SC2006 - TDDB - Team49

NTUFoodie

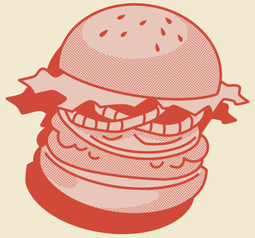
By Students FOR Students

Made by :

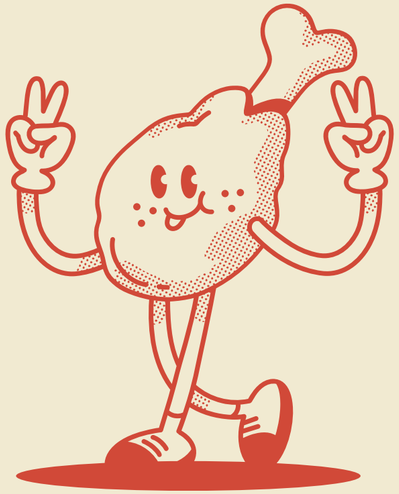
1. CHUA YONG TAI ANTHONY
2. HO KING HEI
3. KHOO QIAN YEE
4. MUHAMMAD WISNU DARMAWAN
5. RACHEL LIM LIN

Content

NTUFOODIE.CO



Problem 10



Problem ID

Two key problems:

- University students due to restricted income sources have limited purchasing power, making it a challenge to afford nutritious meals . Can lead to students experience food insecurity.
- Due to low sales, store owners often end up with excess cooked , leading to them to throw away instead since they might be unwilling to give it away for free.

The Solution

NTUFoodie

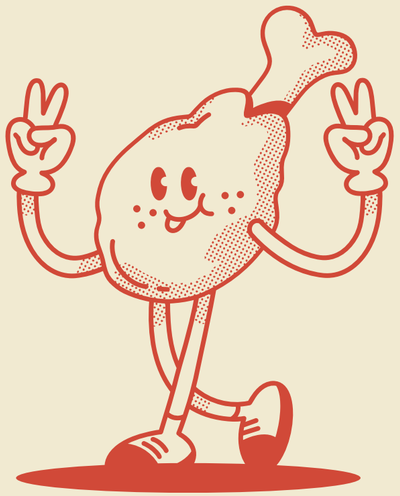
- Discounted meals to cut student costs:
  - Features a list of discounted meals for consumers, where store owners can regularly post discounted meals specifically for students
  - E asier for students to find affordable dining options on campus.

- Reducing Food Wastage
  - Store owners can post real-time “last-minute deals” on unsold food, providing deep discounts before closing times.
  - Encourages meals to be purchased at a reduced cost reducing food wastage.

NTUFoodie can not only support students financially but also promote a more sustainable campus dining ecosystem.

