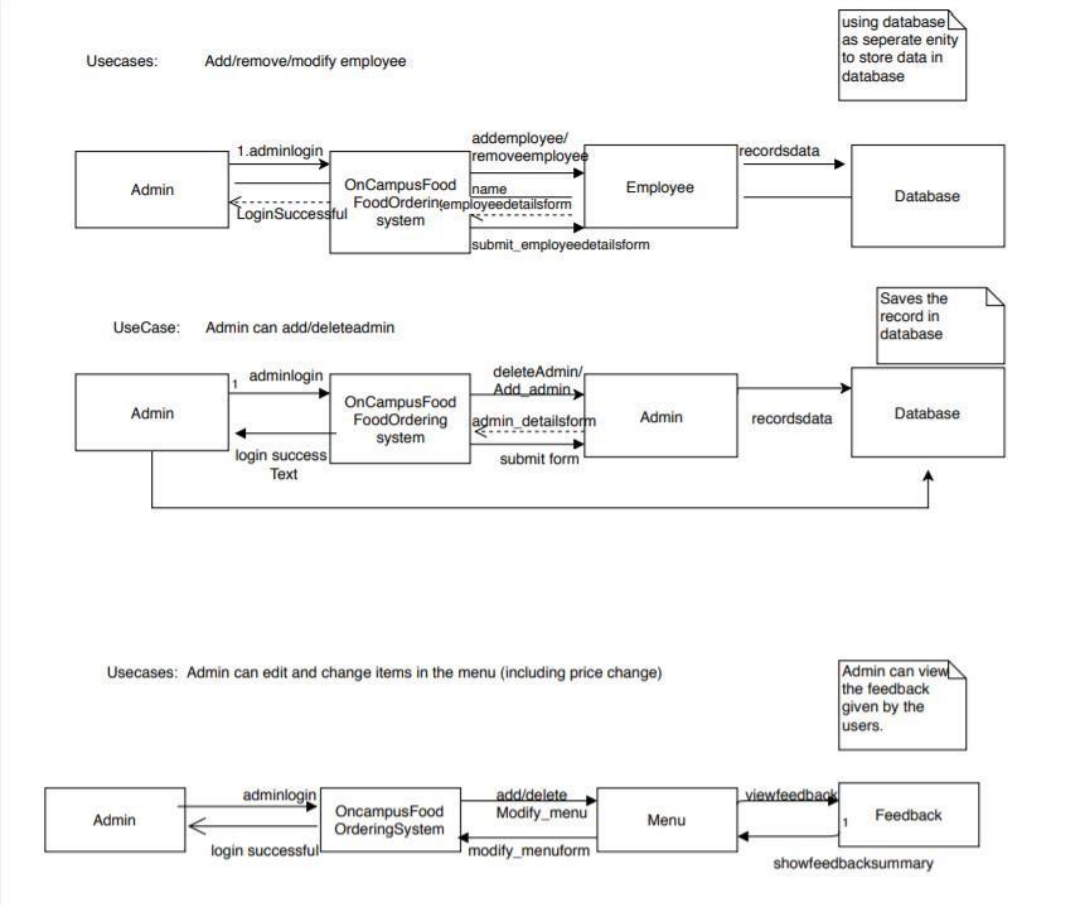
3.3. Interaction Diagram

3.3.1 Communication Diagram

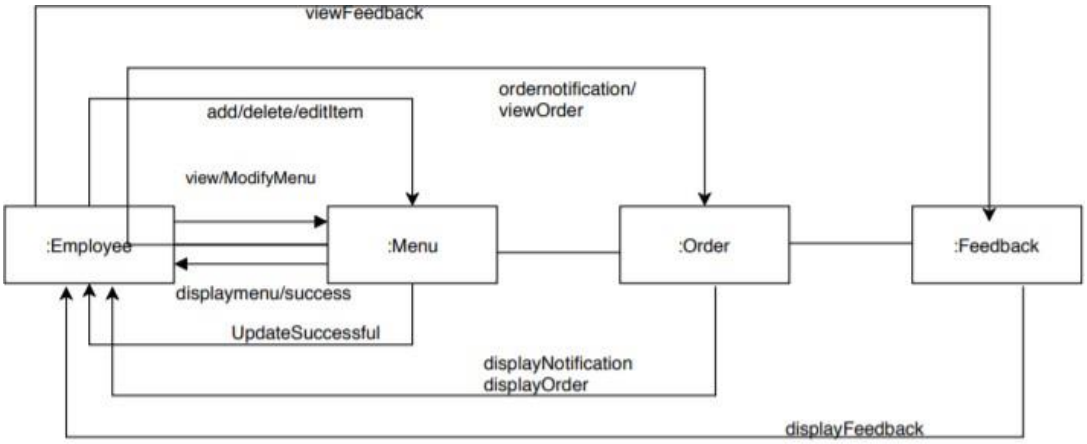
a. Customer



b. Admin



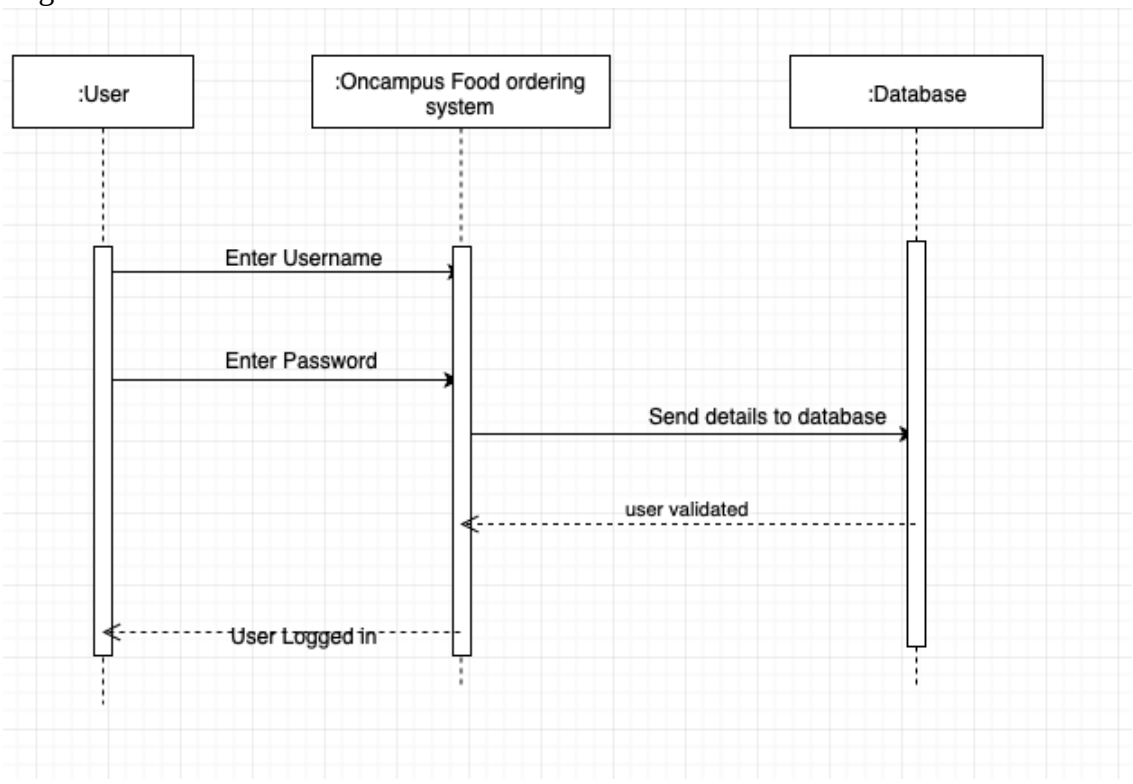
c. Employee



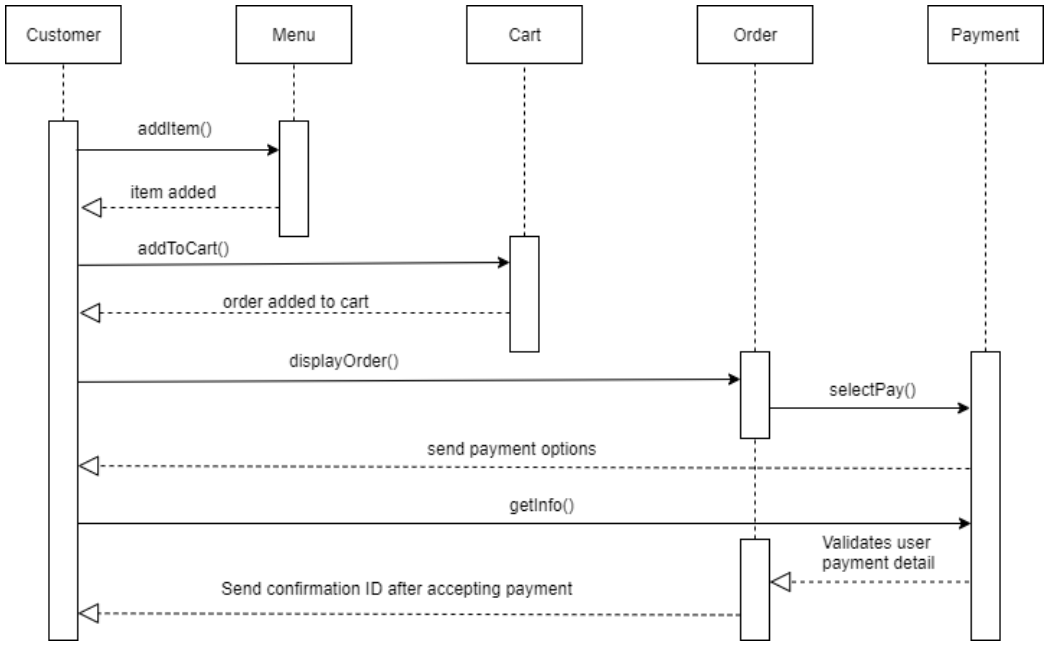
Communication Diagram for Employee (All usecases)

3.3.2 Interaction Diagram

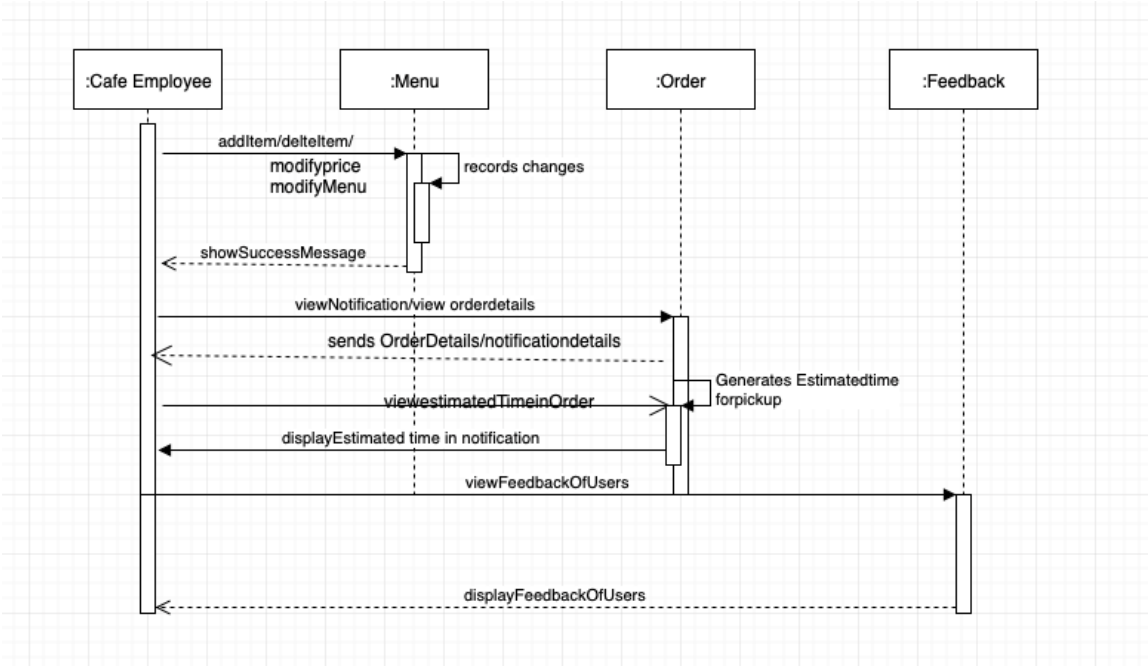
a. Login



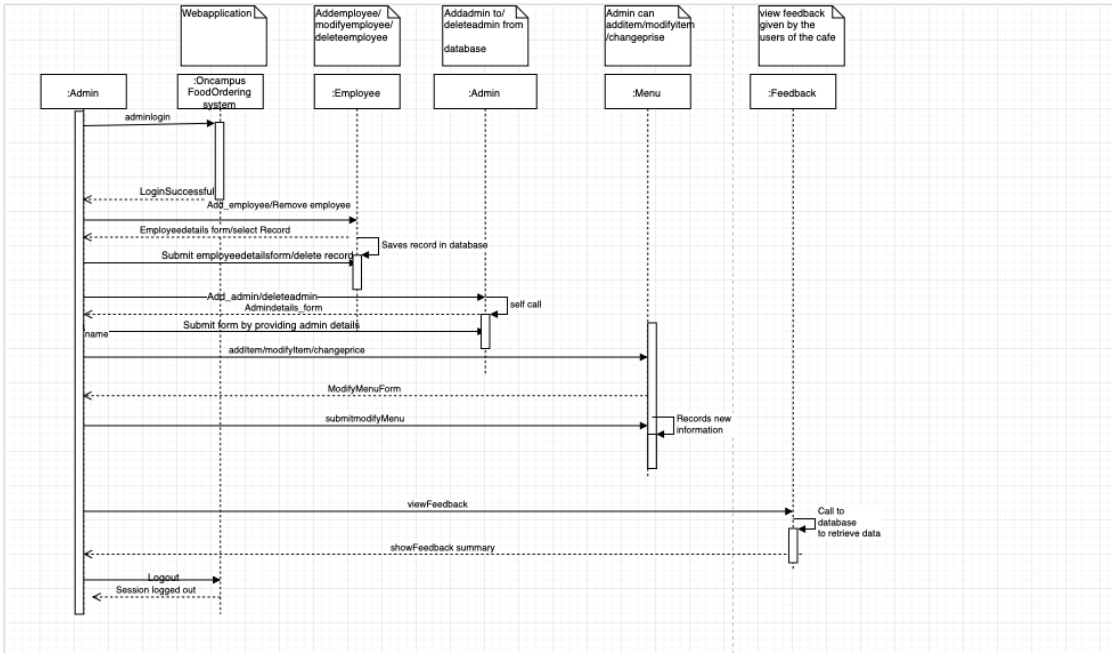
b. Customer



c. Employee

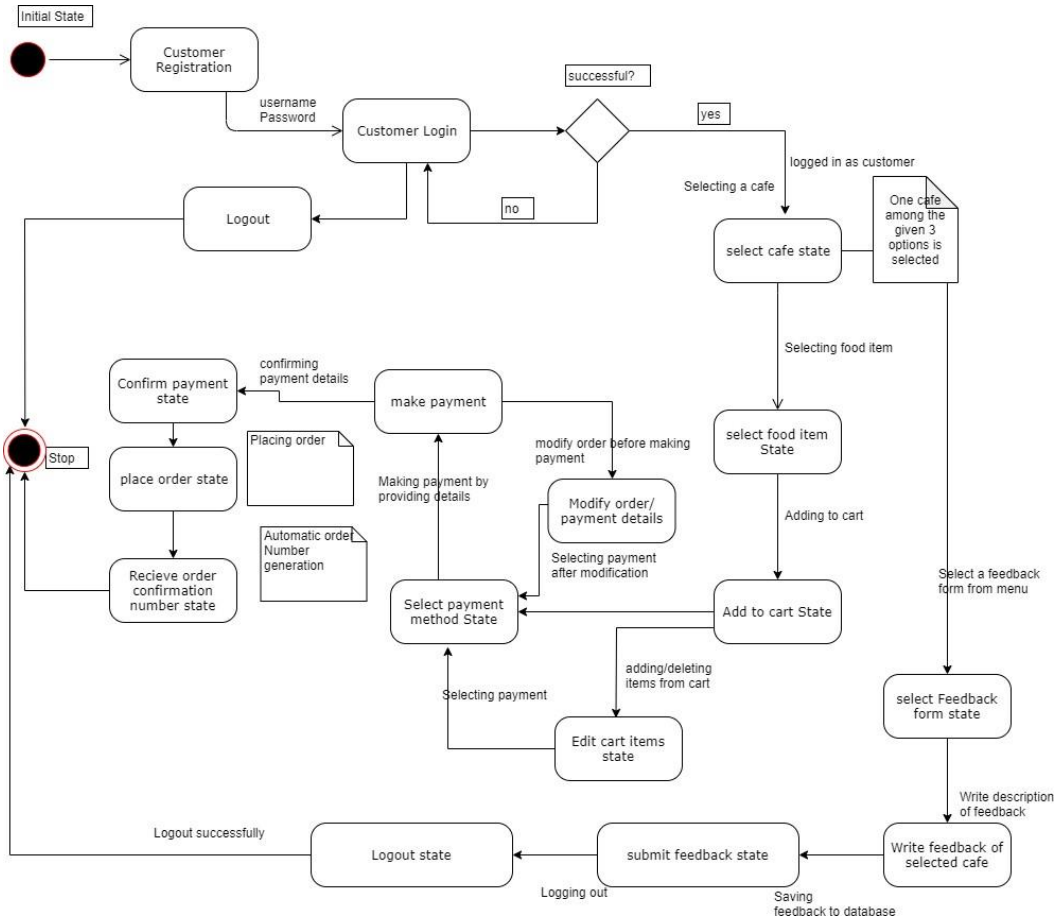


d. Admin

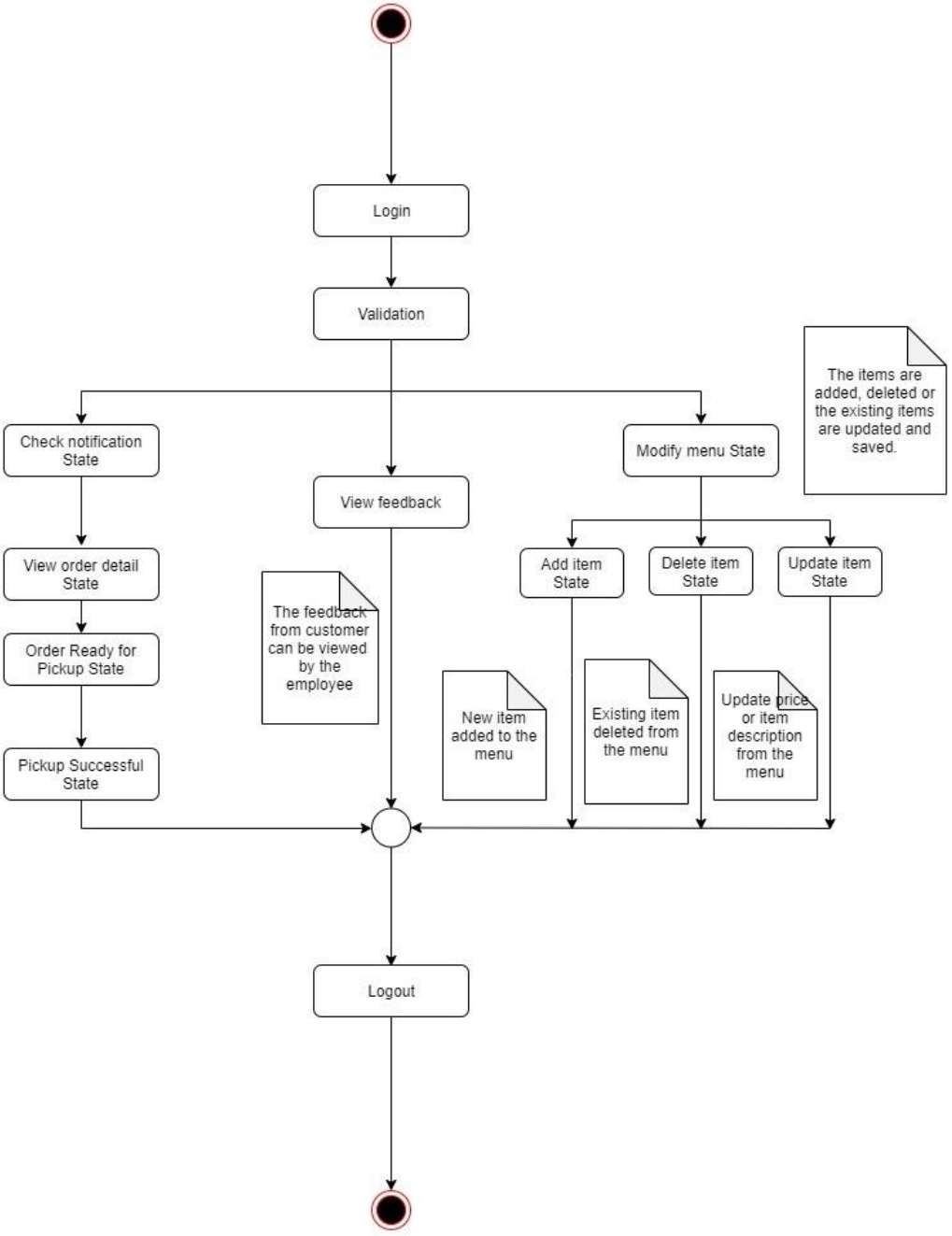

Webapplication
Addemployee/ modifyemployee/ deleteemployee
Addadmin to/ deleteadmin from database
Admin can additem/modifyitem /changeprise
view feedback given by the users of the cafe
:Admin
:Oncampus FoodOrdering system
:Employee
:Admin
:Menu
:Feedback
adminlogin
LoginSuccessful
Add_employee/Remove employee
Employeedetails form/select Record
Saves record in database
Submit employeedetailsform/delete record
Add_admin/deleteadmin
Admindetails_form
self call
Submit form by providing admin details
name
addItem/modifyItem/changeprice
ModifyMenuForm
submitmodifyMenu
Records new information
viewFeedback
Call to database to retrieve data
showFeedback summary
Logout
Session logged out

