**Software** Phase2
Final Document

# Clothes Trading website.


**By: Nourhan Mohamed 202100538**

**Nada Nabil 202101220**

**Rghda Salah 202101510**

**Tasneem Mohammed 202101031**

## Our Development process

**As a team, we chose to adopt the Scrum development process for our project.**

**We believe that Scrum is the most suitable development process for our project due to the following reasons:**

- **Scrum allows us to change our requirements and priorities throughout the project. We did so by working in short iterations (sprints), so in each sprint we updated our iterated SRS and SDS with the changelog, and this enhanced the flexibility.**
- **Scrum provides mechanisms to achieve cooperation and visualization and there are many tools to use. So we used Jira as our Scrum management tool, we prioritize our backlogs, and start to assign tasks from our scrum master to us, then we can comment on each task and see project progress as a visualized done or in progress tasks.**

## Backlogs and Team Cooperation

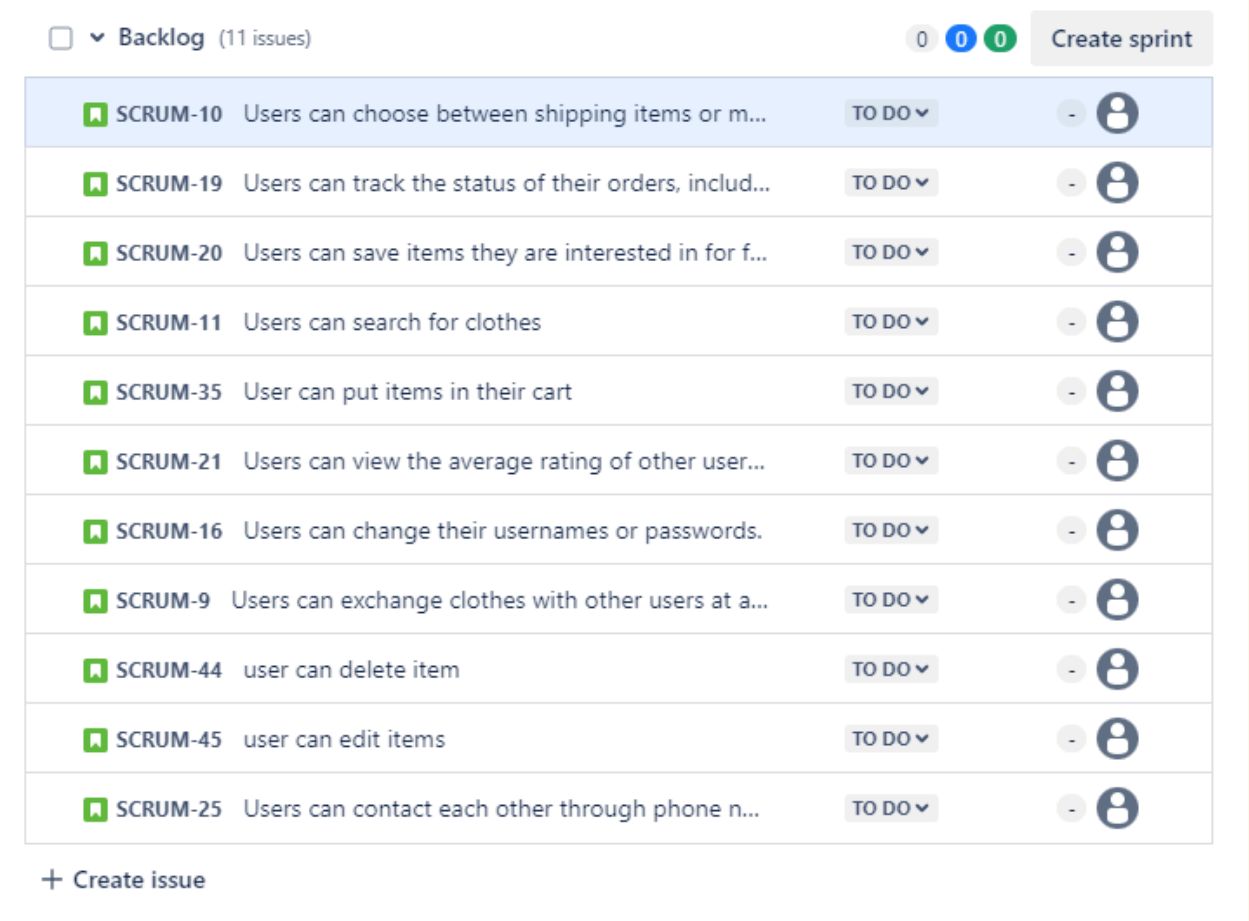
1. **All the created tasks with deadlines, Assignee, state and actions :**