NTUFOODIE.CO

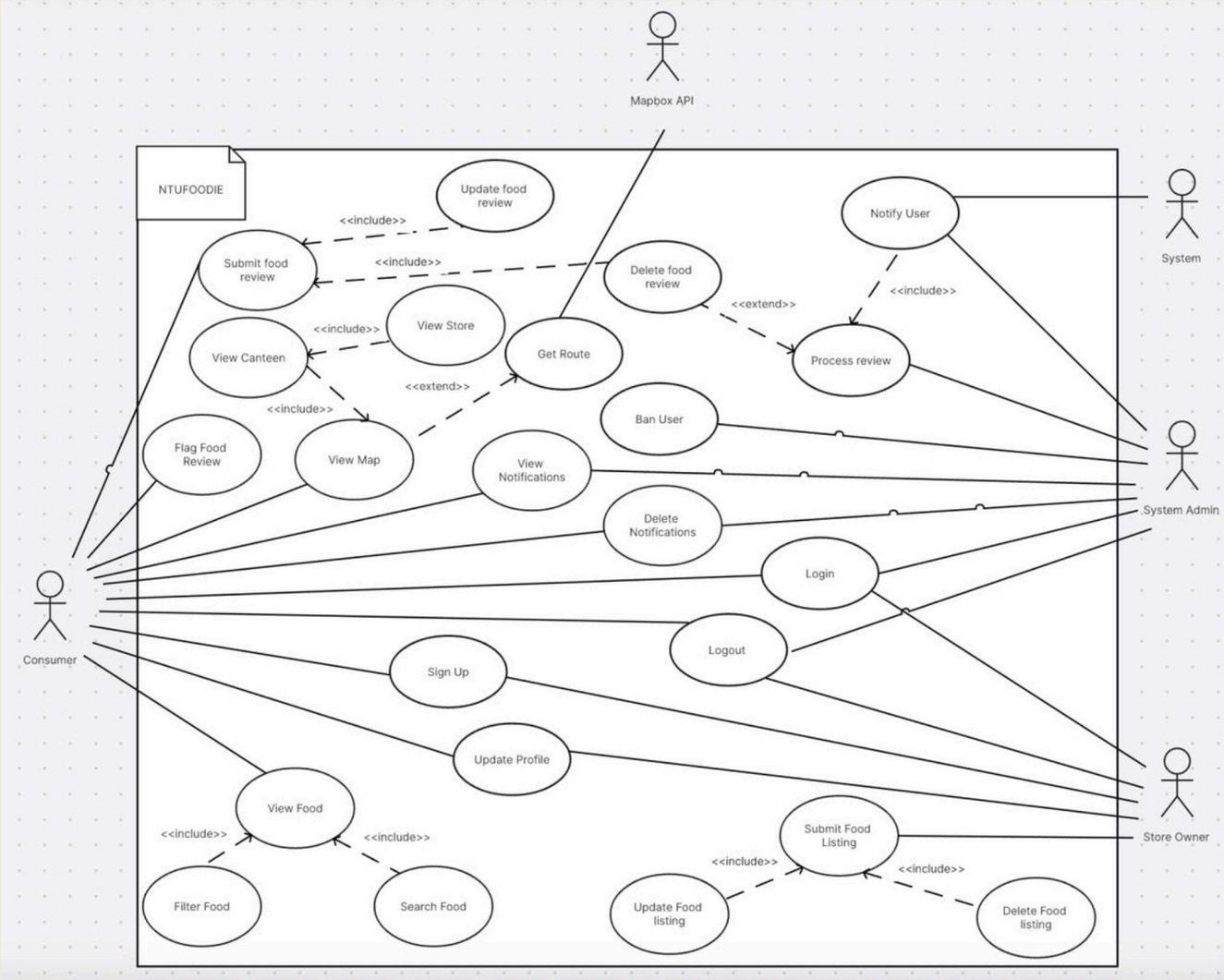
Features

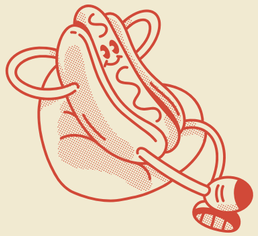


Key Features

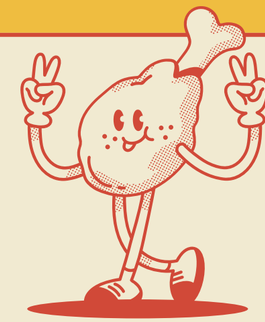
- Authentication
- Consumers
- Update Profile
- Real-time Notifications
- Live Map & Live Route
- Food Review
- Store Owners
- Food Listings
- Real-time notifications
- Admin Management
- Flagging Reviews

Overview : Use Case Diagram



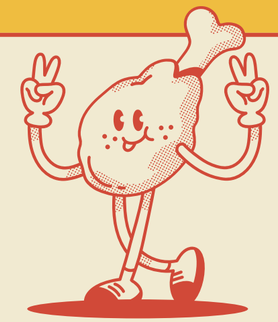


Feature List



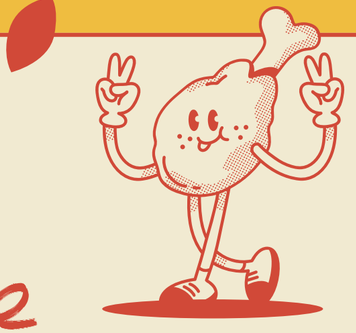
1. Authentication
    - Sign Up(for different roles - Consumer, Store Owner)
    - Login
    - Logout
2. Consumer
    - View/Search/ Filter Food
    - Submit/Update/Delete Food Review
    - Flag Food Review
    - View Map & Get Route
        - From map, view for canteen- store- food
    - View & Delete Notifications
    - Update Profile

Feature List



3. Store Owner
    - Submit/ Update Food Listing
    - Delete Food Listing
    - Update Profile
4. Admin
    - View Notifications/ Delete Notifications
    - Process Review
    - Ban User
    - Notify User

External APIs

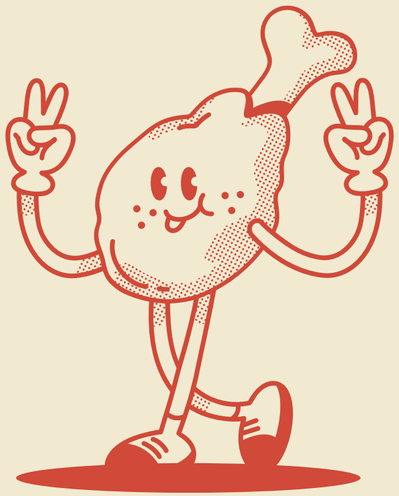


1. Mapbox APIs
    a. Live Map - Mapbox Streets
    b. Geocoding
    c. Directions

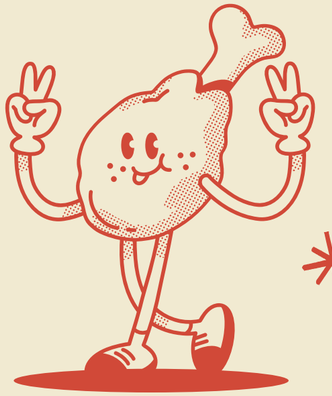
Displays registered and existing canteens within **NTU** on the map in **real-time**

NTUFOODIE.CO

Live Demo!



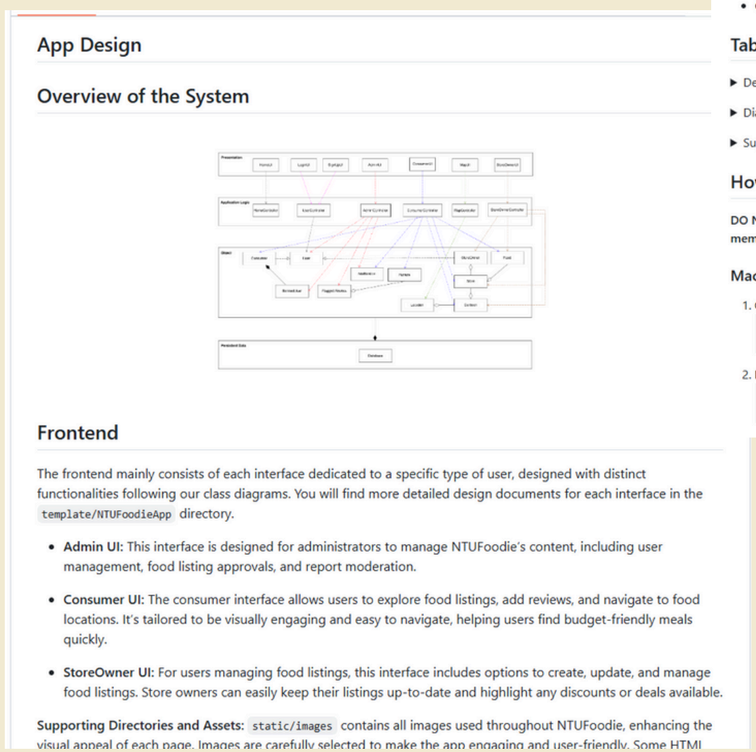
NTUFOODIE.CO



Software Engineering Practices

Documentation - README

Serves as a guide for future developers, enabling long-term consistency and showcasing good development