3.4. State Diagrams

3.4.1. Customer

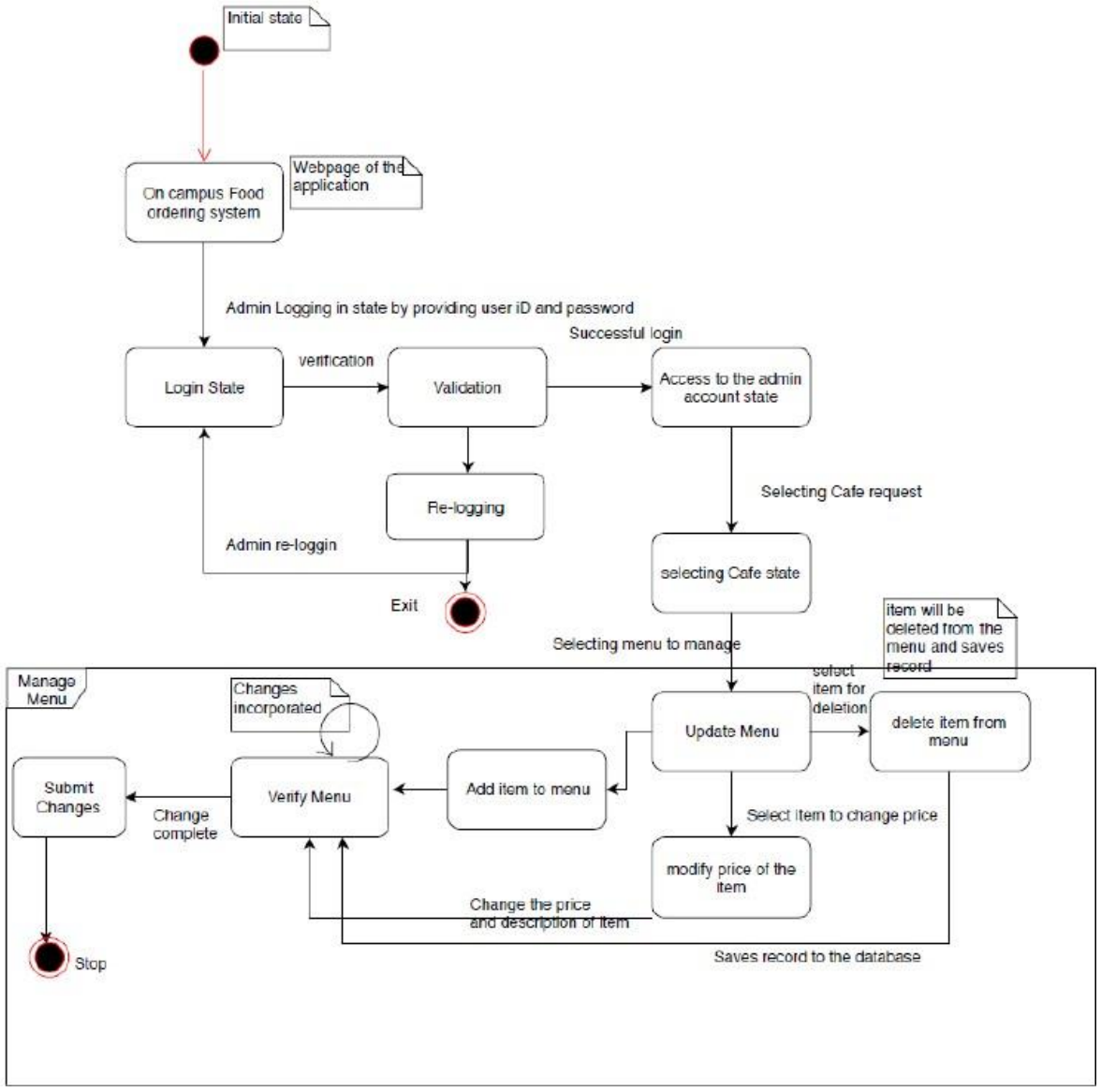


3.4.2. Employee

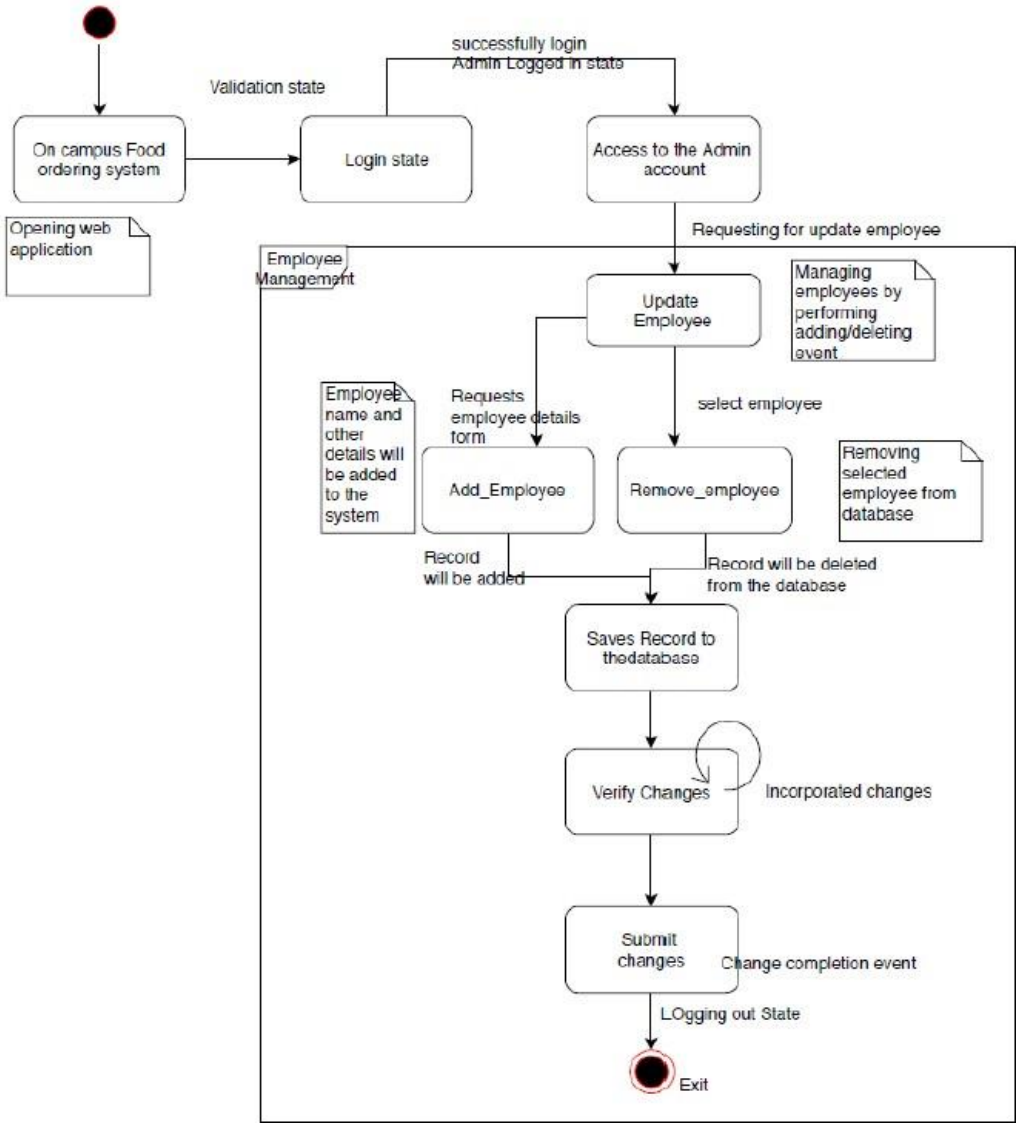


3.4.3. Admin

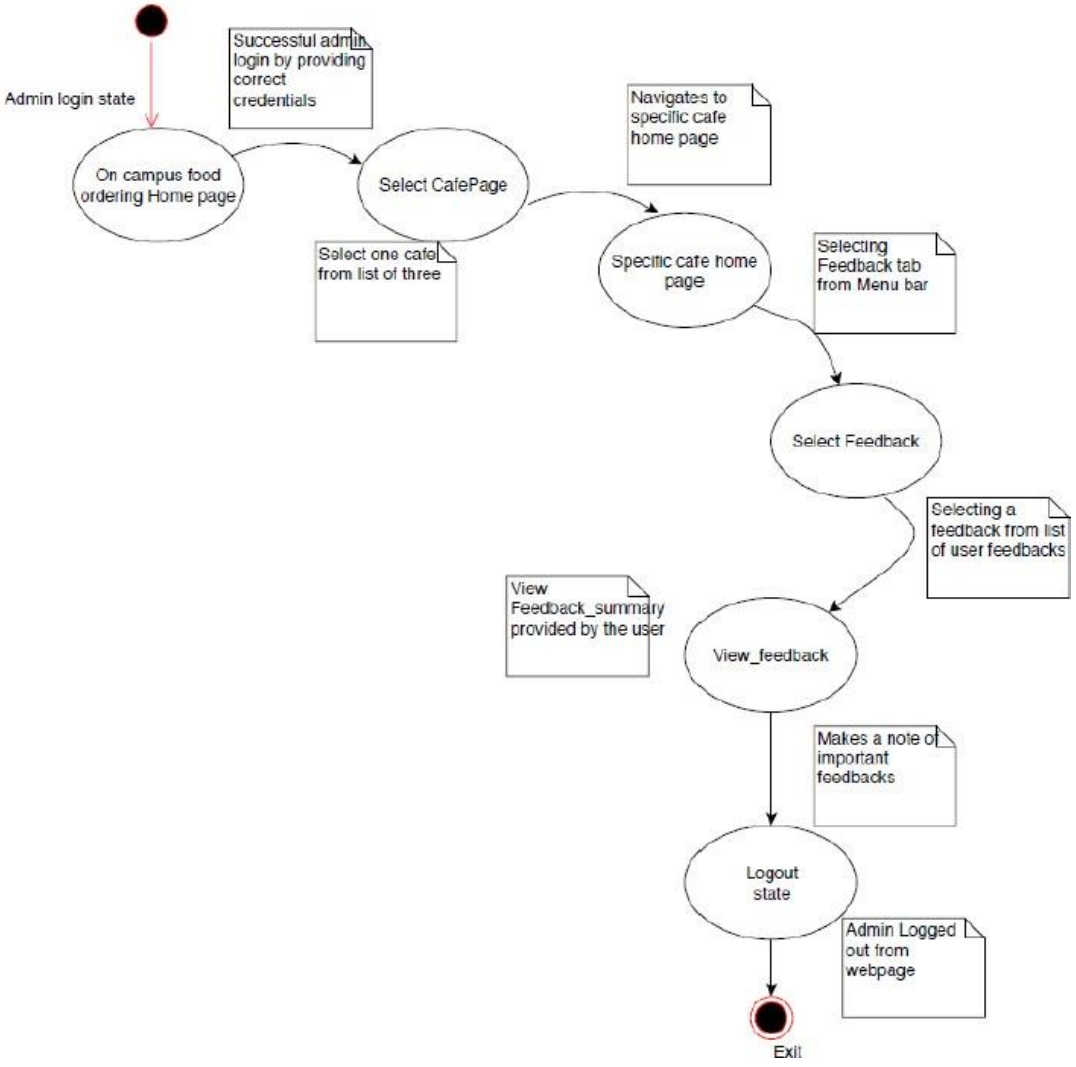
i) Use Case: Admin can edit and change items in the menu (including price change)



ii) Use Cases: Admin can Add/ Remove Cafe employee from the system

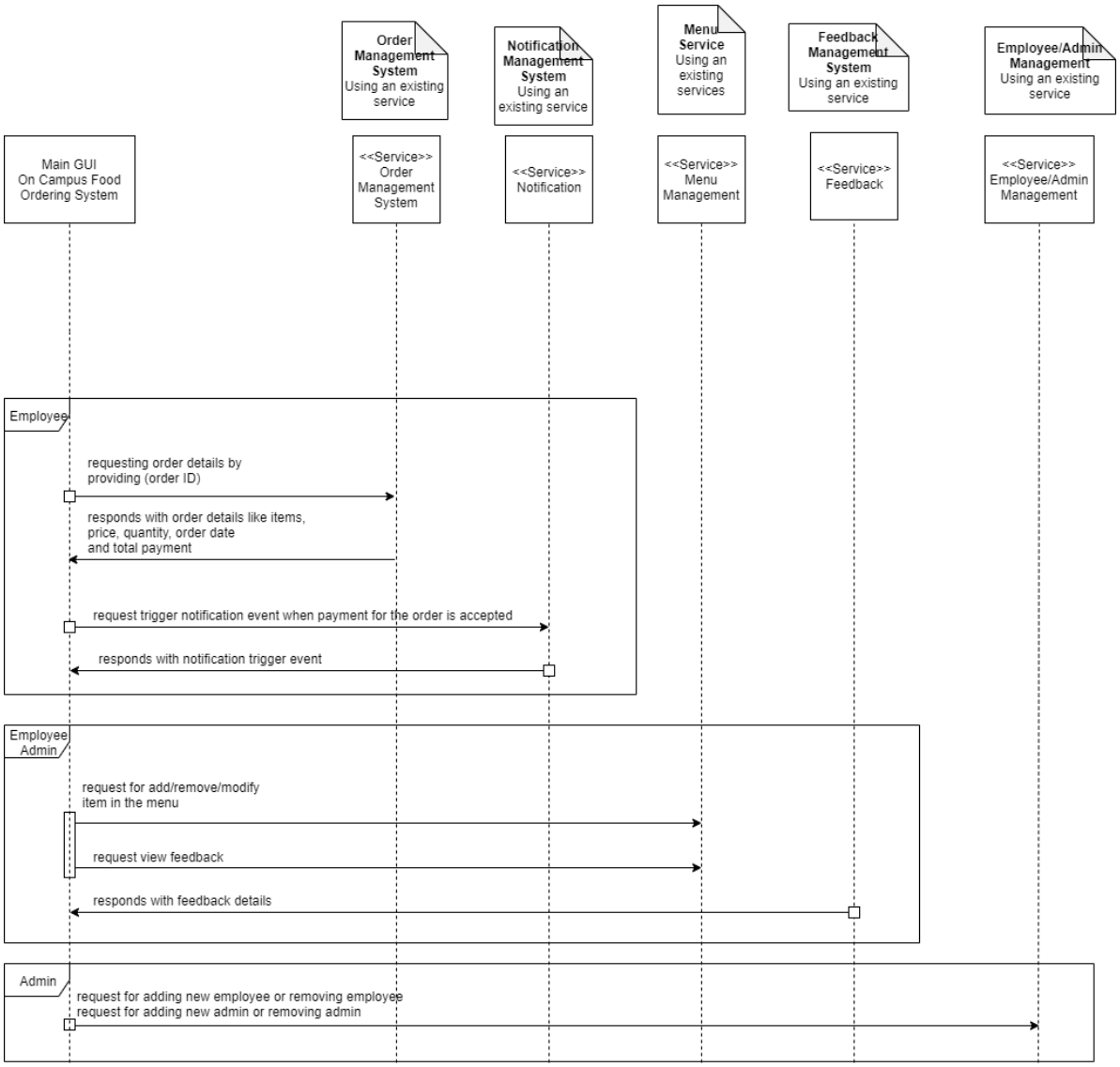


iii) Use Case: Admin can view the Feedback Provided by the customers of café

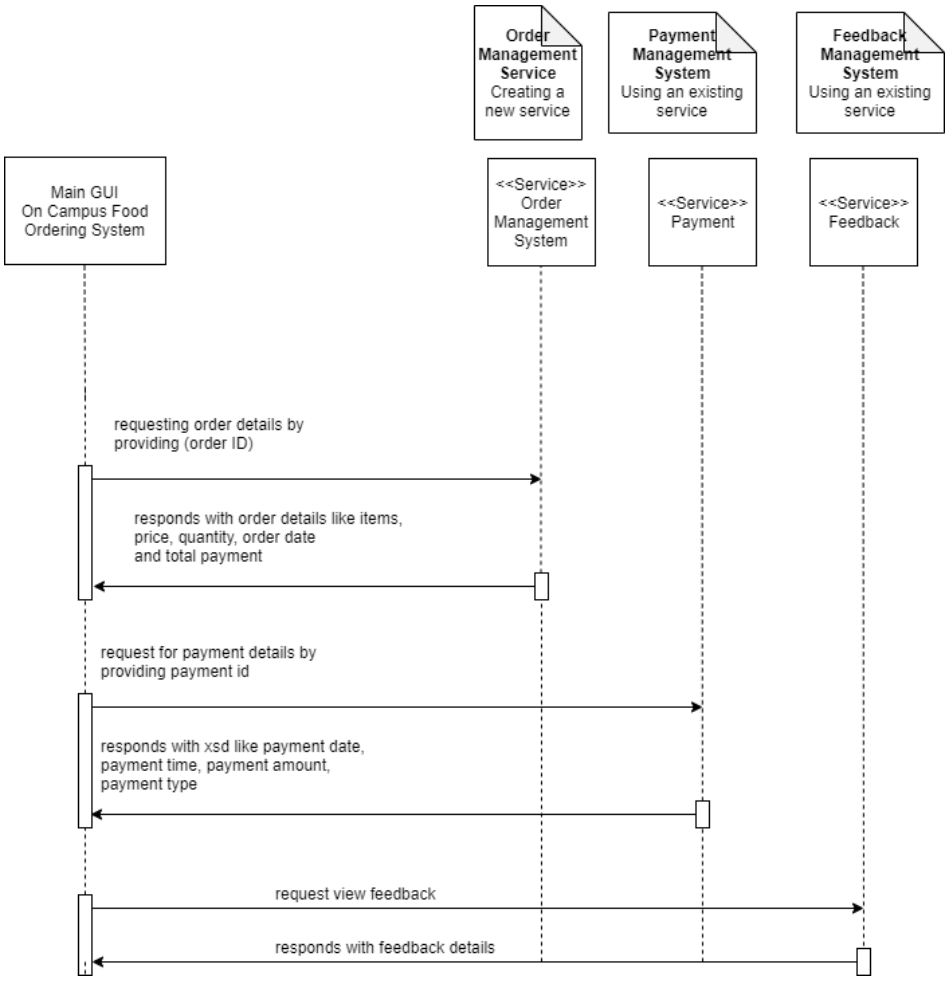


3.5. SOA

3.5.1 Employee and Admin



3.5.2 Customer



3.6 GRASP PATTERN

Acronym for **General Responsibility Assignment Software Patterns is GRASP.**

A **pattern** is a recurring solution to standard problem in a context. As per Christopher Alexander, professor of architecture "A pattern describes a problem which occurs over and over again in our environment, and then describes the core solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice".

Object Oriented pattern is not a finished design that can be converted/ transformed directly into code. It is considered as a template for how to solve a problem that can be reused in many other situations. IT shows the interactions and relationships between the classes or objects which are involved in that context.

We are using LOW COUPLING GRASP pattern to show how the objects are coupled in our project design. We chose this pattern as it is very Important in making decisions and evaluating designs while working on a project.

Definition of Low Coupling: Coupling is a measure of how strongly one object or element is connected to, dependent on other elements (One object may dependent on only one or many other objects).

An object/class with low coupling means it is not dependent on too many other objects/classes.

A class with strong coupling means it relies on many other classes. which leads to many other problems which are listed below.

Problems with HIGH COUPLING:

1.We need to do forced local changes because of changes in related classes or objects.

2.We cannot reuse the independent components or modules of code as it also requires additional classes in which it is dependent (Interdependencies between many classes).

3.Very difficult to understand the design in isolation.

Solution:

We can overcome above problems by not assigning too many responsibilities to the controller class.

By allowing the controller class to delegate task.

By creating multiple controller classes to create less dependencies between objects.