App Design

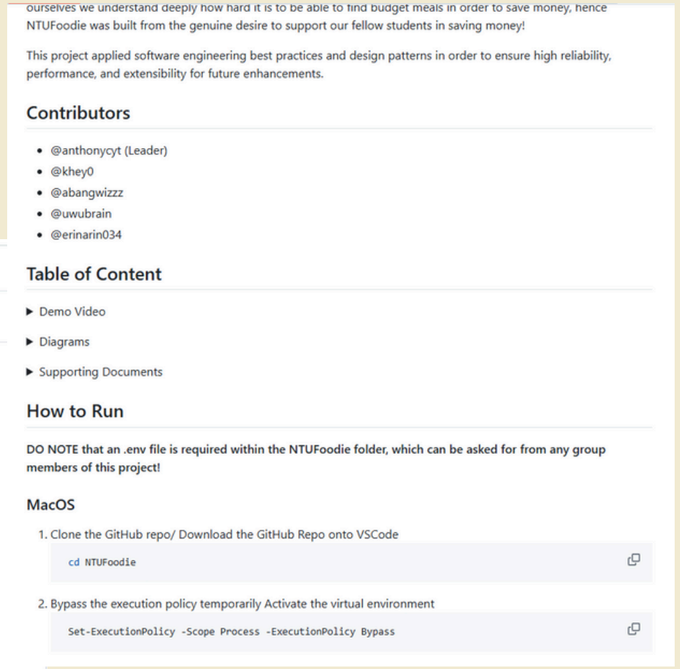
Overview of the System

Frontend

The frontend mainly consists of each interface dedicated to a specific type of user, designed with distinct functionalities following our class diagrams. You will find more detailed design documents for each interface in the `template/NTUFoodieApp` directory.

- **Admin UI:** This interface is designed for administrators to manage NTUFoodie's content, including user management, food listing approvals, and report moderation.
- **Consumer UI:** The consumer interface allows users to explore food listings, add reviews, and navigate to food locations. It's tailored to be visually engaging and easy to navigate, helping users find budget-friendly meals quickly.
- **StoreOwner UI:** For users managing food listings, this interface includes options to create, update, and manage food listings. Store owners can easily keep their listings up-to-date and highlight any discounts or deals available.

Supporting Directories and Assets: `static/images` contains all images used throughout NTUFoodie, enhancing the visual appeal of each page. Images are carefully selected to make the app engaging and user-friendly. Some HTML



ourselves we understand deeply how hard it is to be able to find budget meals in order to save money, hence NTUFoodie was built from the genuine desire to support our fellow students in saving money!

This project applied software engineering best practices and design patterns in order to ensure high reliability, performance, and extensibility for future enhancements.

Contributors

- @anthonycyt (Leader)
- @khey0
- @abangwizzz
- @uwubrain
- @erinarin034

Table of Content

- ▶ Demo Video
- ▶ Diagrams
- ▶ Supporting Documents

How to Run

DO NOTE that an .env file is required within the NTUFoodie folder, which can be asked for from any group members of this project!

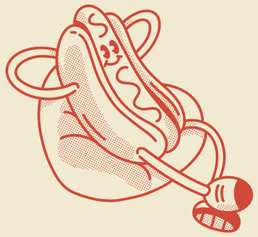
MacOS

1. Clone the GitHub repo/ Download the GitHub Repo onto VSCode

```
cd NTUFoodie
```

2. Bypass the execution policy temporarily Activate the virtual environment

```
Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass
```



Documentation - Code Comments

- **Makes code easier to understand (and readable) for enhanced collaboration**
- **For collaborates after the most recent push to understand what was added and how each new functionality works**

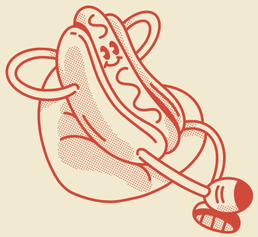
```python
urlpatterns = [
    # Consumer URL pathways
    path("home/", views.consumerHome, name="cHome"),
    path("food/", views.food, name="food"),
    path("food/<str:pk_id>/", views.fooddetail, name="foodDetail"),
```

```python
class CanteenModelTest(BaseTestData):
    @print_success
    def test_canteen_creation(self):
        self.assertIsInstance(self.canteen, Canteen)  # Test if Canteen instance is created
```

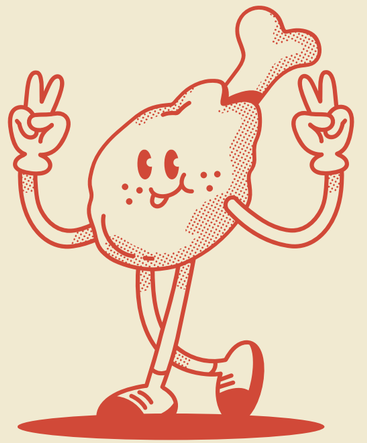
```python
# Helper decorator to print success message after each test with formatting
def print_success(func):
    def wrapper(*args, **kwargs):
        func(*args, **kwargs)
        print(f"{'-'*10} {func.__name__} passed successfully {'-'*10}\n")
    return wrapper
```

```python
def clean(self):
    cleaned_data = super().clean()
    is_discounted = cleaned_data.get('isDiscounted')
    discount_rate = cleaned_data.get('discountRate')

    # Validate discountRate only if isDiscounted is True
    if is_discounted:
        if discount_rate is None:
            self.add_error('discountRate', 'Please provide a discount rate when the item is discounted.')
        else:
            if discount_rate < 0:
                self.add_error('discountRate', 'Discount rate cannot be negative.')
            elif discount_rate >= 100:
                self.add_error('discountRate', 'Discount rate must be less than 100%.')
    else:
        # If the item is not discounted, ensure discountRate is None
        cleaned_data['discountRate'] = None
    return cleaned_data
```

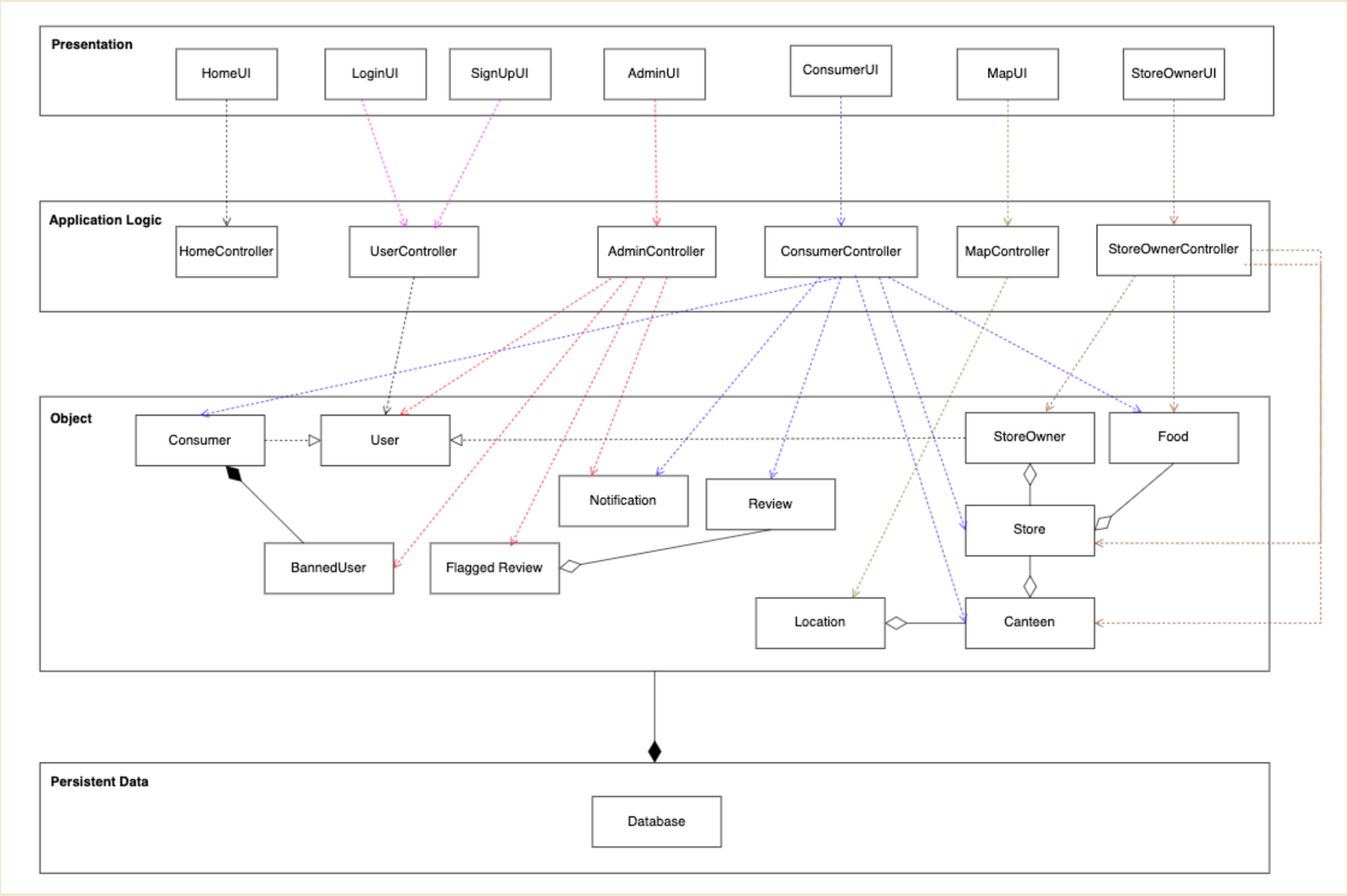


NTUFOODIE.CO

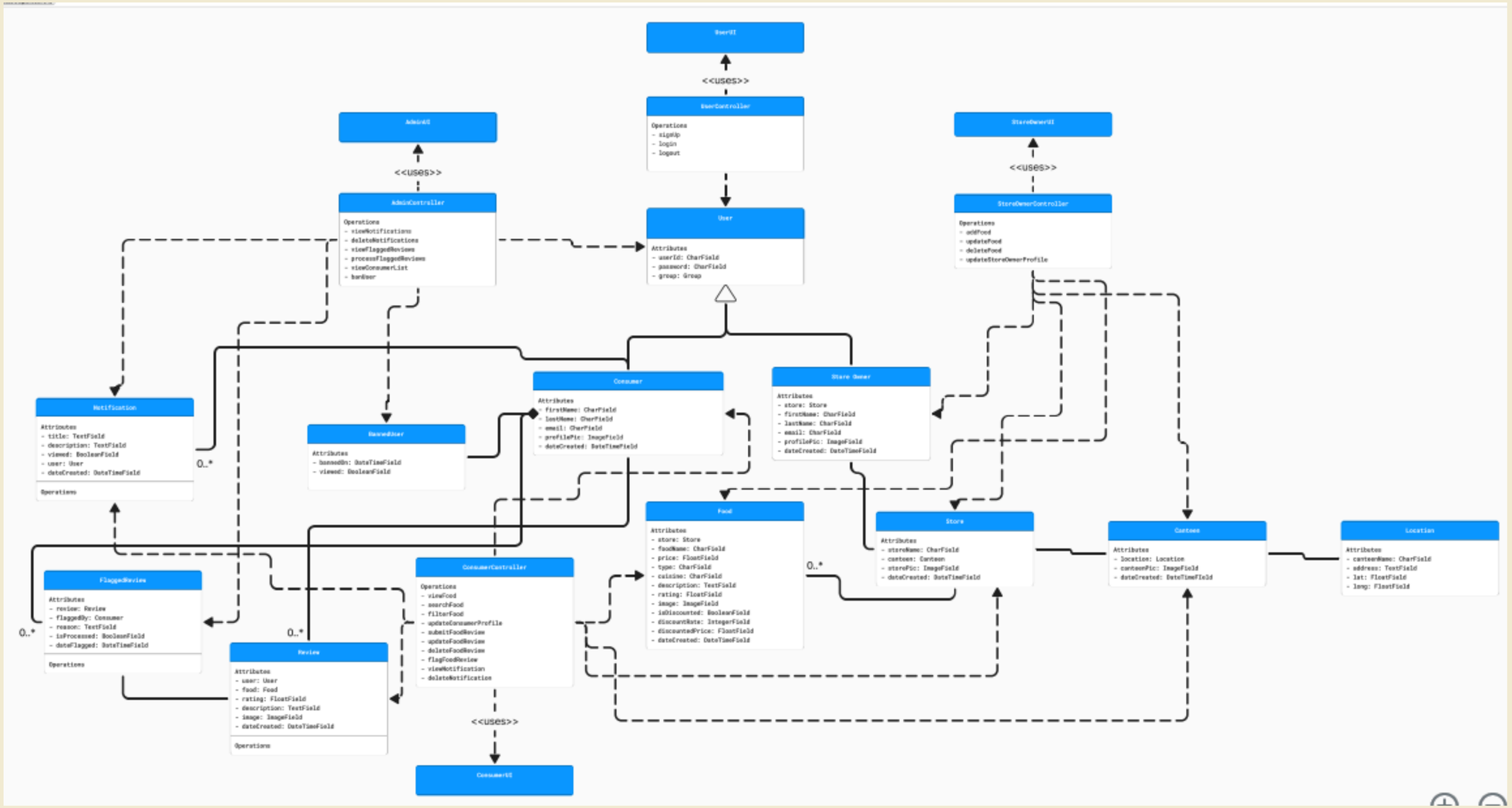


System Design

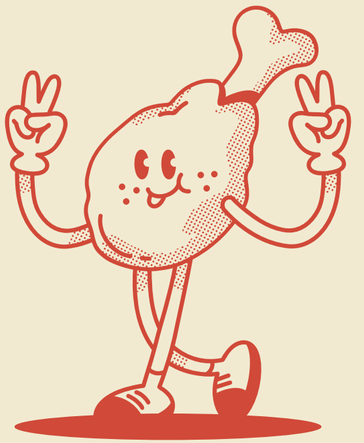
Architecture Diagram (Layered Architecture)



Class Diagram



NTUFOODIE.CO

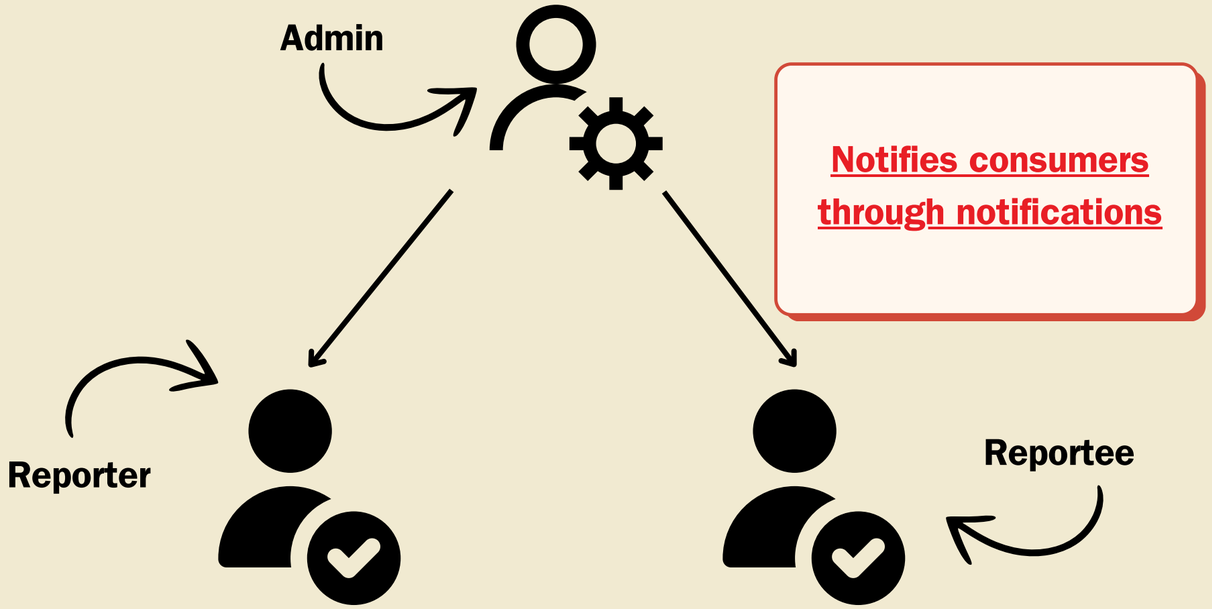


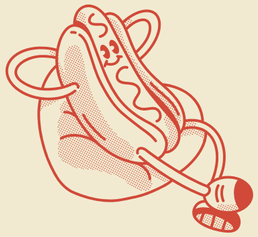
Design Patterns

Observer Pattern

Observer Pattern

- The Observer pattern is a behavioral design pattern that establishes a one-to-many relationship between objects.
- Admin would send a notification to the reporter if the flaged review is not deleted.
- Admin would send a notification to the reportee if the flagged review is deleted.