Benefits of Low coupling:

1. Easier to understand the code.
2. Reusable codes.
3. Minimization of localized changes.

Contradiction:

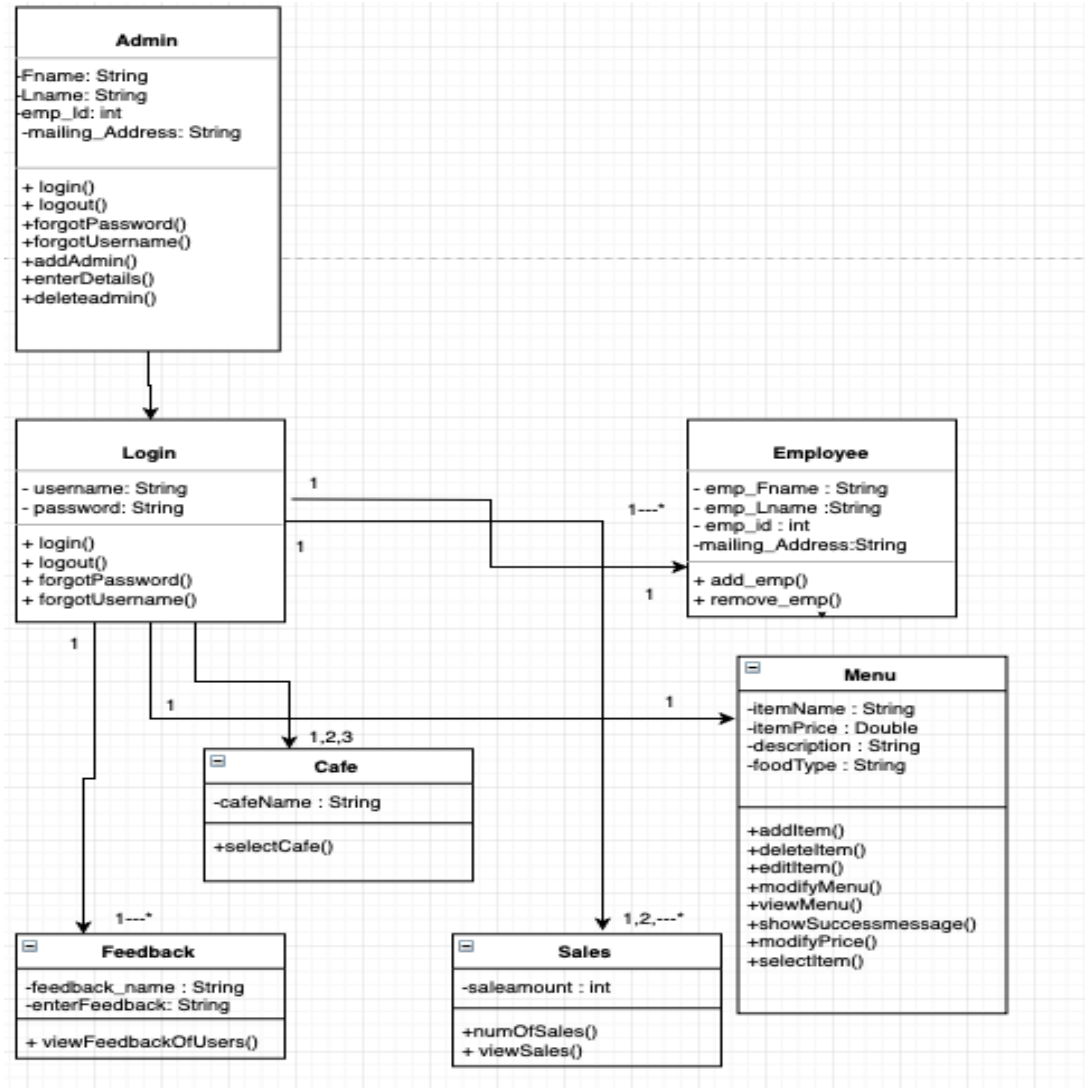
Extremely low coupling may lead to poor design.

Implementation of Low coupling design pattern in our project:

Low Coupling Example Scenario:

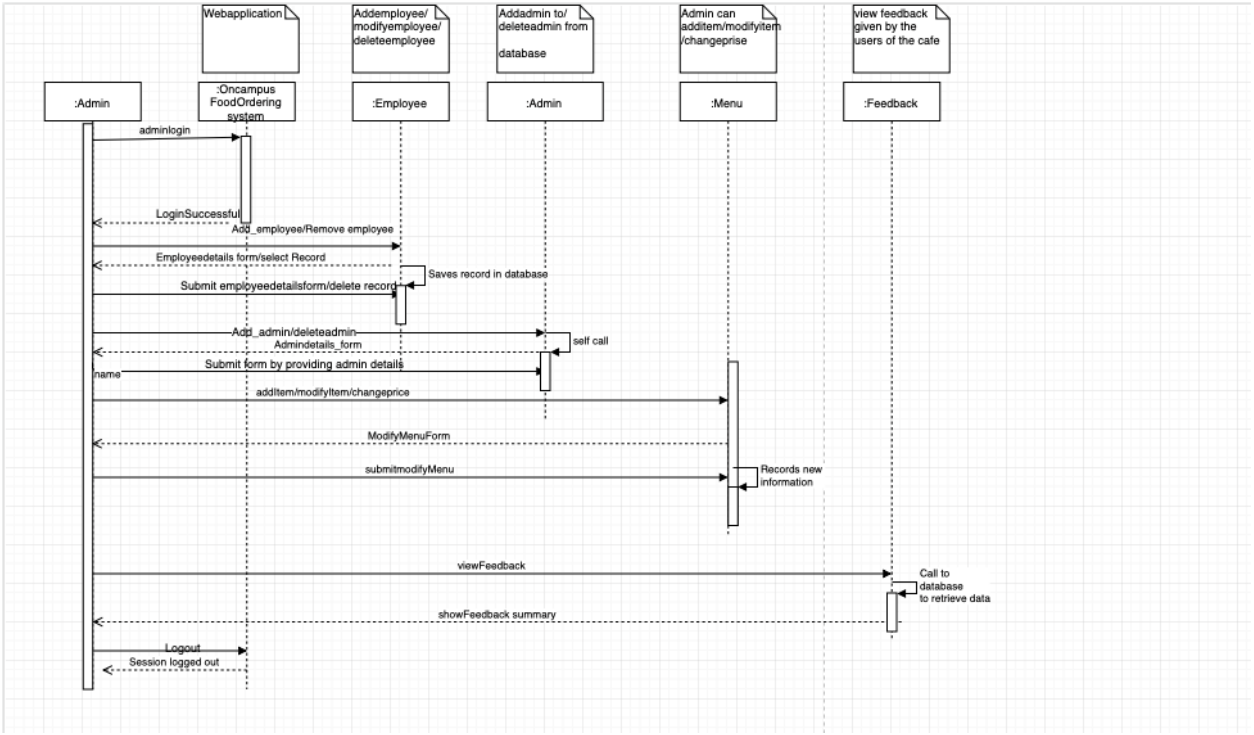
Below class diagram represents the Low coupling Grasp pattern for on campus food ordering system.

Admin managing employee, manager and feedback. There exists only few important associations between the classes which are necessary.



High Coupling class diagram consists of Many associations from One class to many classes(Which shows lot of dependency on multiple or on single module). Which can be addressed by low coupling design as shown in above class diagram.

Sequence for Low coupling:



High Coupling Design Pattern VS Low coupling design pattern:

High coupling Design Pattern:

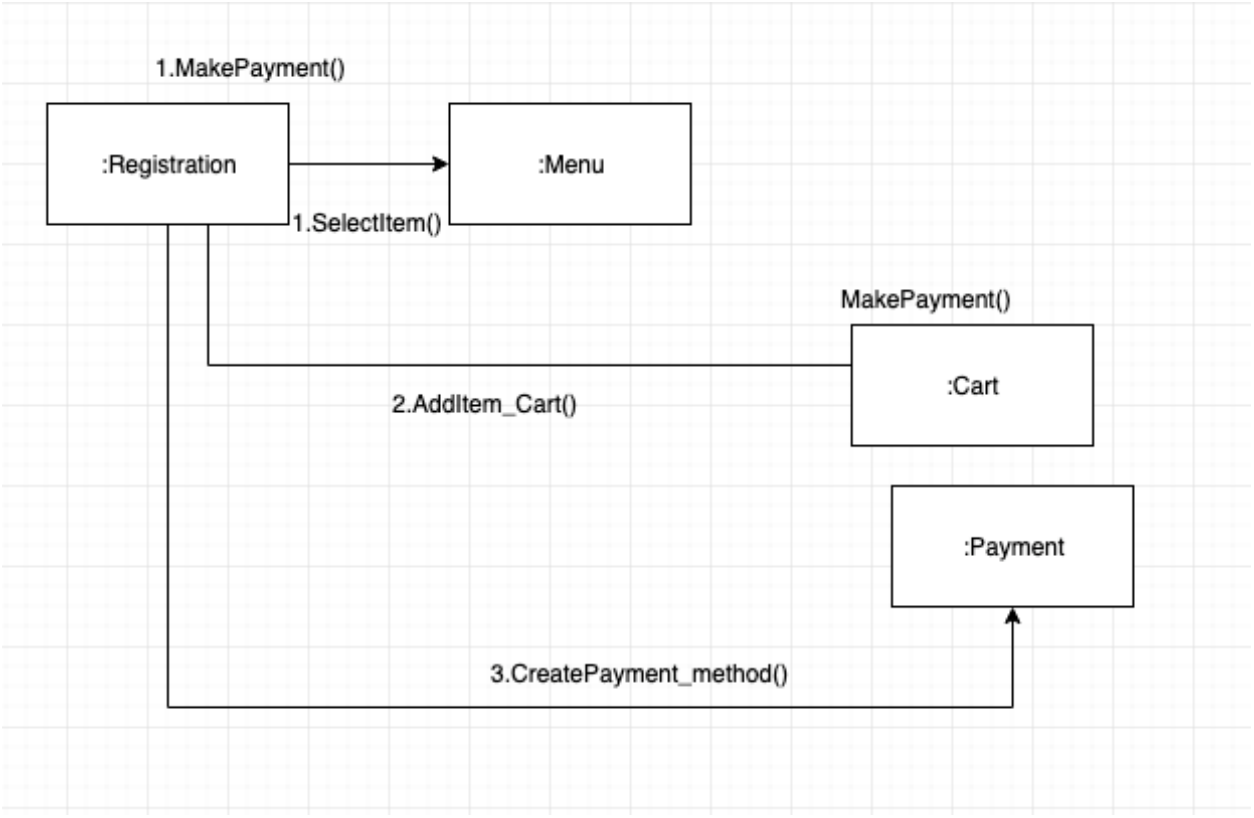
Let us consider the below scenario of make payment () we need to go through 3 steps through Registration.

1.Registration-> Menu

2.Registration->Cart

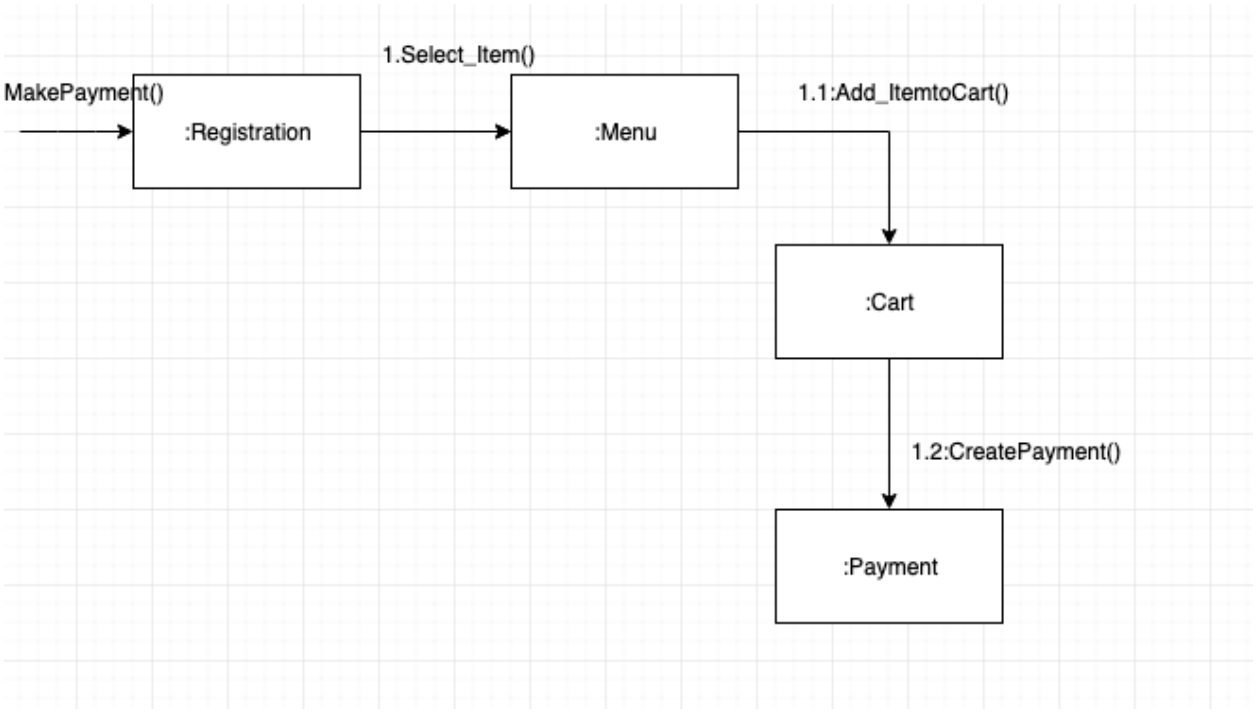
3.Registration->Payment.

In this High coupling scenario if there is any change in the registration class all the local variables in the remaining classes need to be changed accordingly. In this scenario we visited registration class 3 times to make payment. There exists a lot of dependency between the classes.



Low coupling scenario with less dependency.

We just need to register once to make payment then the steps are sequential. Only once we are visiting the registration class. To avoid this problem, we can use low coupling scenario. Below diagram explains the Low coupling scenario.



Benefits

· Maintainability -Little or not affected by changes in other components
· Understandability- simple to understand in the isolation and Minimization of localized changes
· Reusability and modularity- convenient to reuse and easier to grab hold of classes

Liabilities / Contradictions:

· Extremely low coupling may lead to poor design.

3.7 GOF Pattern

A design pattern in software engineering is a general solution to a common problem in software design. A pattern of design is not a finished design that can be directly translated into code. It is a summary or model for addressing a problem that can be used in several different situations.
The book has been named "The Gang of Four" by the four Authors: Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. There are 23 Design patterns described in short form as GOF.

Uses of Design Patterns

Design patterns can speed up the development process by providing tested, proven development paradigms. Effective software design requires considering issues that may not become visible until later in the implementation. Reusing design patterns helps to prevent subtle issues that can cause major problems and improves code readability for coders and architects familiar with the patterns.

These design patterns are classified into three types:

CREATIONAL PATTERN:
The first type of design pattern is the *creational* pattern. Creational patterns provide ways to instantiate single objects or groups of related objects. There are five such patterns:

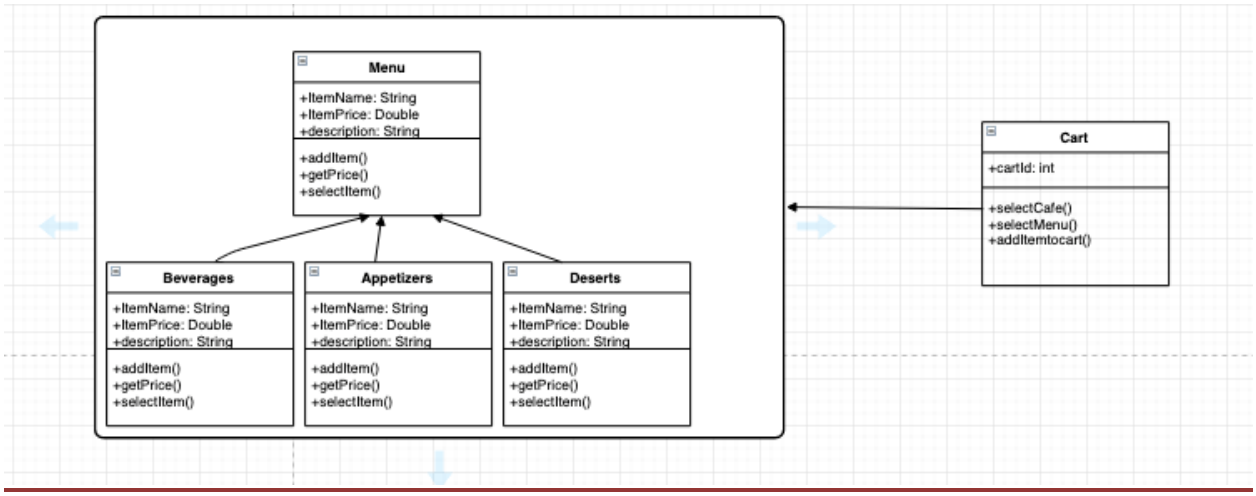
1. **Abstract Factory:** The abstract factory pattern is used to provide a client with a set of related or dependent objects. The "family" of objects created by the factory are determined at run-time.
2. **Builder** : The builder pattern is used to create complex objects with constituent parts that must be created in the same order or using a specific algorithm. An external class controls the construction algorithm.
3. **Factory Method** : The factory pattern is used to replace class constructors, abstracting the process of object generation so that the type of the object instantiated can be determined at run-time.
4. **Prototype** :The prototype pattern is used to instantiate a new object by copying all of the properties of an existing object, creating an independent clone. This practice is particularly useful when the construction of a new object is inefficient.
5. **Singleton** :The singleton pattern ensures that only one object of a particular class is ever created. All further references to objects of the singleton class refer to the same underlying instance.

Factory Pattern:

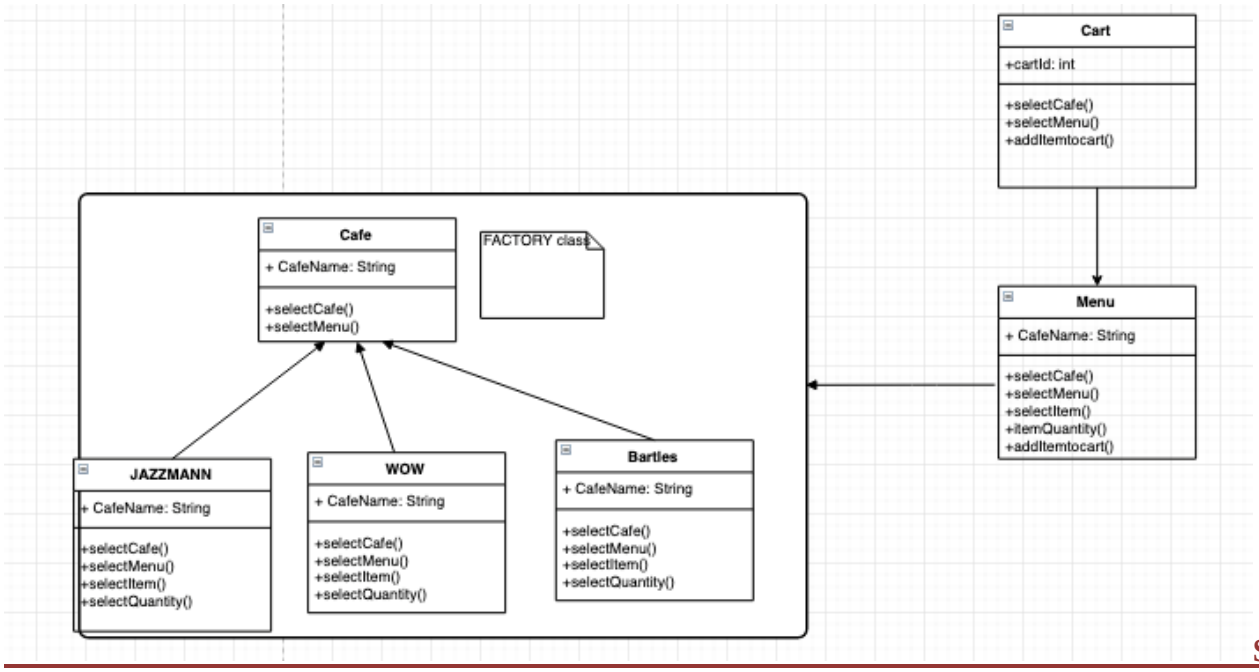
Factory pattern is one of the most used design patterns in Java. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

In Factory pattern, we create object without exposing the creation logic to the client and refer to newly created object using a common interface.

Example Menu Type:
Factory Cart, our Cart class will use *Menu Factory* to get Type of Menu. It will pass information (*Beverage / appetizer/ Beverage*) to Cart *Factory* to get the type of object it needs.



Similar Example2:



Patterns

The second type of design pattern is the *structural* pattern. Structural patterns provide a manner to define relationships between classes or objects.