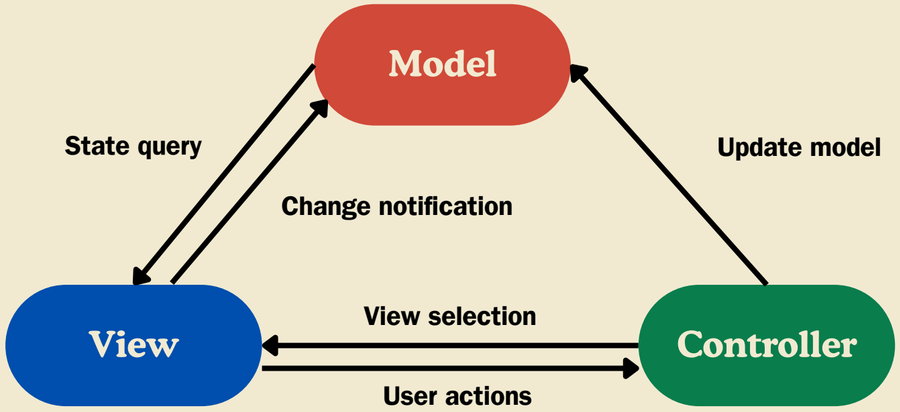




MVC Framework

How our app adheres to it:

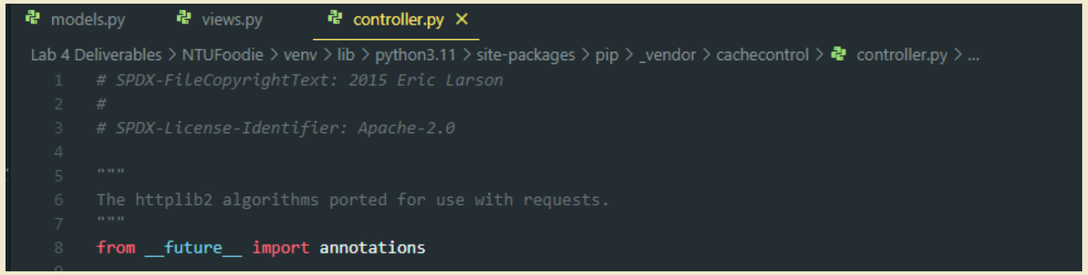
- **Model** - Manages the business logic-related data on the web-app, including images, pricing, etc. It handles the storage, retrieval, and manipulation of this information.
- **View -** The different UI elements interacted with are managed and controlled which are part of the MVC framework.
- **Controller -** Handles the execution of business logic. It processes user requests and generates responses by executing the necessary logic, which involves interacting with databases and other backend controller components.



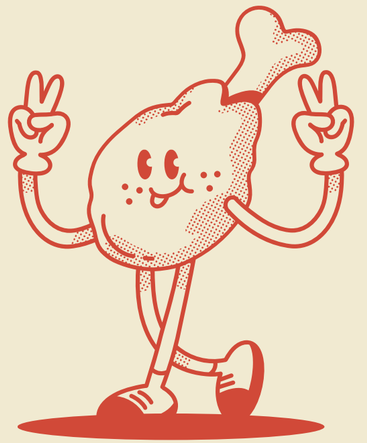
Also adheres to Single-responsibility Principle (SRP)

- Each component (Model, View, Controller) has a clear, distinct role.
- Changes in one part don't affect others directly.
- Responsibility are kept isolated.
- Makes extending the web-app alot easier.