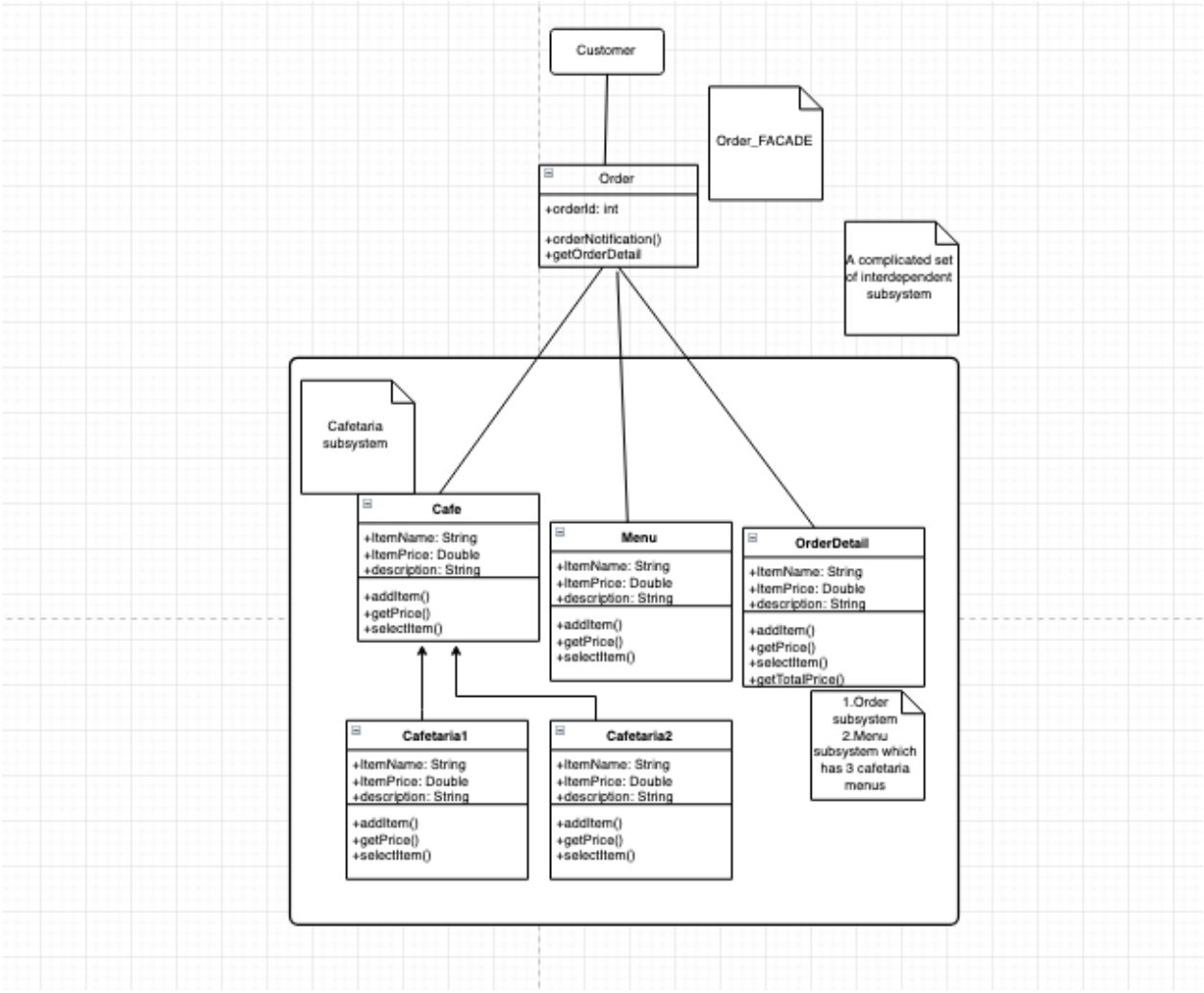
- **Adapter** : The adapter pattern is used to provide a link between two otherwise incompatible types by wrapping the "adaptee" with a class that supports the interface required by the client.
- **Bridge** : The bridge pattern is used to separate the abstract elements of a class from the implementation details, providing the means to replace the implementation details without modifying the abstraction.
- **Composite** : The composite pattern is used to create hierarchical, recursive tree structures of related objects where any element of the structure may be accessed and utilized in a standard manner.
- **Decorator** : The decorator pattern is used to extend or alter the functionality of objects at run-time by wrapping them in an object of a decorator class. This provides a flexible alternative to using inheritance to modify behavior.
- **Facade** : The facade pattern is used to define a simplified interface to a more complex subsystem.
- **Flyweight** : The flyweig000 ht pattern is used to reduce the memory and resource usage for complex models containing many hundreds, thousands or hundreds of thousands of similar objects.
- **Proxy** :The proxy pattern is used to provide a surrogate or placeholder object, which references an underlying object. The proxy provides the same public interface as the underlying subject class, adding a level of indirection by accepting requests from a client object and passing these to the real subject object as necessary.

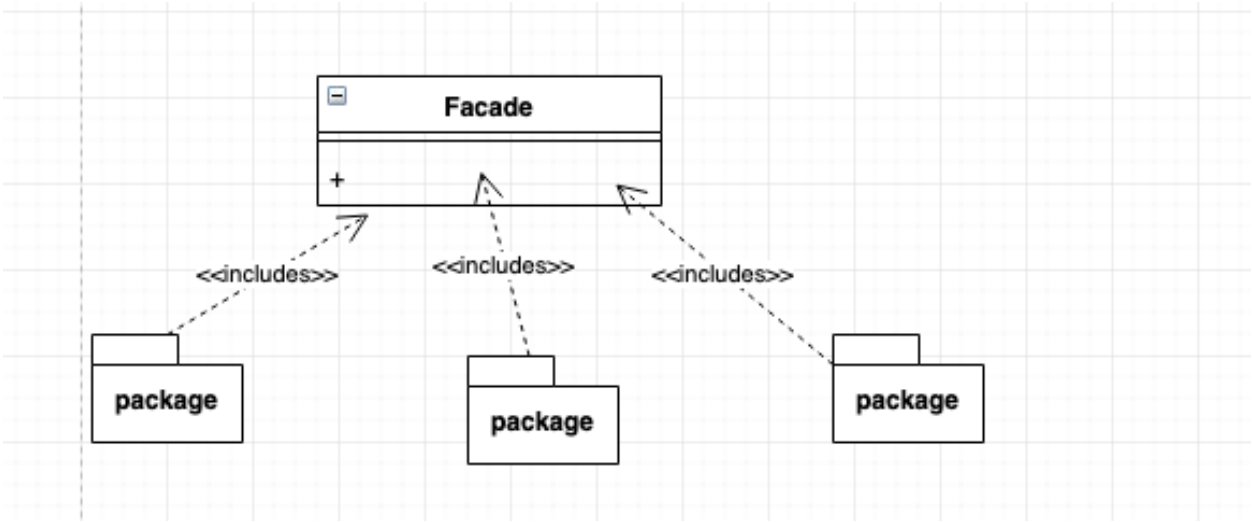
1.1.1 The Facade Pattern

Like the Adapter pattern, Facade is known as a **structural** pattern, as it's used to identifying a simple way to realize relationships between entities. The definition of Facade provided in the original Gang of Four book.

1.1.2 Where Would I Use This Pattern?

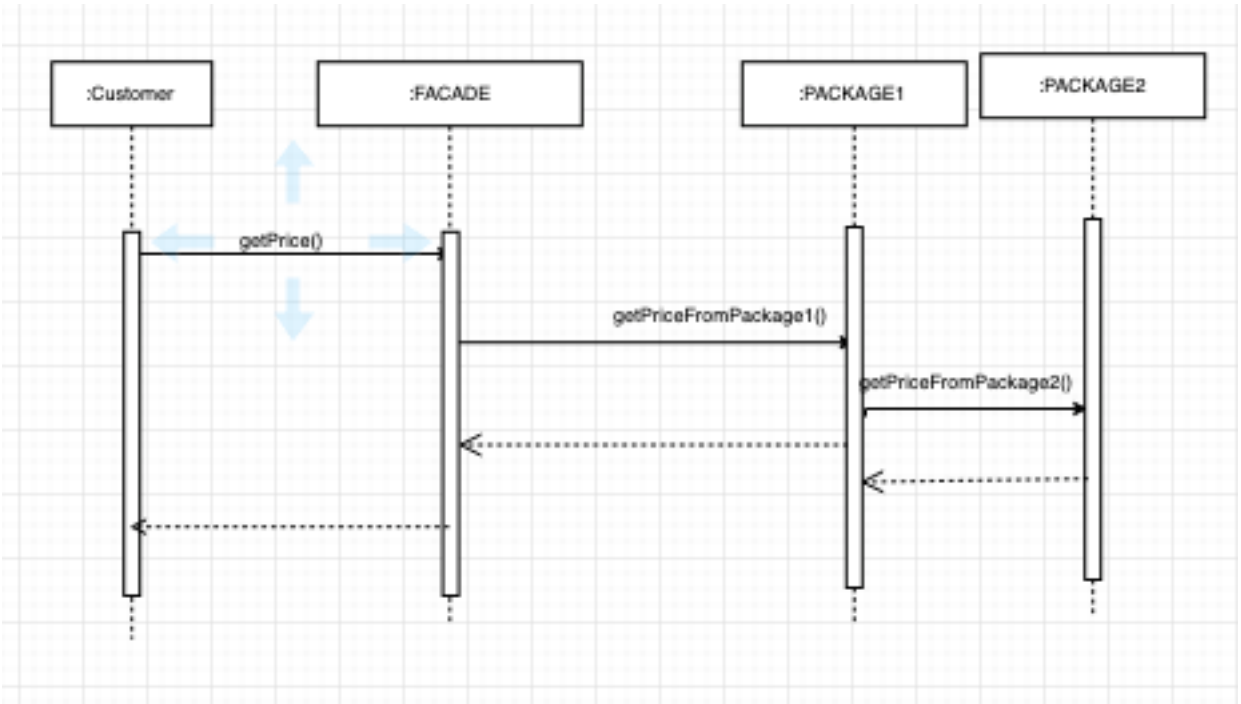
As the concept behind facade is to simplify an interface, service oriented architectures make use of the facade pattern. For example, in web services, one web service might provide access to a number of smaller services that have been hidden from the caller by the facade. Similarly, a typical pattern in OSGi bundles is to provide an interface package that is exposed to users of the bundle. All other packages are hidden from the user.





Like the adapter pattern, the Facade can be used to hide the inner workings of a third party library, or some legacy code. All that the customer needs to do is interact with the Facade, and not the subsystem that it is encompassing.

The following sequence diagram illustrates how the pattern is used by a customer:



1.1.3

1.1.4 Behavioral Patterns

The final type of design pattern is the behavioral pattern. Behavioral patterns define manners of communication between classes and objects.

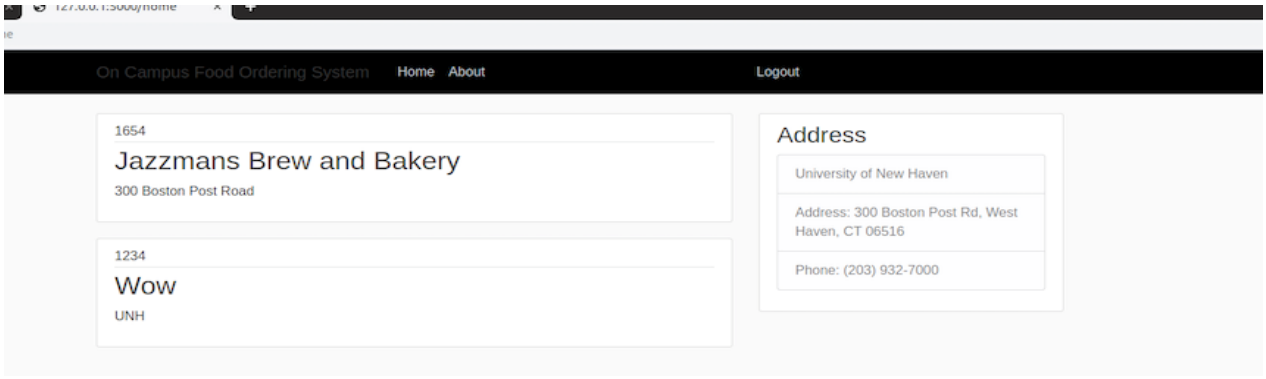
1. **Chain of Responsibility** : The chain of responsibility pattern is used to process varied requests, each of which may be dealt with by a different handler.
2. **Command** :The command pattern is used to express a request, including the call to be made and all of its required parameters, in a command object. The command may then be executed immediately or held for later use.
3. **Interpreter :** The interpreter pattern is used to define the grammar for instructions that form part of a language or notation, whilst allowing the grammar to be easily extended.
4. The **iterator** pattern is used to provide a standard interface for traversing a collection of items in an aggregate object without the need to understand its underlying structure.
5. **Mediator:** The mediator pattern is used to reduce coupling between classes that communicate with each other. Instead of classes communicating directly, and thus requiring knowledge of their implementation, the classes send messages via a mediator object.
6. **Mediator:** The memento pattern is used to capture the current state of an object and store it in such a manner that it can be restored at a later time without breaking the rules of encapsulation.
7. **Observer:** The observer pattern is used to allow an object to publish changes to its state. Other objects subscribe to be immediately notified of any changes.
8. **State :** The state pattern is used to alter the behavior of an object as its internal state changes. The pattern allows the class for an object to apparently change at run-time.
9. **Strategy** : The strategy pattern is used to create an interchangeable family of algorithms from which the required process is chosen at run-time.
10. **Template Method** : The template method pattern is used to define the basic steps of an algorithm and allow the implementation of the individual steps to be changed.
11. **Visitor** : The visitor pattern is used to separate a relatively complex set of structured data classes from the functionality that may be performed upon the data that they hold.

Reference Links:

1. http://www.blackwasp.co.uk/gofpatterns.aspx
2. https://dzone.com/articles/design-patterns-uncovered-1
GOF Patterns

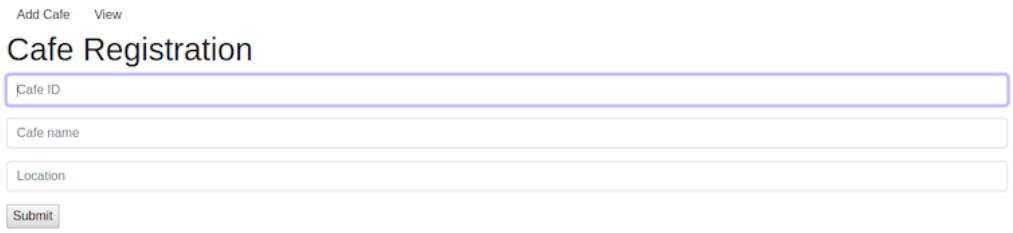
4.User Interface and Functions

4.1. Home page:

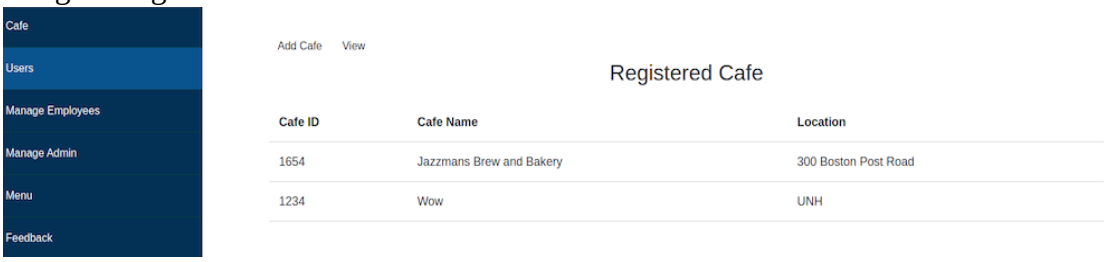


4.2. Café Registration Page:

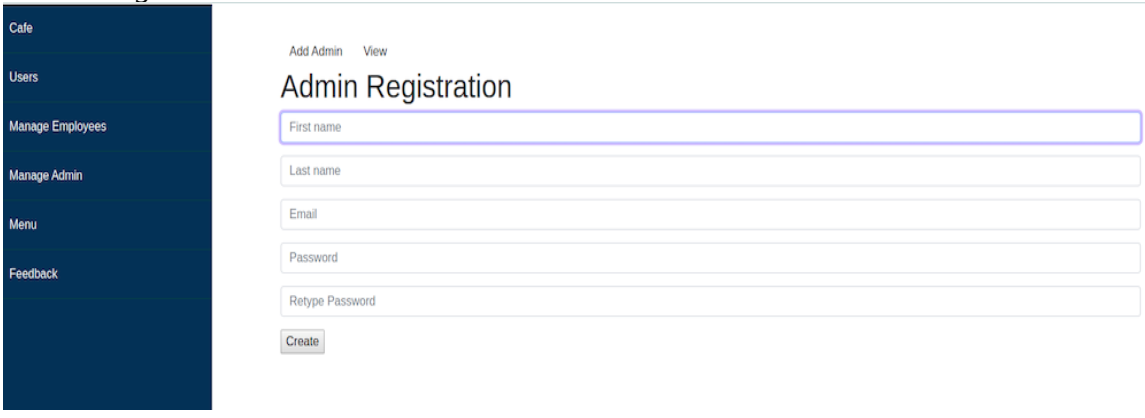
Café Registration by admin:

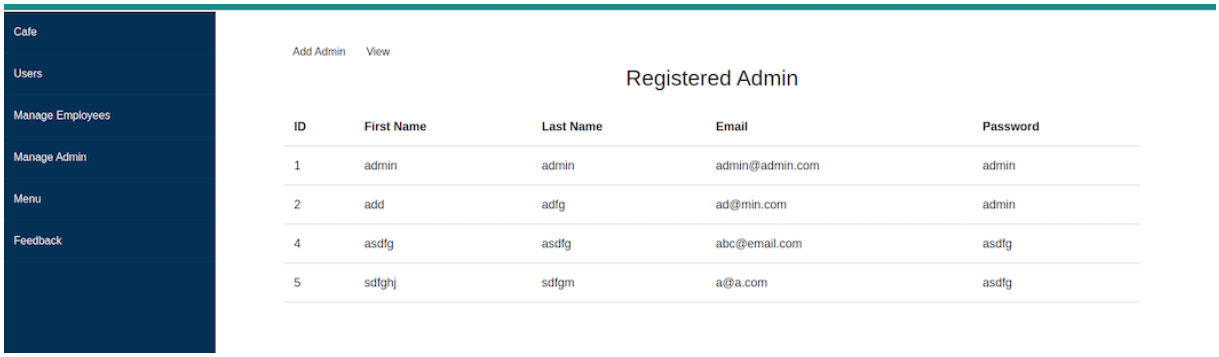


After registering New café:

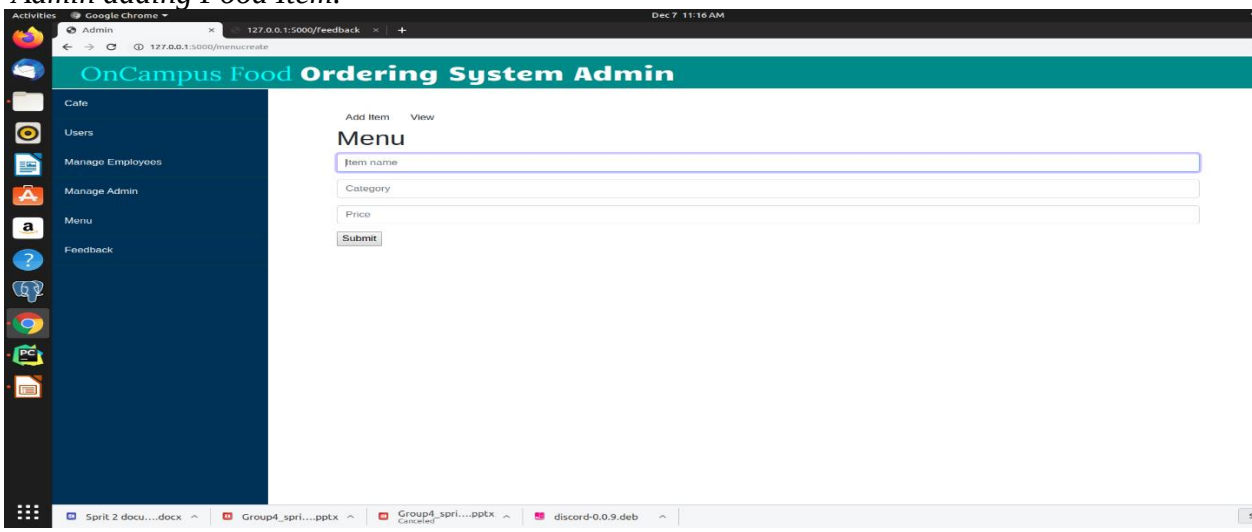


4.3. *Admin Registration*

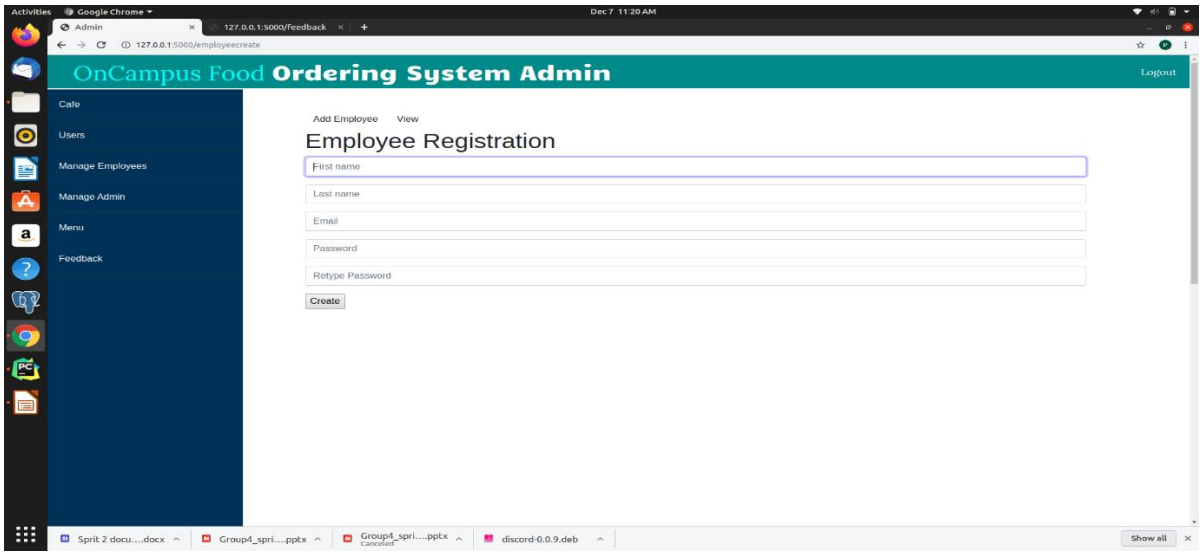




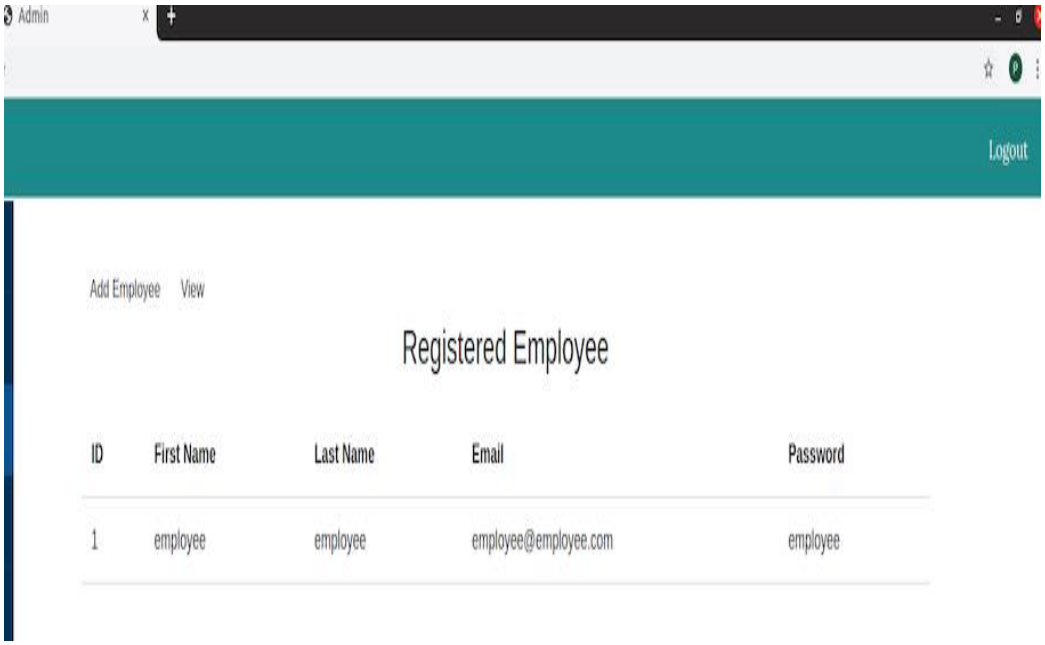
4.4. *Admin adding Food Item:*



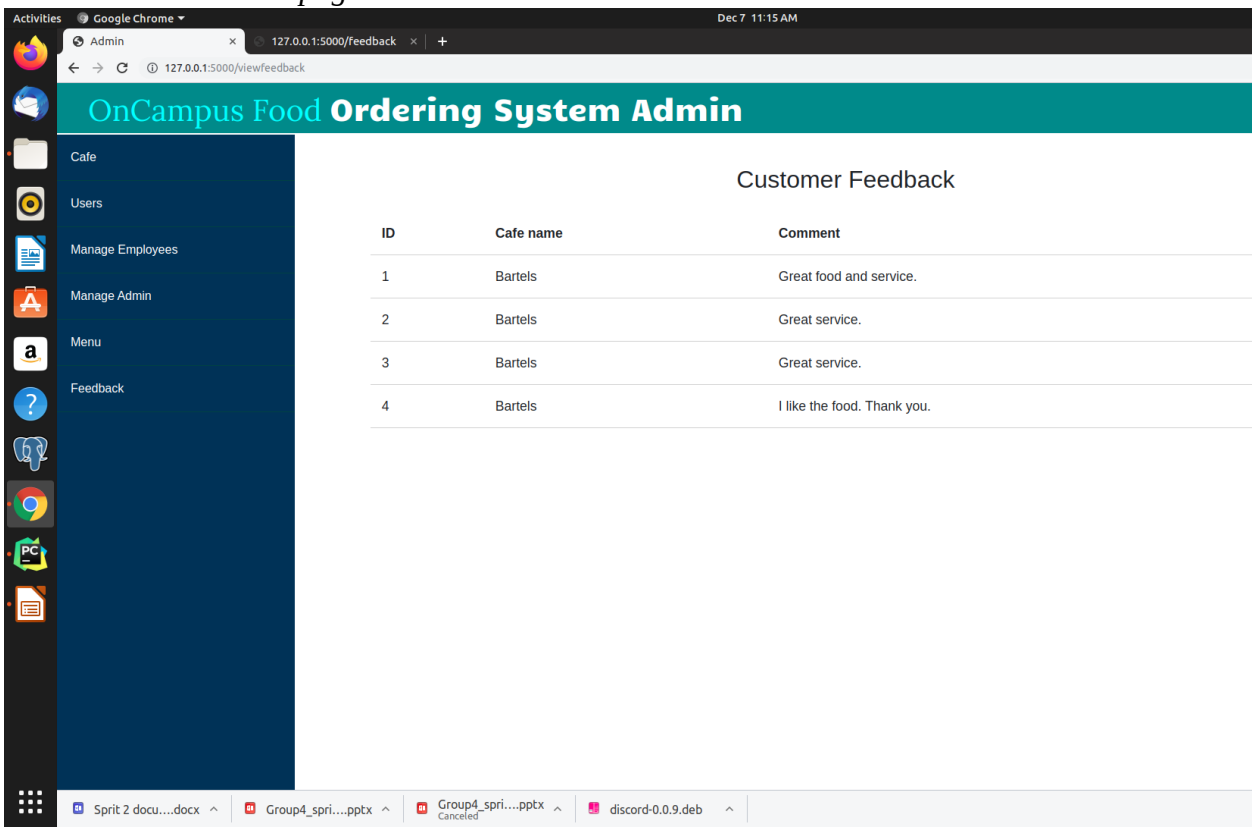
4.5. *Admin Registering New Employee:*



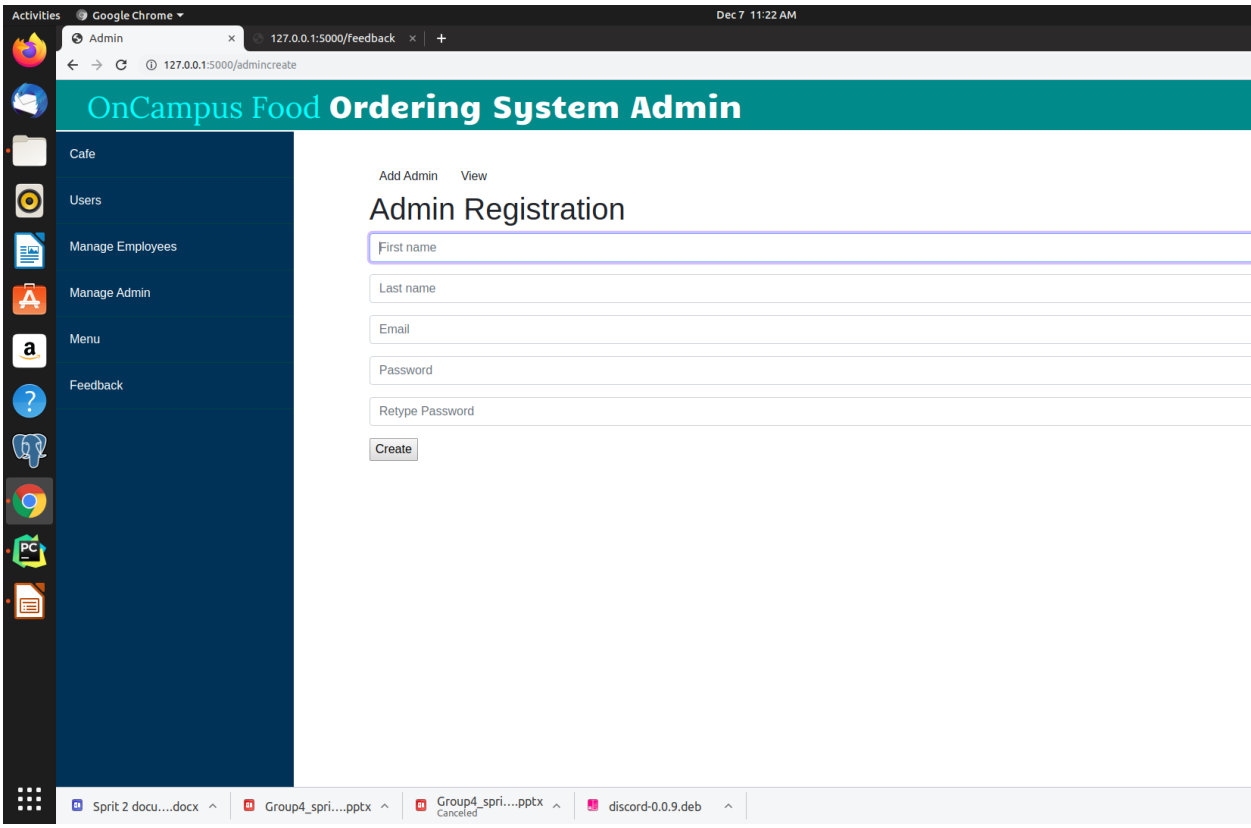
4.6. Manage employee page from admin View:



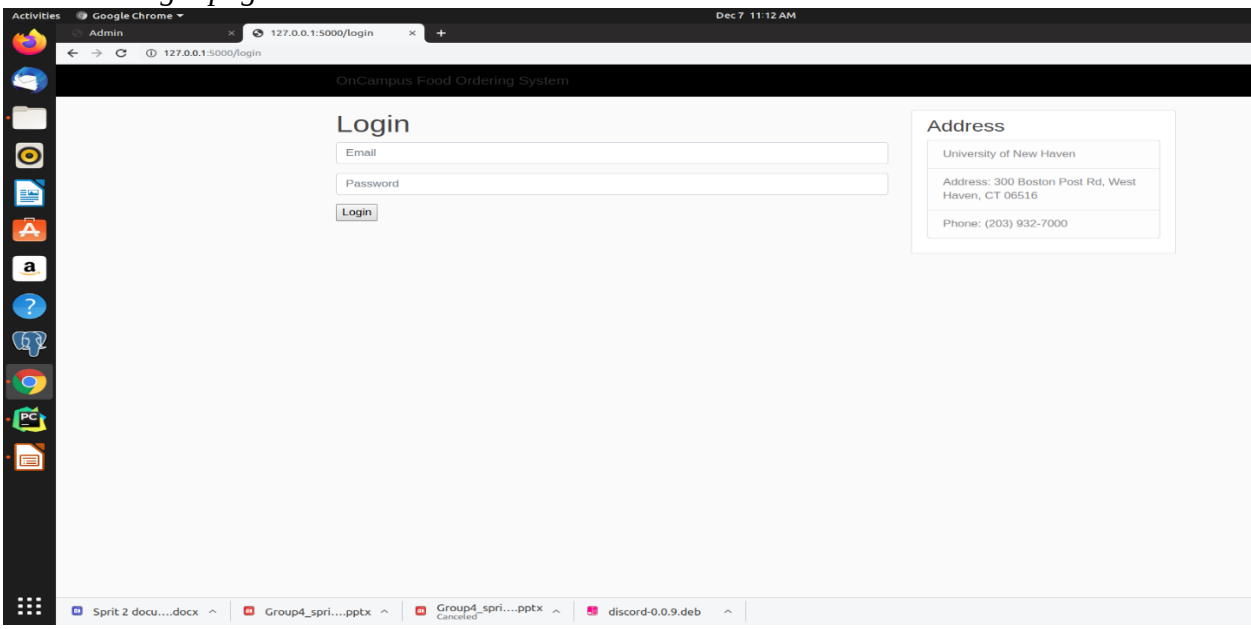
4.7. Admin View Feedback page:



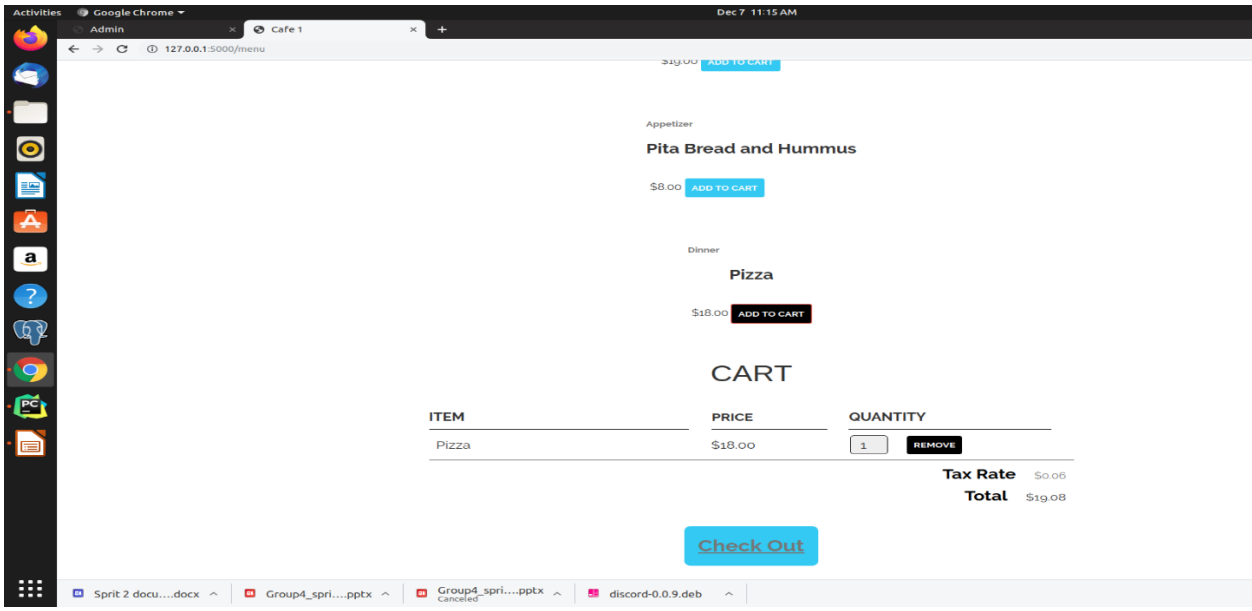
4.8. Admin Registering New Admin:



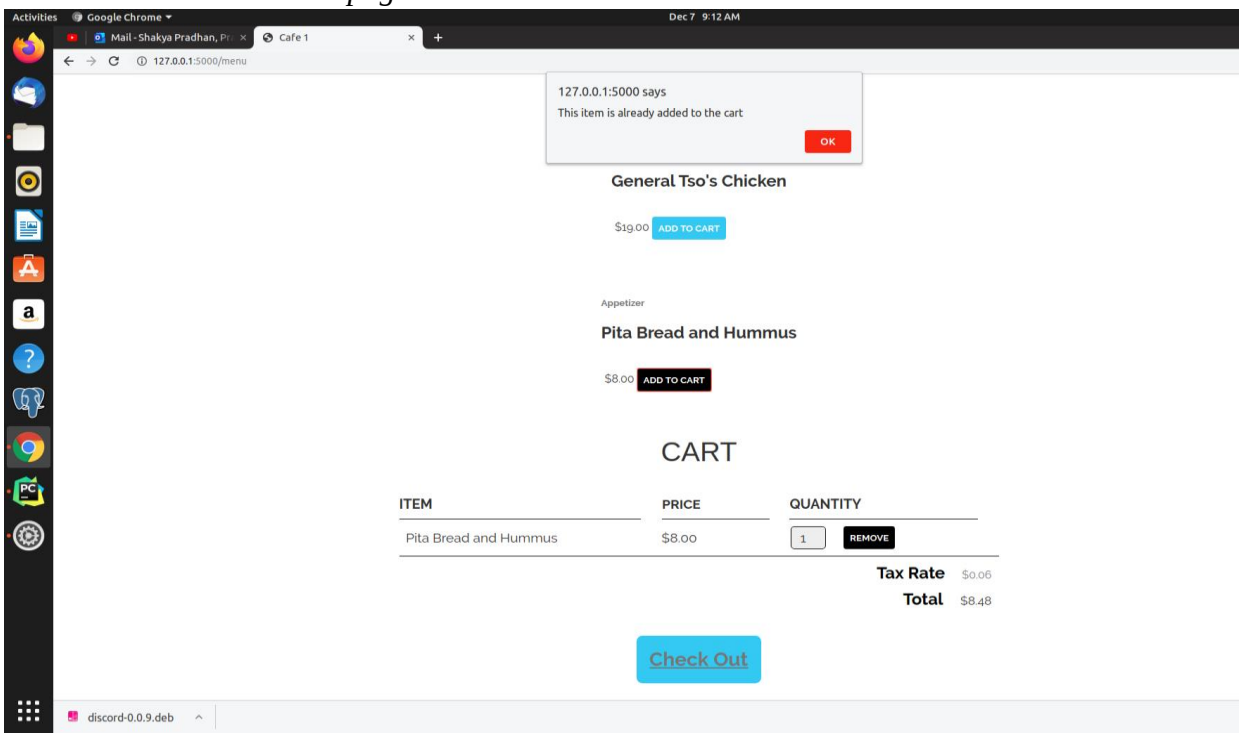
4.9. Customer login page:



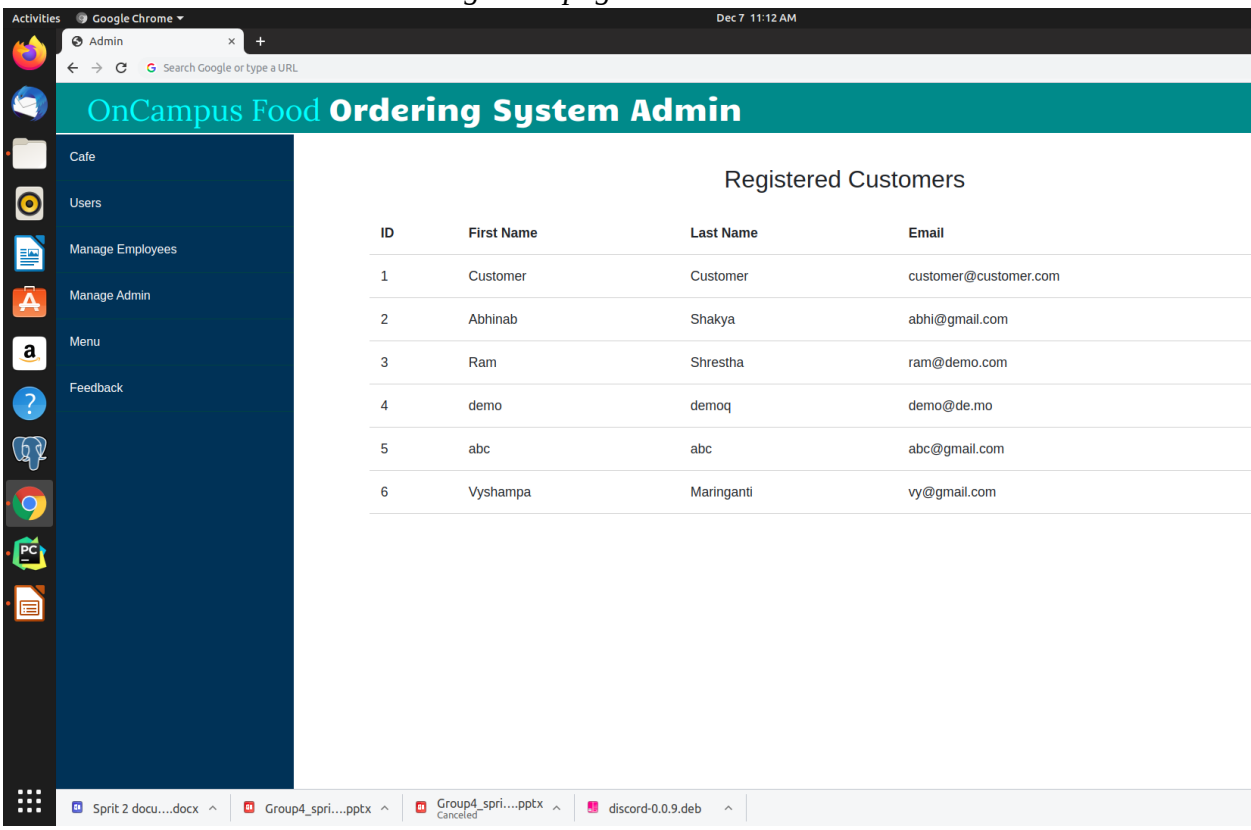
4.10. Customer Add to cart Page:

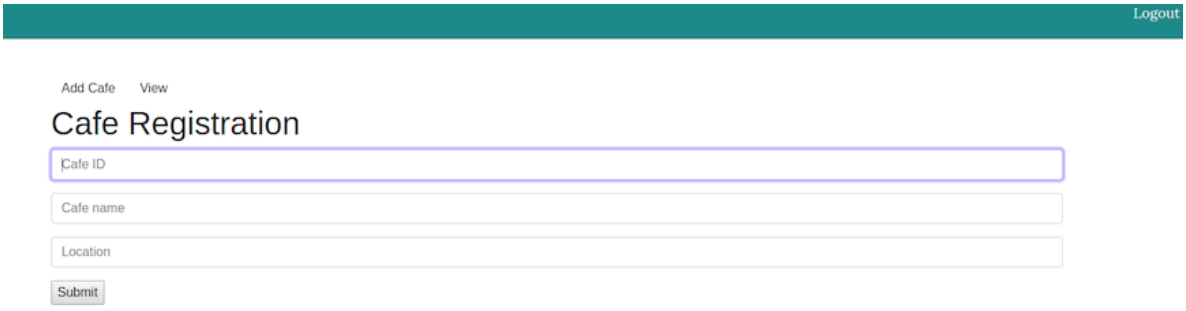


4.11. Customer Checkout page:

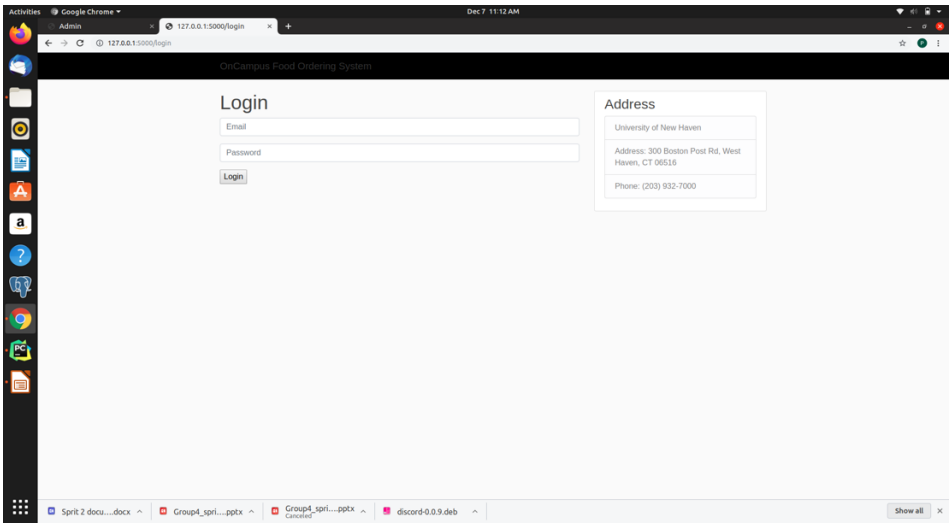


4.12. Admin View Users-Monitoring users page:

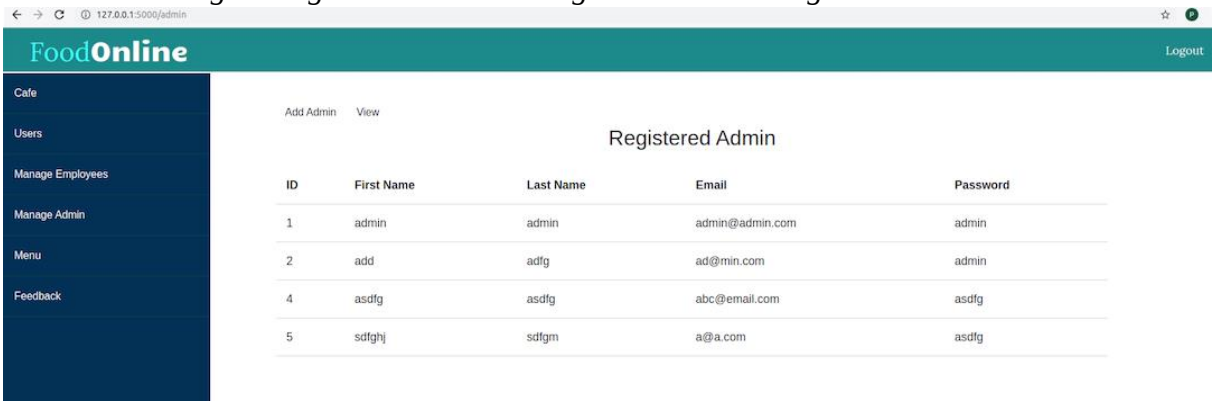




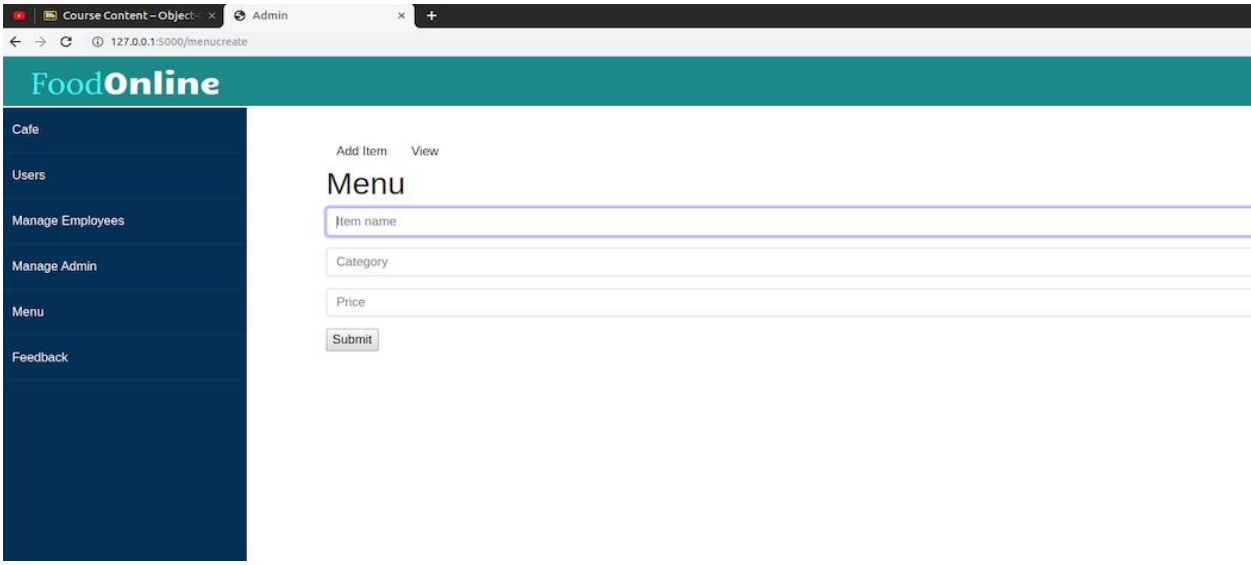
4.13. Customer Login Page

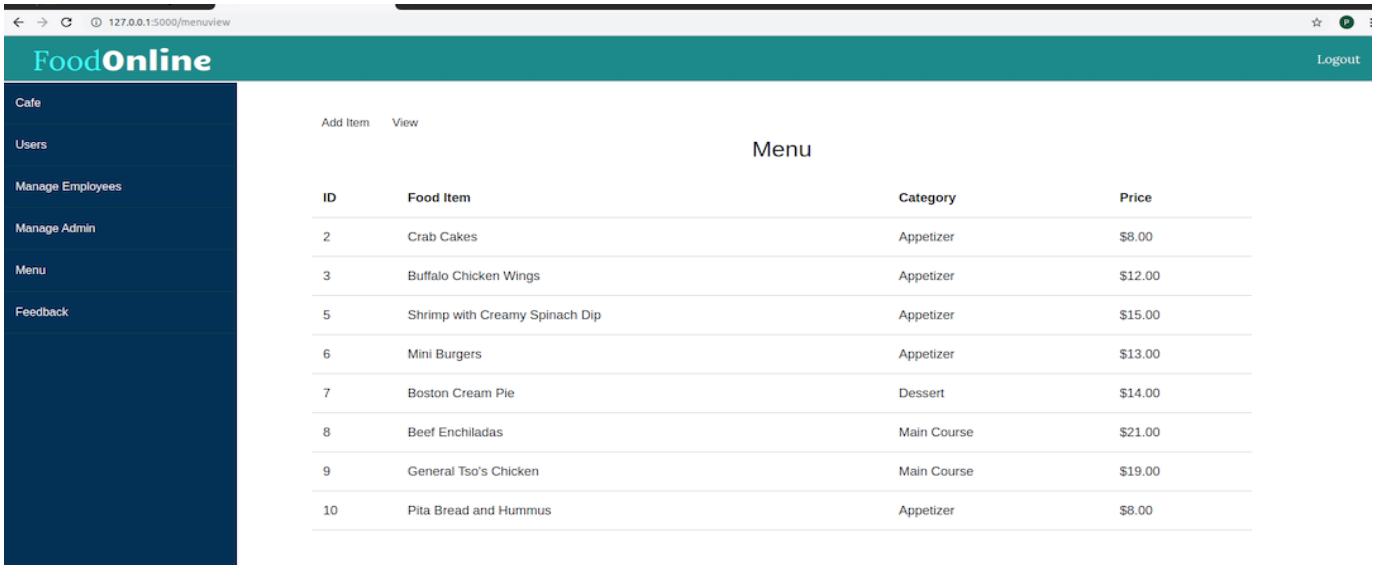


4.14. Admin Registering another admin- Registered admin Page



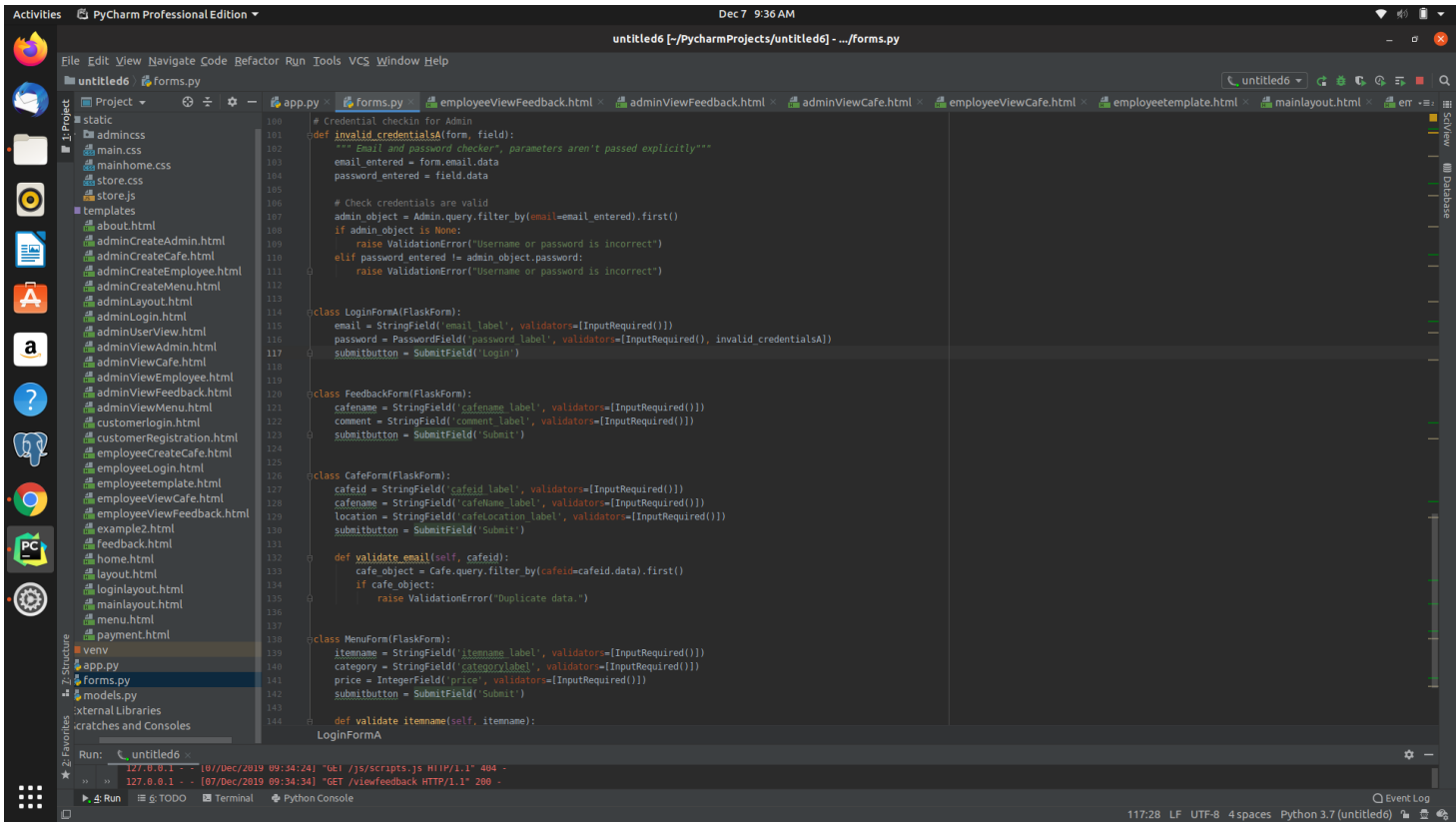
4.15. Food Category Addition Page

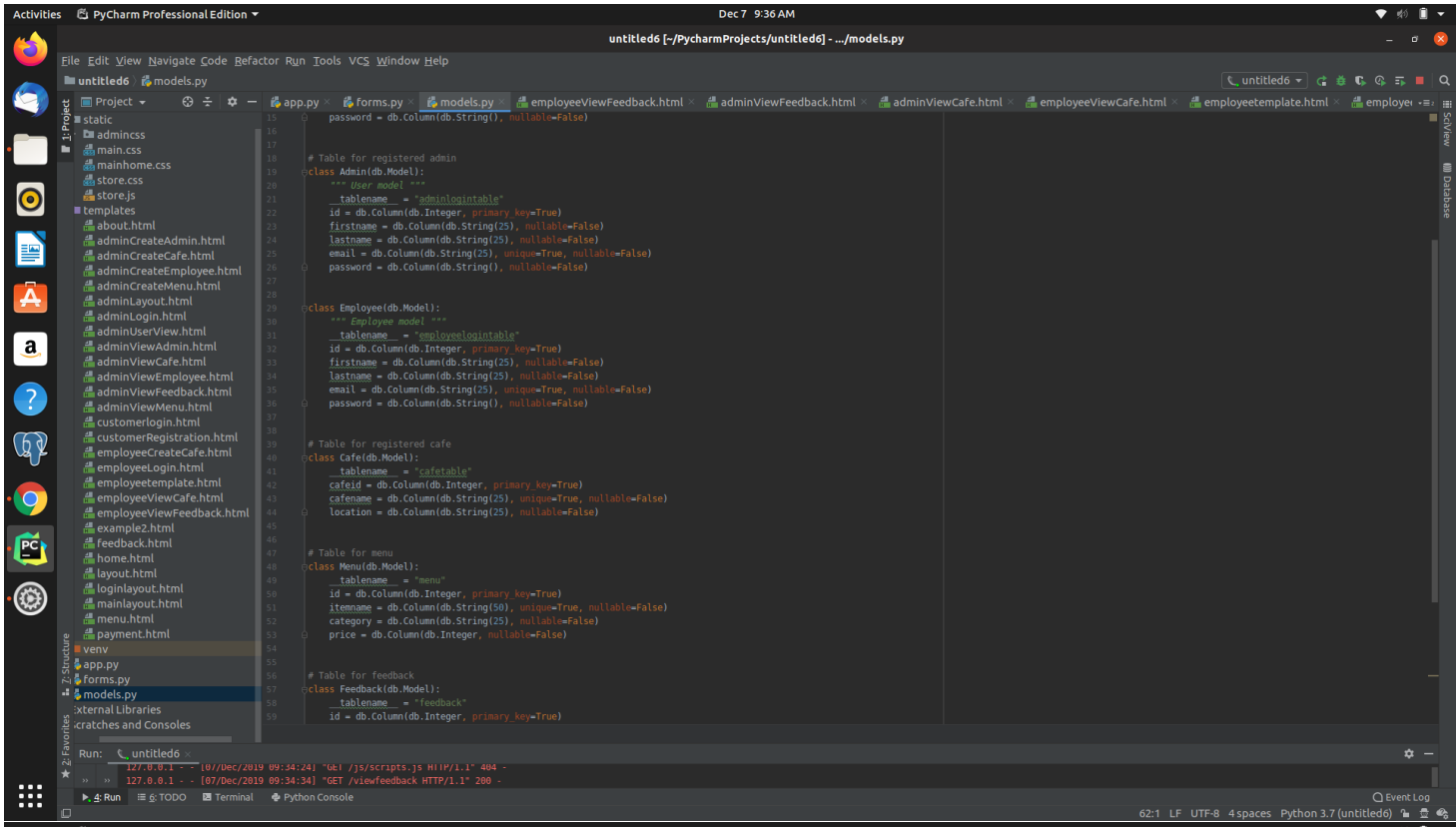


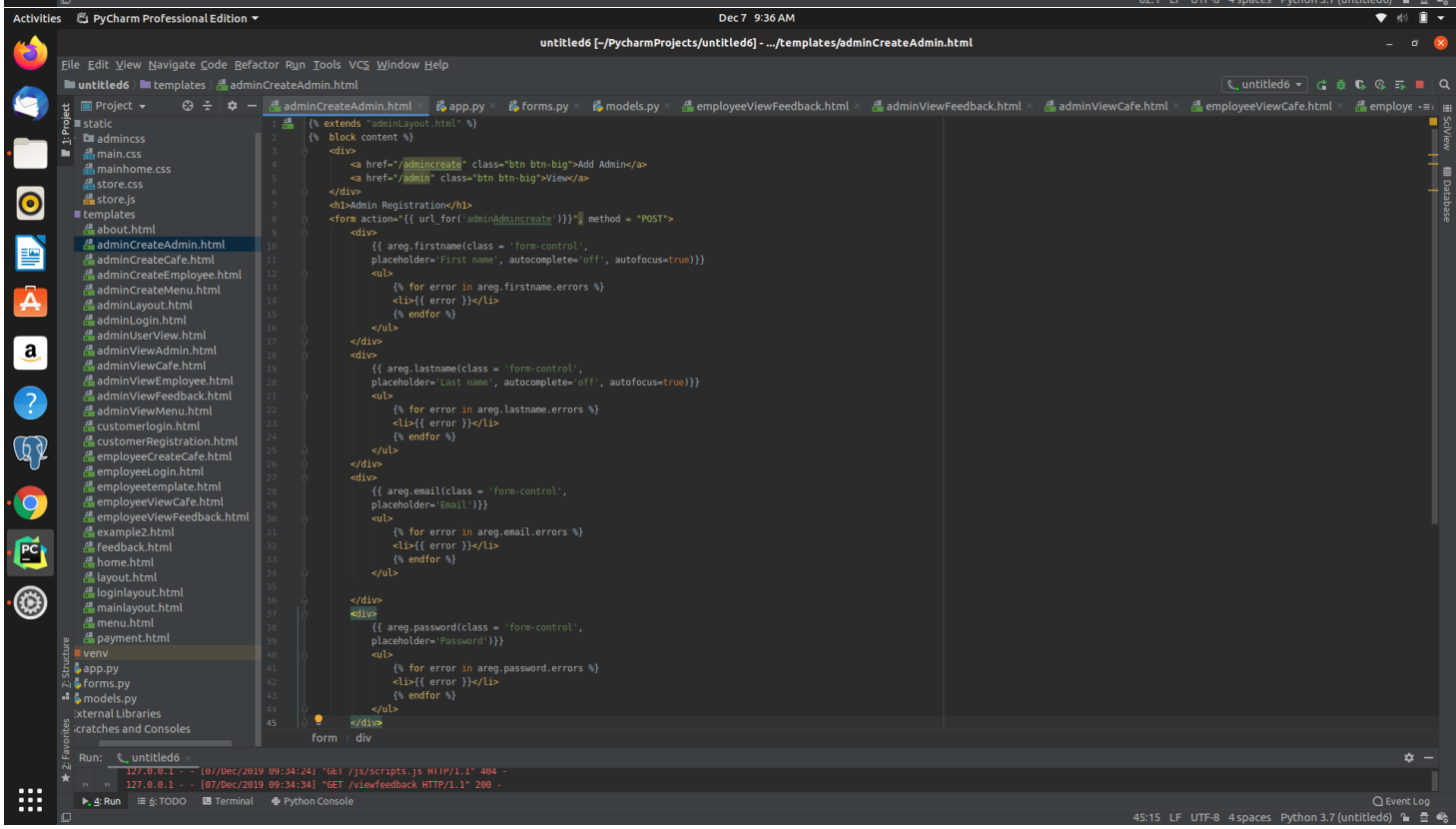


5. SCREENSHOTS OF CODE:

1. Program




Activities
PyCharm Professional Edition
Dec 7 9:36 AM
untitled6 [~/PycharmProjects/untitled6] - .../models.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
untitled6 models.py
app.py forms.py models.py employeeViewFeedback.html adminViewFeedback.html adminViewCafe.html employeeViewCafe.html employeetemplate.html
Project
static
admincss
main.css
mainhome.css
store.css
store.js
templates
about.html
adminCreateAdmin.html
adminCreateCafe.html
adminCreateEmployee.html
adminCreateMenu.html
adminLayout.html
adminLogin.html
adminUserView.html
adminViewAdmin.html
adminViewCafe.html
adminViewEmployee.html
adminViewFeedback.html
adminViewMenu.html
customerlogin.html
customerRegistration.html
employeeCreateCafe.html
employeeLogin.html
employeetemplate.html
employeeViewCafe.html
employeeViewFeedback.html
example2.html
feedback.html
home.html
layout.html
loginlayout.html
mainlayout.html
menu.html
payment.html
venv
app.py
forms.py
models.py
External Libraries
Scratches and Consoles
password = db.Column(db.String(), nullable=False)
# Table for registered admin
class Admin(db.Model):
""" User model """
__tablename__ = "adminlogintable"
id = db.Column(db.Integer, primary_key=True)
firstname = db.Column(db.String(25), nullable=False)
lastname = db.Column(db.String(25), nullable=False)
email = db.Column(db.String(25), unique=True, nullable=False)
password = db.Column(db.String(), nullable=False)
class Employee(db.Model):
""" Employee model """
__tablename__ = "employeelogintable"
id = db.Column(db.Integer, primary_key=True)
firstname = db.Column(db.String(25), nullable=False)
lastname = db.Column(db.String(25), nullable=False)
email = db.Column(db.String(25), unique=True, nullable=False)
password = db.Column(db.String(), nullable=False)
# Table for registered cafe
class Cafe(db.Model):
__tablename__ = "cafetable"
cafeid = db.Column(db.Integer, primary_key=True)
cafename = db.Column(db.String(25), unique=True, nullable=False)
location = db.Column(db.String(25), nullable=False)
# Table for menu
class Menu(db.Model):
__tablename__ = "menu"
id = db.Column(db.Integer, primary_key=True)
itemname = db.Column(db.String(50), unique=True, nullable=False)
category = db.Column(db.String(25), nullable=False)
price = db.Column(db.Integer, nullable=False)
# Table for feedback
class Feedback(db.Model):
__tablename__ = "feedback"
id = db.Column(db.Integer, primary_key=True)
Run: untitled6
127.0.0.1 - - [07/Dec/2019 09:34:24] "GET /js/scripts.js HTTP/1.1" 404 -
127.0.0.1 - - [07/Dec/2019 09:34:34] "GET /viewfeedback HTTP/1.1" 200 -
4: Run 6: TODO Terminal Python Console
Event Log
62:1 LF UTF-8 4 spaces Python 3.7 (untitled6)


Activities
PyCharm Professional Edition
Dec 7 9:36 AM
untitled6 [~/PycharmProjects/untitled6] - .../templates/adminCreateAdmin.html
File Edit View Navigate Code Refactor Run Tools VCS Window Help
untitled6 templates adminCreateAdmin.html
adminCreateAdmin.html app.py forms.py models.py employeeViewFeedback.html adminViewFeedback.html adminViewCafe.html employeeViewCafe.html
{% extends "adminLayout.html" %}
{% block content %}

Add Admin
View

<h1>Admin Registration</h1>
<form action="{{ url_for('adminAdmincreate')}}" method = "POST">

{{ areg.firstname(class = 'form-control',
placeholder='First name', autocomplete='off', autofocus=true)}}

{% for error in areg.firstname.errors %}
{{ error }}
{% endfor %}

{{ areg.lastname(class = 'form-control',
placeholder='Last name', autocomplete='off', autofocus=true)}}

{% for error in areg.lastname.errors %}
{{ error }}
{% endfor %}

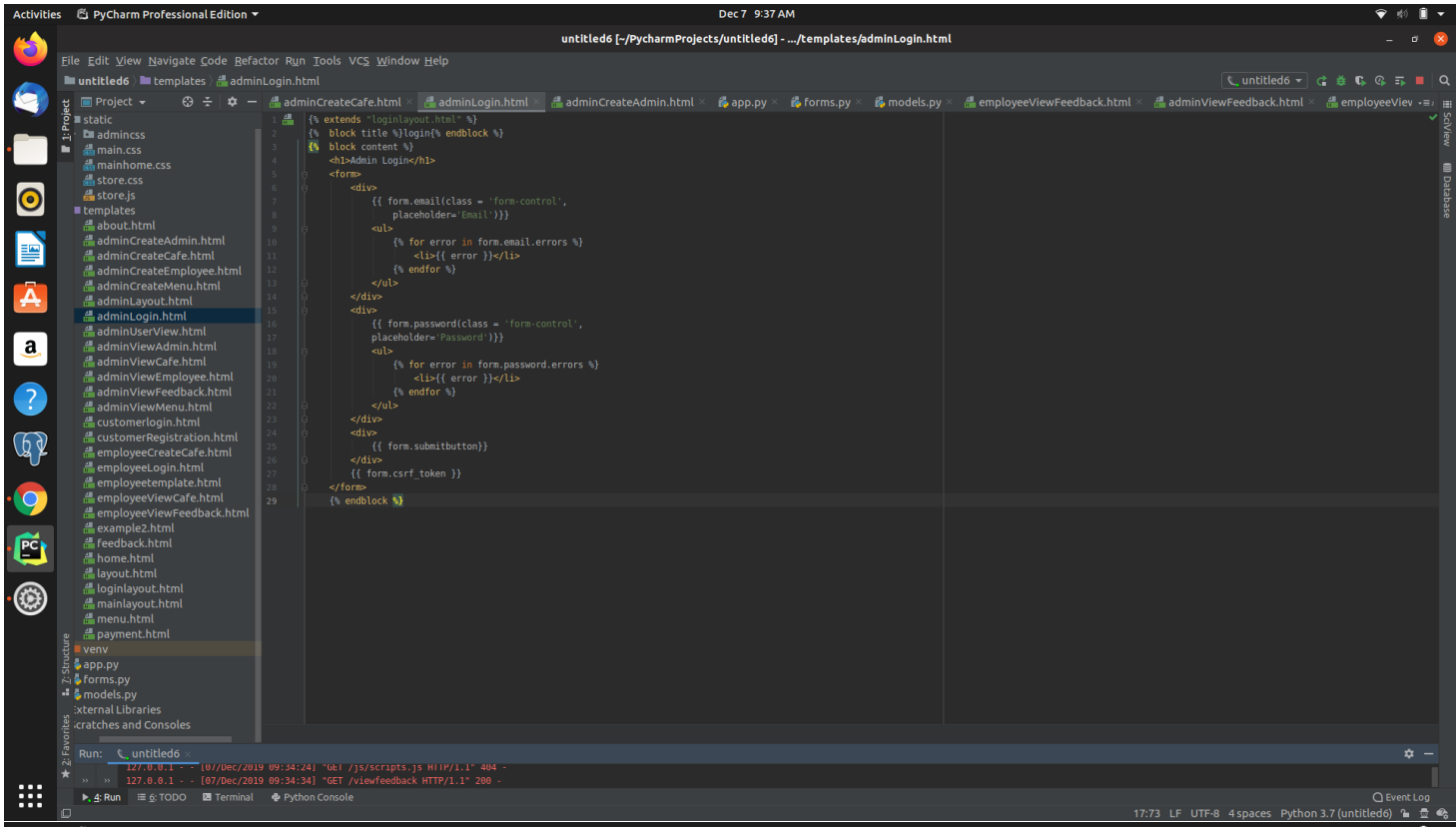
{{ areg.email(class = 'form-control',
placeholder='Email')}}

{% for error in areg.email.errors %}
{{ error }}
{% endfor %}

{{ areg.password(class = 'form-control',
placeholder='Password')}}

{% for error in areg.password.errors %}
{{ error }}
{% endfor %}

form div
Run: untitled6
127.0.0.1 - - [07/Dec/2019 09:34:24] "GET /js/scripts.js HTTP/1.1" 404 -
127.0.0.1 - - [07/Dec/2019 09:34:34] "GET /viewfeedback HTTP/1.1" 200 -
4: Run 6: TODO Terminal Python Console
Event Log
45:15 LF UTF-8 4 spaces Python 3.7 (untitled6)


Activities
PyCharm Professional Edition
Dec 7 9:37 AM
untitled6 [~/PycharmProjects/untitled6] - .../templates/adminLogin.html
File Edit View Navigate Code Refactor Run Tools VCS Window Help
untitled6 templates adminLogin.html
Project
static
admincss
main.css
mainhome.css
store.css
store.js
templates
about.html
adminCreateAdmin.html
adminCreateCafe.html
adminCreateEmployee.html
adminCreateMenu.html
adminLayout.html
adminLogin.html
adminUserView.html
adminViewAdmin.html
adminViewCafe.html
adminViewEmployee.html
adminViewFeedback.html
adminViewMenu.html
customerlogin.html
customerRegistration.html
employeeCreateCafe.html
employeeLogin.html
employeetemplate.html
employeeViewCafe.html
employeeViewFeedback.html
example2.html
feedback.html
home.html
layout.html
loginlayout.html
mainlayout.html
menu.html
payment.html
venv
app.py
forms.py
models.py
External Libraries
Scratches and Consoles
adminCreateCafe.html
adminLogin.html
adminCreateAdmin.html
app.py
forms.py
models.py
employeeViewFeedback.html
adminViewFeedback.html
{% extends "loginlayout.html" %}
{% block title %}login{% endblock %}
{% block content %}
<h1>Admin Login</h1>
<form>

{{ form.email(class = 'form-control',
placeholder='Email')}}

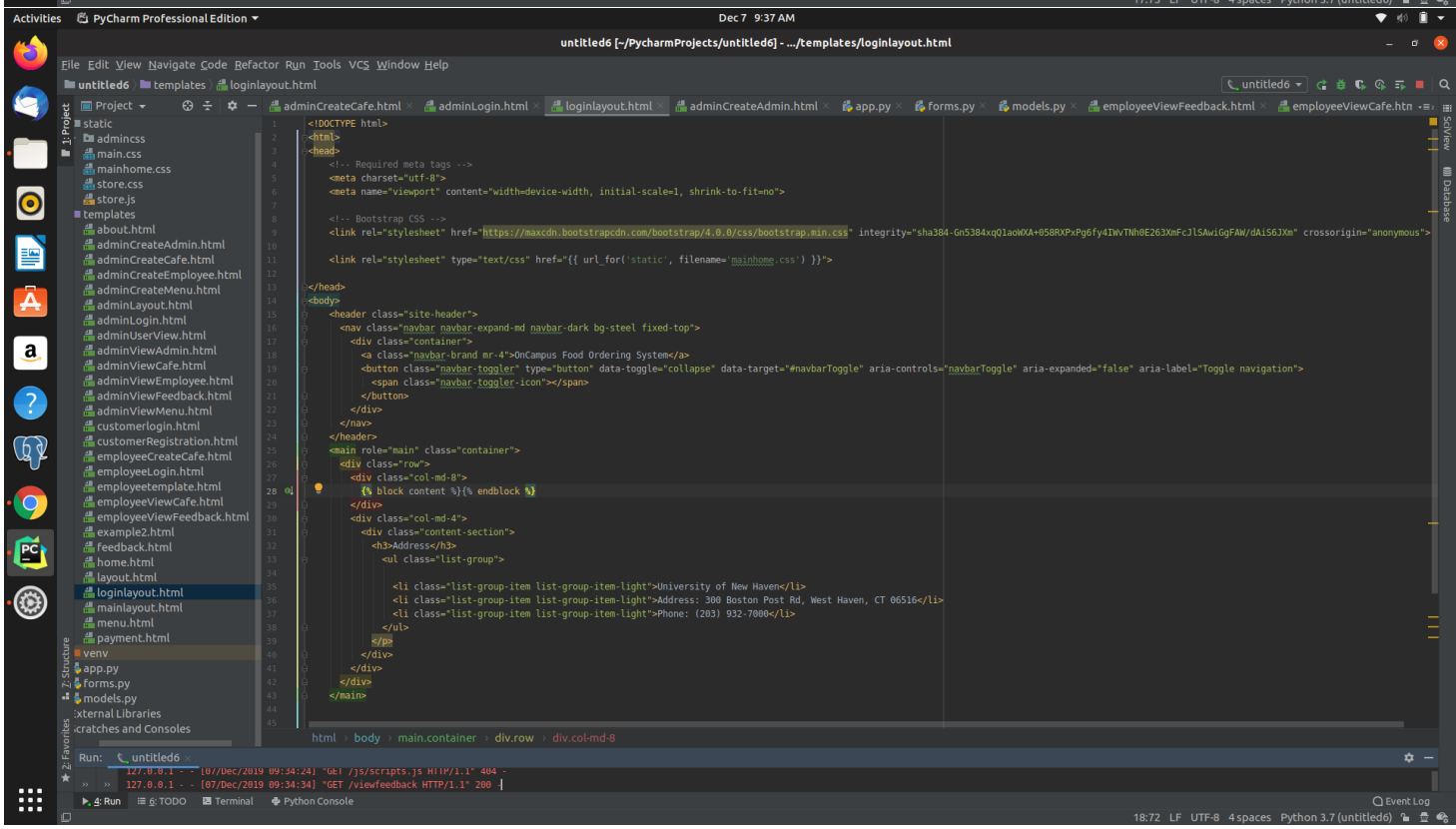
{% for error in form.email.errors %}
{{ error }}
{% endfor %}

{{ form.password(class = 'form-control',
placeholder='Password')}}

{% for error in form.password.errors %}
{{ error }}
{% endfor %}

{{ form.submitbutton}}

{{ form.csrf_token }}
</form>
{% endblock %}
Run: untitled6
127.0.0.1 - - [07/Dec/2019 09:34:24] "GET /js/scripts.js HTTP/1.1" 404 -
127.0.0.1 - - [07/Dec/2019 09:34:34] "GET /viewfeedback HTTP/1.1" 200 -
4: Run
6: TODO
Terminal
Python Console
Event Log
17:73 LF UTF-8 4 spaces Python 3.7 (untitled6)


Activities
PyCharm Professional Edition
Dec 7 9:37 AM
untitled6 [~/PycharmProjects/untitled6] - .../templates/loginlayout.html
File Edit View Navigate Code Refactor Run Tools VCS Window Help
untitled6 templates loginlayout.html
Project
static
admincss
main.css
mainhome.css
store.css
store.js
templates
about.html
adminCreateAdmin.html
adminCreateCafe.html
adminCreateEmployee.html
adminCreateMenu.html
adminLayout.html
adminLogin.html
adminUserView.html
adminViewAdmin.html
adminViewCafe.html
adminViewEmployee.html
adminViewFeedback.html
adminViewMenu.html
customerlogin.html
customerRegistration.html
employeeCreateCafe.html
employeeLogin.html
employeetemplate.html
employeeViewCafe.html
employeeViewFeedback.html
example2.html
feedback.html
home.html
layout.html
loginlayout.html
mainlayout.html
menu.html
payment.html
venv
app.py
forms.py
models.py
External Libraries
Scratches and Consoles
adminCreateCafe.html
adminLogin.html
loginlayout.html
adminCreateAdmin.html
app.py
forms.py
models.py
employeeViewFeedback.html
<!DOCTYPE html>
<html>
<head>
<!-- Required meta tags -->
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
<!-- Bootstrap CSS -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">
<link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='mainhome.css') }}">
</head>
<body>
<header class="site-header">
<nav class="navbar navbar-expand-md navbar-dark bg-steel fixed-top">

OnCampus Food Ordering System
<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarToggle" aria-controls="navbarToggle" aria-expanded="false" aria-label="Toggle navigation">

</button>

</nav>
</header>
<main role="main" class="container">

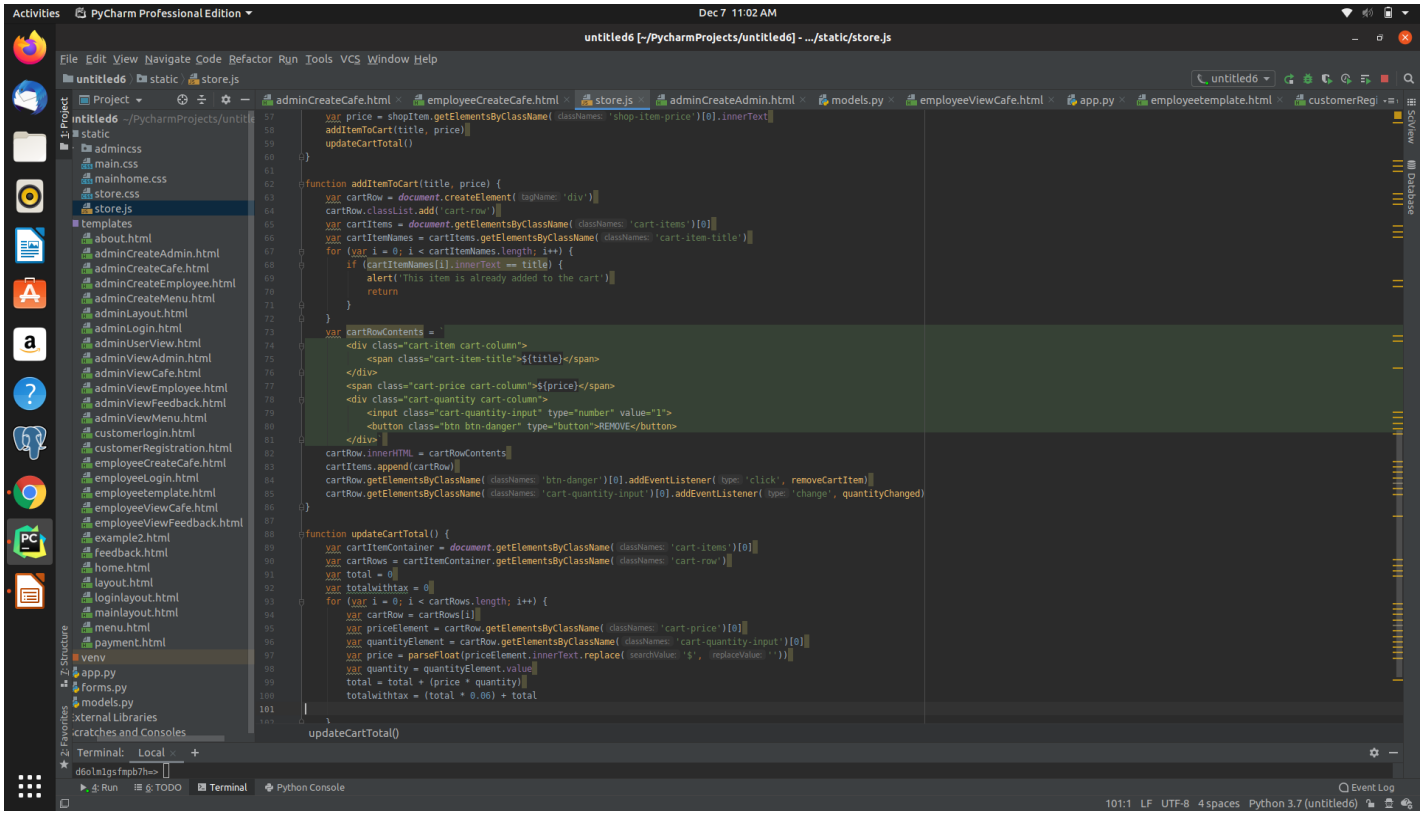
{% block content %}{% endblock %}

<h3>Address</h3>
<ul class="list-group">
<li class="list-group-item list-group-item-light">University of New Haven
<li class="list-group-item list-group-item-light">Address: 300 Boston Post Rd, West Haven, CT 06516
<li class="list-group-item list-group-item-light">Phone: (203) 932-7000