For extensibility, flexibility, maintainability, and ease of understanding



NTUFOODIE.CO



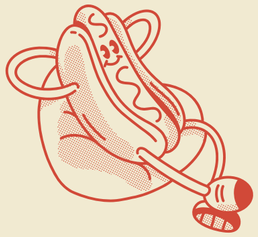
Traceability

Use-Case Description - Login

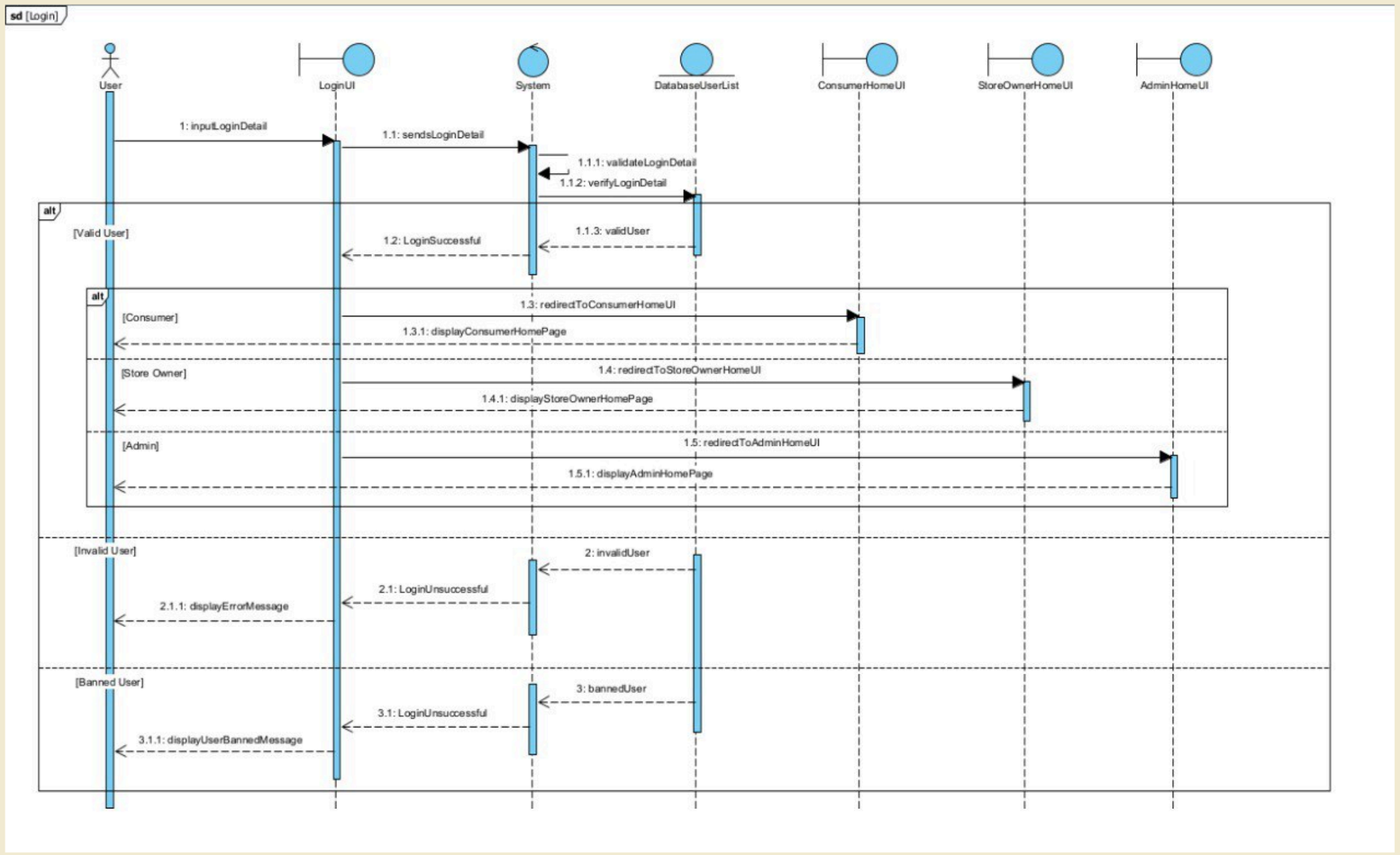
Use Case: Login

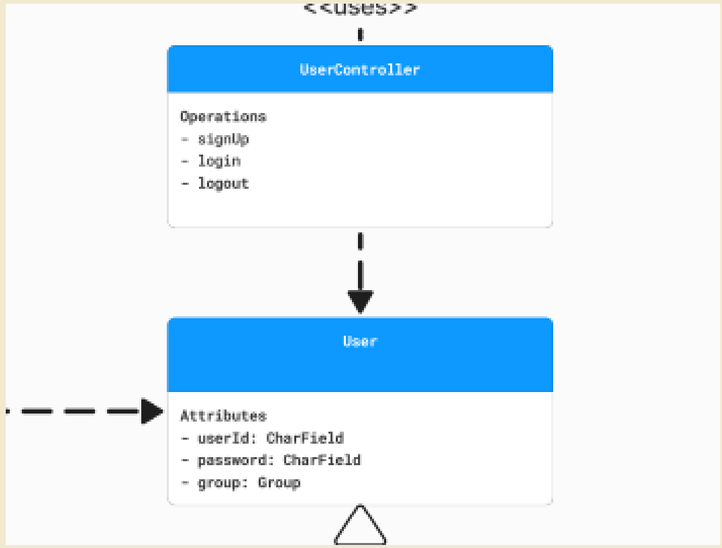
Use Case ID:	ACCOUNT_UC_2		
Use Case Name:	Login		
Created By:	Anthony	Last Updated By:	Anthony
Date Created:	03/09/2024	Date Last Updated:	23/09/2024

Actor:	User / Store Owner / System Admin / Google API
Description:	This use case allows users to log into the system to access the relevant functions according to the user's role. The various user roles that can log in are user, store owner & system admin. To log in, all users have to enter their email and password or sign in with their Google Account. Upon successful login, the system display the relevant user's home page
Preconditions:	The user has to have a valid account created beforehand
Postconditions:	The system displays the relevant homepage
Priority:	High
Frequency of Use:	High



Relevant Diagrams - Login





Visual Paradigm

Code - Login

Visual Studio Code

Linking code with comments

```
if request.user.is_authenticated:
    user = request.user
    role = user.groups.first()
    if str(role) == "consumer":
        return redirect('cHome')
    elif str(role) == "storeowner":
        return redirect('soHome')
    else:
        return redirect('aHome')
```

```
# Authenticate the user
user = authenticate(request, username=username, password=password)

if user is not None:
    role = user.groups.first()
    if str(role) == "consumer":
        login(request, user)
        return redirect('cHome')
    elif str(role) == "storeowner":
        login(request, user)
        return redirect('soHome')
    else:
        login(request, user)
        return redirect('aHome')
else:
    messages.error(request, 'Invalid username or password.')
    return render(request, "NTUFoodieApp/login.html")
```

```
# Check if the user is banned
if not user.is_active and BannedUser.objects.filter(user=user, isBanned=True).exists():
    messages.error(request, "Your account has been banned. Please contact support.")
    return render(request, "NTUFoodieApp/login.html")
```

Role-base access:

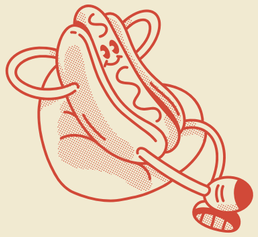
- Each role is redirected to specific pages after login
- Enhances maintainability & clarity

Secure access:

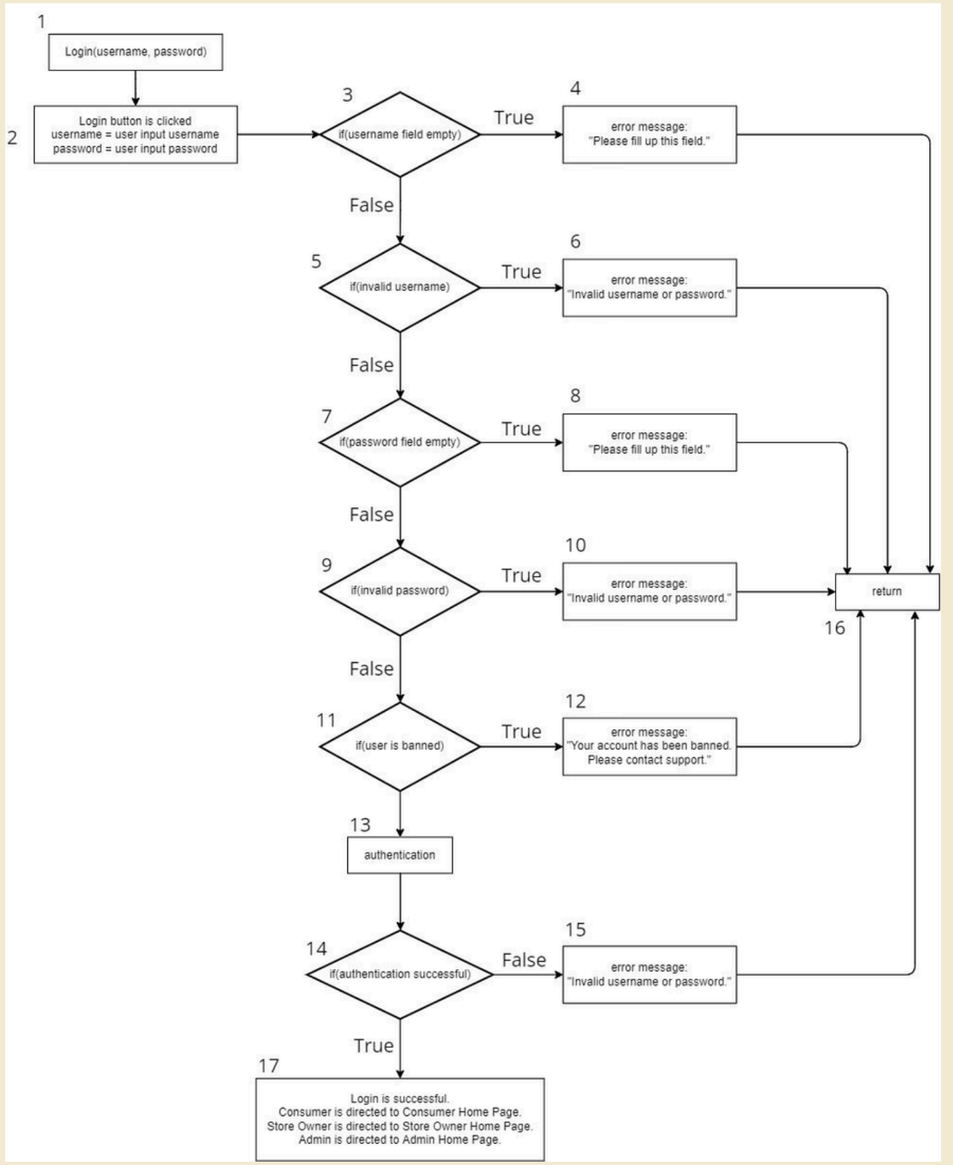
- Authentication with username & password
- Successful login: Redirect to user pages
- Unsuccessful login: Error messages to prompt relogin

Authentication for banned users:

- Additional security layer
- Prevents unauthorized access



White Box Test - Login



Visual Paradigm

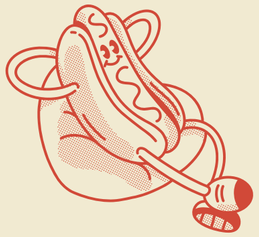
White Box Test - Login

Basic Path Testing

Cyclomatic Complexity = |Decision points| + 1 = 6 + 1 = 7

Basis Paths:

1. **Baseline path: 1,2,3,5,7,9,11,13,14,17**
2. **Basis path 2: 1,2,3,4,16**
3. **Basis path 3: 1,2,3,5,6,16**
4. **Basis path 4: 1,2,3,5,7,8,16**
5. **Basis path 5: 1,2,3,5,7,9,10,16**
6. **Basis path 6: 1,2,3,5,7,9,11,12,16**
7. **Basis path 7: 1,2,3,5,7,9,11,13,14,15,16**

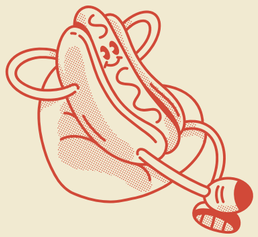


Black Box Test - Login

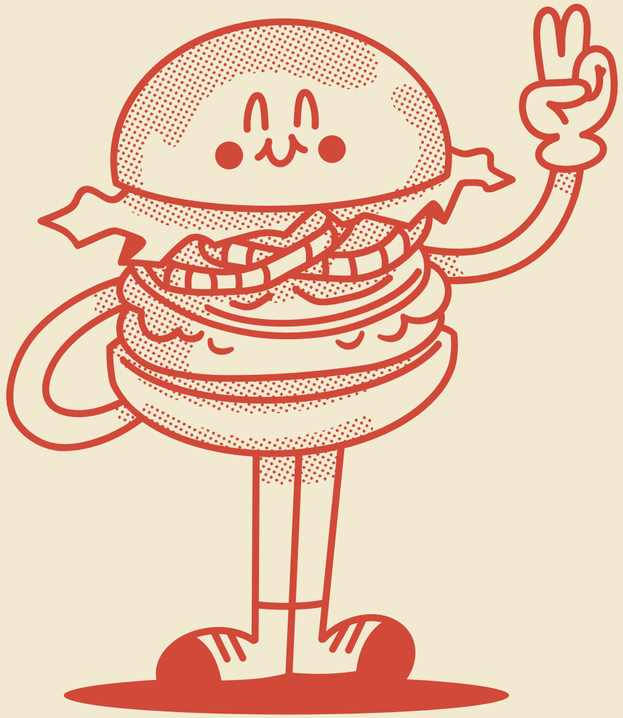
d) Sign In Function (User Controller)

Input parameter: Username, Password (**Fields in red belongs to invalid class**)

Test Case ID	Description	Test Type	Priority	Input	Expected Output	Actual Output
TC_L_001	Valid login	Equivalence Partitioning	High	**(All valid inputs)** Name: "teststoreowner" Password: "testpassword"	Redirects to home page for store owners	Redirects to home page for store owners
TC_L_002	Empty username	Negative Testing	High	**(Invalid Name, Valid Password)** Name: "" Password: "testpassword"	System outputs "Invalid username or password."	System outputs "Invalid username or password."
TC_L_003	Invalid username	Negative Testing	High	**(Invalid Name, Valid Password)** Name: "teststoreowners" Password: "testpassword"	System outputs "Invalid username or password."	System outputs "Invalid username or password."
TC_L_004	Empty password	Negative Testing	High	**(Invalid Password, Valid Name)** Name: "teststoreowner" Password: ""	System outputs "Invalid username or password."	System outputs "Invalid username or password."
TC_L_005	Wrong password	Negative Testing	High	**(Invalid Password, Valid Name)** Name: "teststoreowner" Password: "wrongpass"	System outputs "Invalid username or password."	System outputs "Invalid username or password."



Thank You!



20 24

NTUFOODIE.CO

100% BY STUDENTS