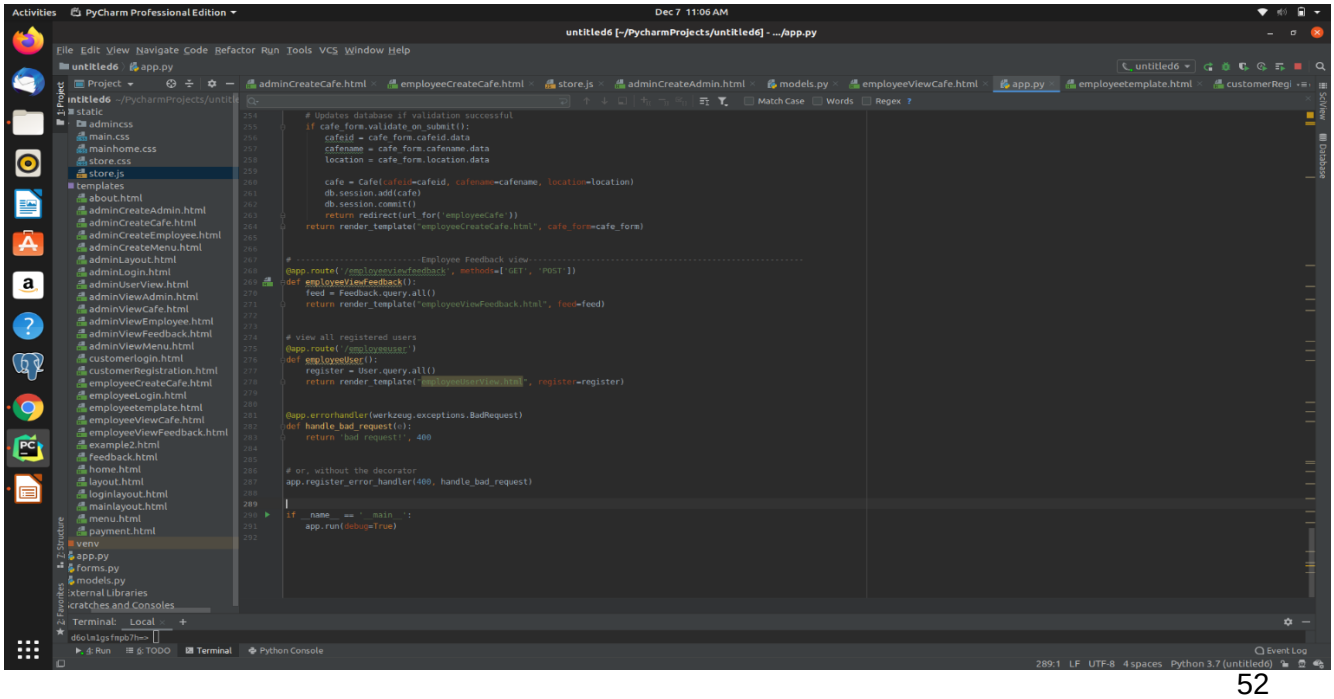
</p>

</main>
html body main.container div.row div.col-md-8
Run: untitled6
127.0.0.1 - - [07/Dec/2019 09:34:24] "GET /js/scripts.js HTTP/1.1" 404 -
127.0.0.1 - - [07/Dec/2019 09:34:34] "GET /viewfeedback HTTP/1.1" 200 -
4: Run
6: TODO
Terminal
Python Console
Event Log
18:72 LF UTF-8 4 spaces Python 3.7 (untitled6)

5.1 Try Catch Error Handler:

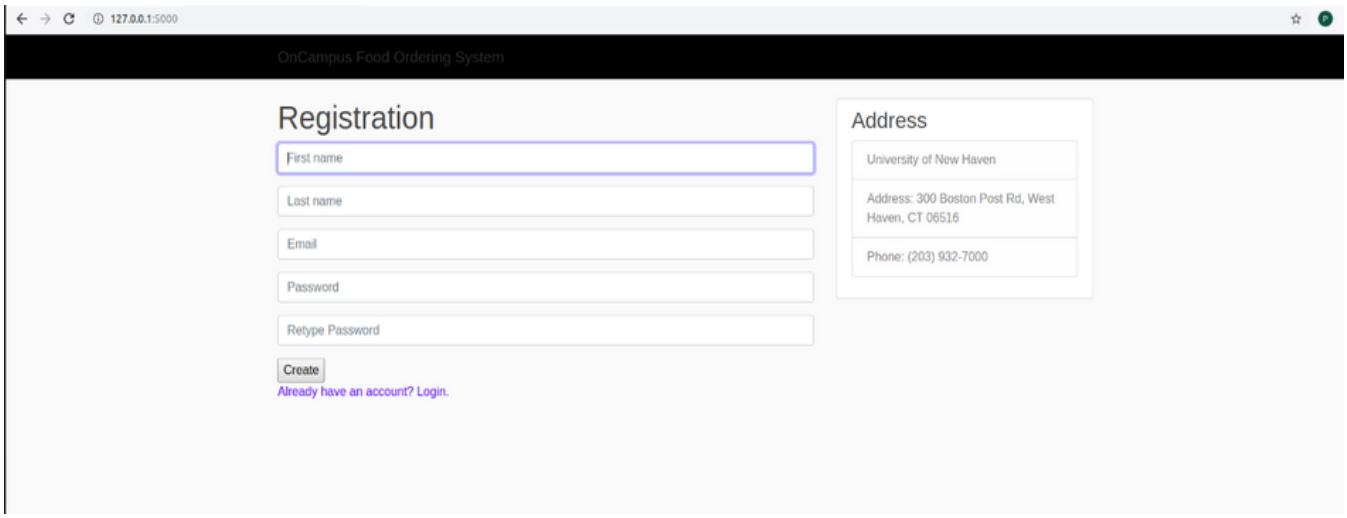


5.2 **ADD TO CART CODE:**

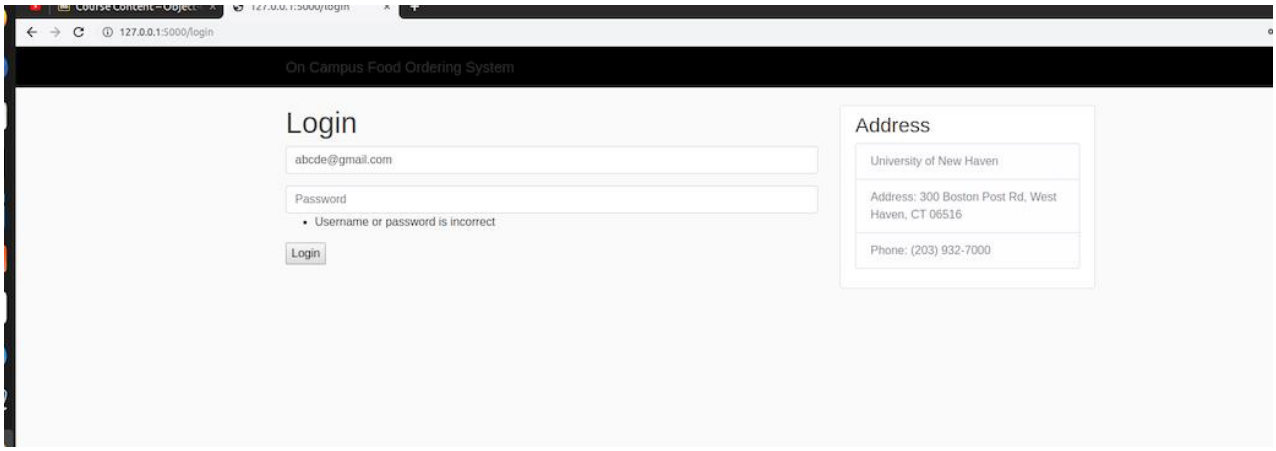


Testcases:

Test Case 1	
Positive Test Case	1. User enters the credentials 2. User registers successfully
Negative Test Case	1. User enters already existing email id. 2. Error message is displayed as “email already exists”.

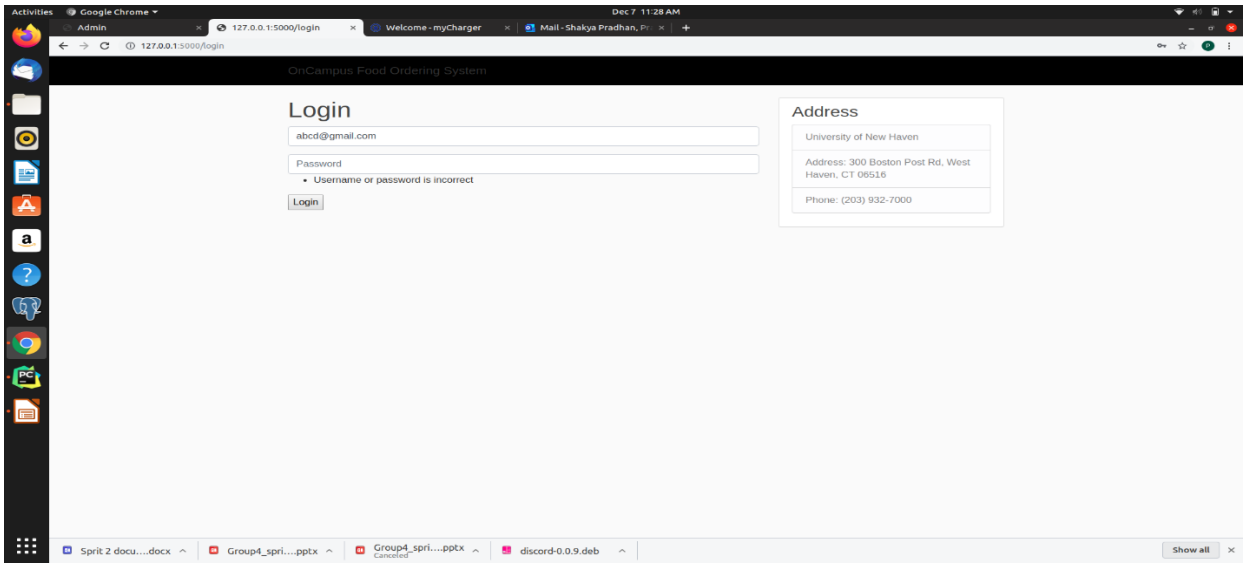


Test Case 2	
Positive Test Case	1. User enters the credentials 2. User logged in to the system
Negative Test Case	1. User enters invalid credentials 2. Error message is displayed



Positive test case Screenshot:

Test Case 3	
Positive Test Case	1. Admin enters the credentials 2. Admin is logged in to the system
Negative Test Case	1. Admin enters invalid credentials 2. Error message is displayed



6. Reflection

6.1 What worked well

1.2. It gave us an opportunity to learn new framework and database.

1.3. No issues with the group members.

6.2. What didn't work well

1.4. Time crunch due to which certain parts of the project were overlooked.

1.5. Restarting and completing the entire project with OO language after second sprint was difficult.

1.6. Due to this we faced time crunch and lot of stress in completing the project

1.7. Difference in the vision for the project led us to splitting group in two.

6.3. Lessons learned

1.8. We learned a new framework within the given time frame and tried our best to implement all the features.

1.9. Since none of group members had previous experience on working on a real time project, at times the understanding concepts of linking different aspects of the application was difficult.

1.10. Working as a group requires patience and understanding.

1.11. Debugging and error correction is painful and time-consuming task.

1.12. It is important to have similar vision between the group members.

6.4. Impact on project

1.13. Could have worked on more features but couldn't due to time crunch.

7.Appendices

7.1 Quality Attributes

Attribute	Bus Rank	Design Align	Run Time Quality Attribute Documents
Performance	3	Meets	Definition: System performance describes attributes of the system including system navigational abilities and connectivity with interfacing systems. May include system response time, number of transactions per minute, and other critical performance characteristics which define this project.

			Reason for Business Rank: Performance is an important attribute for this application. As transactions are completed online, the navigational abilities and response time plays a critical role in success of the business.
			Explanation of Design Alignment <> Meets: The user can easily navigate through the website, order items from menu and make payments securely.
Reliability	3	Does not meet	Definition: Reliability defines the ability of the system to operate correctly over time. Includes consistent stability of the system during available hours.
			Reason for Business Rank <> 3: Since the orders are placed online, reliability of the application is essential. The stability of application is important for the smooth transaction.
			Explanation of Design Alignment <> Meets: As of now, the project is in an incomplete state. Hence, more time is required to work on adding this attribute to the application.
Scalability	2	Meets	Definition: Scalability defines the parameters of growth within a given period of time for the project. The solution must scale in such a way that additional capacity can be added expediently in a technically and economically feasible way. Enhancement or project should include existing scalability parameters as the baseline.
			Reason for Business Rank <> 3: This system is built for the students, professors and visitors on campus assuming that there would not be rapid increase in students within few years. Hence, the need to scale the application in the future remain minimal. Hence,

			though the application itself is scalable, the scalability has been ranked 2.
			Explanation of Design Alignment <> Meets: The additional capacity and features can easily be added to this application. More number of cafes can be easily added. The menu, cart and payment modules can be added simply.
Security	2	Meets	Definition: Security is the ability of the system to resist unauthorized attempts to access the system and/or components of the system.
			Reason for Business Rank <> 3: The module that required high level of security is the payment module to secure user payment details and avoid fraud which is handled by the third party. The login module is secure, hence there is low chance of system getting hacked.
			Explanation of Design Alignment <> Meets: The login module is programmed to ensure the security of the system. The payment module is handled by the third party.
Availability	3	Does not meet	Definition: Availability defines the days and hours the system is anticipated and/or required to be available.
			Reason for Business Rank <> 3: As the business relies on online transaction, availability becomes an indispensable part of the system.
			Explanation of Design Alignment <> Meets: The work on the system is not yet complete to meet the criteria of availability.
Supportability	3	Does not meet	Definition: Supportability defines the requirement for problems can be diagnosed and addressed quickly, with appropriately trained resources.
			Reason for Business Rank <> 3: Trouble shooting problems as soon as they arise is important when the business is done online. Addressing and diagnosing problems quickly is essential for an online business. Hence, supportability is ranked 3.
			Explanation of Design Alignment <> Meets: The application is in an incomplete state. This attribute will be added to the application as soon as it reaches its final testing stage.
Traceability	2	Meets	Definition: Traceability defines the ability to trace the history, application or location of an item or activity by means of recorded identification.
			Reason for Business Rank <> 3:

			The transaction history is essential to trace the business pattern to generate the sales report which is necessary for the business. But, since the café are physically present in the university, this application acts as an online extension and not as a medium to calculate profit or loss of their café. Hence, traceability attribute stands at rank 2.
			Explanation of Design Alignment <> Meets: Currently, the system can trace the history of transactions that have been made to calculate and present sales report.
User Usability	3	Meets	Definition: User Usability defines how easy the interface can be used to successfully complete a task. Includes clear, concise navigation, terminology, and error messaging.
			Reason for Business Rank <> 3: The application is user friendly and easy to navigate through pages and carryout orders and transactions.
			Explanation of Design Alignment <> Meets: The user can easily navigate through the pages, choose café, and select food items. Even though the work on the application is in progress state, user usability is one of the key attributes.

Development Qualities

Attribute	Bus Rank	Design Align	Development Quality Attribute Documents
Reusability	3	Meets	Definition: The reusability of the system is the ability to reuse portions of the system in other applications.
			Reason for Business Rank <> 3: Each module in the system are developed with reusability in mind. The modules are not extensively dependent on each other for task completion.
			Explanation of Design Alignment <> Meets: The classes we have created so far are as much independent as possible. The portion that have been created can easily be reused in another application. We have created new services as well as added existing services.
Extensibility	3	Meets	Definition: Extensibility defines the degree to which the system can be extended to incorporate new functionality without changing the fundamental architecture, design or core source code. Extensions should be accomplished via published interfaces such as services, APIs, metadata or well-defined user exists.
			Reason for Business Rank <> 3: This application is extensible and new functionalities can easily be incorporated. For example, the payment gateway module can be easily incorporated with the system to complete the payment process.
			Explanation of Design Alignment <> Meets: The work on payment module is pending but our project meets the extensibility of attribute
Modifiability	1	Meets	Definition: Modifiability (aka maintainability) defines the efficiency (financial and logical) of making enhancements or maintenance changes. If a system uses obscure technology that requires high priced consultants, then even though it may be quick to change, its modifiability can still be low.
			Reason for Business Rank <> 3: The On-campus food ordering system is a lightweight application that can be maintained with a low cost. The application does not use obscure technologies that require high cost for maintenance.
			Explanation of Design Alignment <> Meets: The features and functionalities can be maintained easily. Even though the application is still in progress state, as the

			programmers, we can say the application can be maintained easily and at a low cost.
Ease of Integration	2	Meets	Definition: Ease of Integration defines how easily the application or assets can be integrated or packaged with new applications.
			Reason for Business Rank <> 3: Since, the application is lightweight, the ease of integration has been ranked 2. The application can be easily integrated or packaged with a new application. Since the system is small, it does not require much investment of time or money.
			Explanation of Design Alignment <> Meets: The On-campus food ordering system has been built on python flask which is a lightweight but flexible platform. The application can easily be integrated with new application with the help of API's.
Phase-In "ability" ***(Subset-ability)***	3	Meets	Definition: Phase In "ability" (aka subset-ability) defines how/if the application must accommodate implementation/rollout schedules and its relevance to the project as it can impact design.
			Reason for Business Rank <> 3: It is important for the individual modules of the application to work independently, so that the design can be kept clean which makes the application easy to understand. Even through the individual modules working alone has no significance, it is important to know that they work efficiently and are free of bug before we put all the modules together to make them work as a single application.
			Explanation of Design Alignment <> Meets: The login module works independent of the other modules if the user is authorized. The registration module works independently and allows the user to register. When registration and login work together, they act as the medium for user authentication.
Conceptual Integrity	4	Meets	Definition: Conceptual Integrity refers to the ability of the architecture to communicate a clear, concise vision in line with the enterprise architecture.
			Reason for Business Rank <> 3: The conceptual integrity has been ranked 4 because it plays an integral part on how the application will communicate with its modules. The clarity in concepts will result in better structure of programming and leaves less room for conceptual error.
			Explanation of Design Alignment <> Meets:

			The development of this application started with a vision, product backlog and class diagrams, which helps in conceptualizing what the outcome should accomplish.
Testability	3	Meets	Definition: Testability describes the ability to test each component stand-alone (as with a test harness), as well as in an integrated development environment prior to integrating in a quality assurance environment. Testability needs to include unit, system, integration, end-to-end, stress, regression, and should coordinate with the Testing Strategy Document. (under separate cover)
			Reason for Business Rank <> 3: Even though this application is lightweight and small, the testing part of the application requires more time and effort to ensure the quality of the final product.
			Explanation of Design Alignment <> Meets: The login and registration modules were among the top priority in our product backlog. We tested them several times to ensure that they do not fail.
Portability	2	Meets	Definition: Portability measures the ease at which the system can be moved to different platforms. The platform may consist of hardware, operating system, Virtualization (e.g. Citrix, VMWare), application server or database server software.
			Reason for Business Rank <> 3: The rank for this attribute is 2 because the application is small, it can easily be packaged and moved to different platform. It would not cost a lot of money or time to move the application to different platforms.
			Explanation of Design Alignment <> Meets: The team members use different OS and so far the project has been working on Mac OS, Ubuntu and windows.
Developer Complexity	2	Meets	Definition: Developer complexity is how easy it is for the developer to understand and create the system.
			Reason for Business Rank <> 3: The concept of this application is now new. It is a simple application and similar applications exist in the market. Since, the application is well documented, it aids to better understanding of what the application is trying to accomplish.
			Explanation of Design Alignment <> Meets: The concept and the vision have been well documented which makes it easier for the developer to understand what they are working on, and their priorities.
Buildability	3	Meets	Definition: Buildability defines the degree to which the system can be compiled, packaged, archived and deployed using automated tools and standard processes.

			Reason for Business Rank <> 3: Multiple developers are work on the same project in different platforms. This attribute has been ranked 3 because it is an important attribute that the application must consist. It is important that the application can be compiled and deployed easily in different systems.
			Explanation of Design Alignment <> Meets: The team members work on the same project on different platforms.

7.2 Invoice

Documentation	Shelly: 15 Pratigya: 15 Vyshampa: 20
Research	Shelly: 30 Pratigya: 80 Vyshampa: 30
Design	Shelly: 20 Pratigya: 50 Vyshampa: 20
Code	Shelly: 100 Pratigya: 150 Vyshampa: 100
Testing	Shelly: - Pratigya: 15 Vyshampa:-

7.3 Reliable estimates for remaining work of the project-

Based on remaining Use Cases

As per Product Backlog 70 hours

7.4 Use Cases for the next phase:

- 8.3.1 Update the Design and Add payment module
- 8.3.2 Notify customer and employee when the order is placed
- 8.3.3 Make this application available for iOS and Android
- 8.3.4 Monitoring sales and generating reports

7.5 Tools used

a. Hardware Requirement

- Hardware Pentium
- Speed 1.1 GHz
- RAM 1GB
- Hard Disk 20GB
- Keyboard Standard Windows Keyboard
- Mouse Two or three button mouse
- Monitor SVGA

b. Software Requirements

- Operating System Windows/ Linux/ Mac
- Technology Flask
- Web Technologies HTML, JS, CSS
- IDE Pycharm
- Web Server Flask server
- Database Postgresql
- Other tools draw.io