Jira
Sorted by: Created descending
1–30 of 30 as at: 13/May/24 10:14 PM

| T | Key | Summary | Assignee | Reporter | P | Status | Resolution | Created | Updated | Due |
|---|---|---|---|---|---|---|---|---|---|---|
| | SCRUM-45 | user can edit items | Nada Nabil | Tasneem | = | TO DO | Unresolved | 11/May/24 | 13/May/24 | |
| | SCRUM-44 | user can delete item | Tasneem | Tasneem | = | TO DO | Unresolved | 11/May/24 | 13/May/24 | |
| | SCRUM-39 | adding the advanced options for the product | Tasneem | Tasneem | = | DONE | Done | 10/May/24 | 11/May/24 | |
| | SCRUM-38 | adding product to the database by the basic info with no complications | Tasneem | Tasneem | = | DONE | Done | 10/May/24 | 10/May/24 | |
| | SCRUM-37 | All the product data for add item must be saved in the database | Nourhan Mohamed 202100538 | Tasneem | = | DONE | Done | 10/May/24 | 11/May/24 | |
| | SCRUM-36 | User Can Browse all items | Nourhan Mohamed 202100538 | Nourhan Mohamed 202100538 | = | DONE | Done | 10/May/24 | 10/May/24 | |
| | SCRUM-35 | User can put items in their cart | Nourhan Mohamed 202100538 | Nourhan Mohamed 202100538 | = | TO DO | Unresolved | 09/May/24 | 13/May/24 | |
| | SCRUM-33 | Users can add elements to their carts. | Nourhan Mohamed 202100538 | Tasneem | = | DONE | Done | 09/May/24 | 11/May/24 | |
| | SCRUM-32 | design a logo | Tasneem | Tasneem | = | DONE | Done | 04/May/24 | 10/May/24 | |
| | SCRUM-31 | start the database | Nada Nabil | Tasneem | = | DONE | Done | 04/May/24 | 05/May/24 | |
| | SCRUM-30 | start the frontend | Tasneem | Tasneem | = | DONE | Done | 04/May/24 | 09/May/24 | |
| | SCRUM-29 | choose between templates | s-ghda.ahmed | Tasneem | = | DONE | Done | 04/May/24 | 11/May/24 | |
| | SCRUM-28 | choose the theme, visualized identity, colors and style for the website | Nada Nabil | Tasneem | = | DONE | Done | 04/May/24 | 11/May/24 | |
| | SCRUM-27 | Transferring business requirements to technical one | Nourhan Mohamed 202100538 | Tasneem | = | DONE | Done | 04/May/24 | 04/May/24 | |
| | SCRUM-25 | Users can contact each other through phone number | s-ghda.ahmed | Tasneem | = | TO DO | Unresolved | 04/May/24 | 13/May/24 | |
| | SCRUM-24 | Users can contact the website managers through emails for inquiries. | Nada Nabil | Tasneem | = | DONE | Done | 04/May/24 | 10/May/24 | |
| | SCRUM-22 | Users can browse the website as guests without creating an account | Nourhan Mohamed 202100538 | Tasneem | = | DONE | Done | 04/May/24 | 10/May/24 | |
| | SCRUM-21 | Users can view the average rating of other users based on previous exchanges and ratings. | Tasneem | Tasneem | = | TO DO | Unresolved | 04/May/24 | 13/May/24 | |
| | SCRUM-20 | Users can save items they are interested in for future reference. | Nourhan Mohamed 202100538 | Tasneem | = | TO DO | Unresolved | 04/May/24 | 13/May/24 | |
| | SCRUM-19 | Users can track the status of their orders, including shipping details and delivery. | s-ghda.ahmed | Tasneem | = | TO DO | Unresolved | 04/May/24 | 13/May/24 | |
| | SCRUM-17 | Users should be able to follow the website on social media platforms. | Nada Nabil | Tasneem | = | DONE | Done | 04/May/24 | 11/May/24 | |
| | SCRUM-16 | Users can change their usernames or passwords. | s-ghda.ahmed | Tasneem | = | TO DO | Unresolved | 04/May/24 | 13/May/24 | |
| | SCRUM-15 | The website should provide advanced search options such as filtering by brand, size, or color. | Nourhan Mohamed 202100538 | Tasneem | = | DONE | Done | 04/May/24 | 11/May/24 | |
| | SCRUM-13 | Users can rate the state of clothes on a scale of 1-5. | s-ghda.ahmed | Tasneem | = | DONE | Done | 04/May/24 | 11/May/24 | |
| | SCRUM-12 | Users can log in and manage their profiles, including personal information. | s-ghda.ahmed | Tasneem | = | DONE | Done | 04/May/24 | 11/May/24 | |
| | SCRUM-11 | Users can search for clothes | Nada Nabil | Tasneem | = | TO DO | Unresolved | 04/May/24 | 13/May/24 | |
| | SCRUM-10 | Users can choose between shipping items or meeting up with other users for trading. | Tasneem | Tasneem | = | TO DO | Unresolved | 04/May/24 | 13/May/24 | |
| | SCRUM-9 | Users can exchange clothes with other users at a similar rate. | Nada Nabil | Tasneem | = | TO DO | Unresolved | 04/May/24 | 13/May/24 | |
| | SCRUM-8 | Users can provide a description of their clothes | s-ghda.ahmed | Tasneem | = | DONE | Done | 04/May/24 | 11/May/24 | |
| | SCRUM-6 | Users can create an account on the website. | s-ghda.ahmed | Tasneem | = | DONE | Done | 04/May/24 | 11/May/24 | |

## 2. Last Backlog before the final Phase :

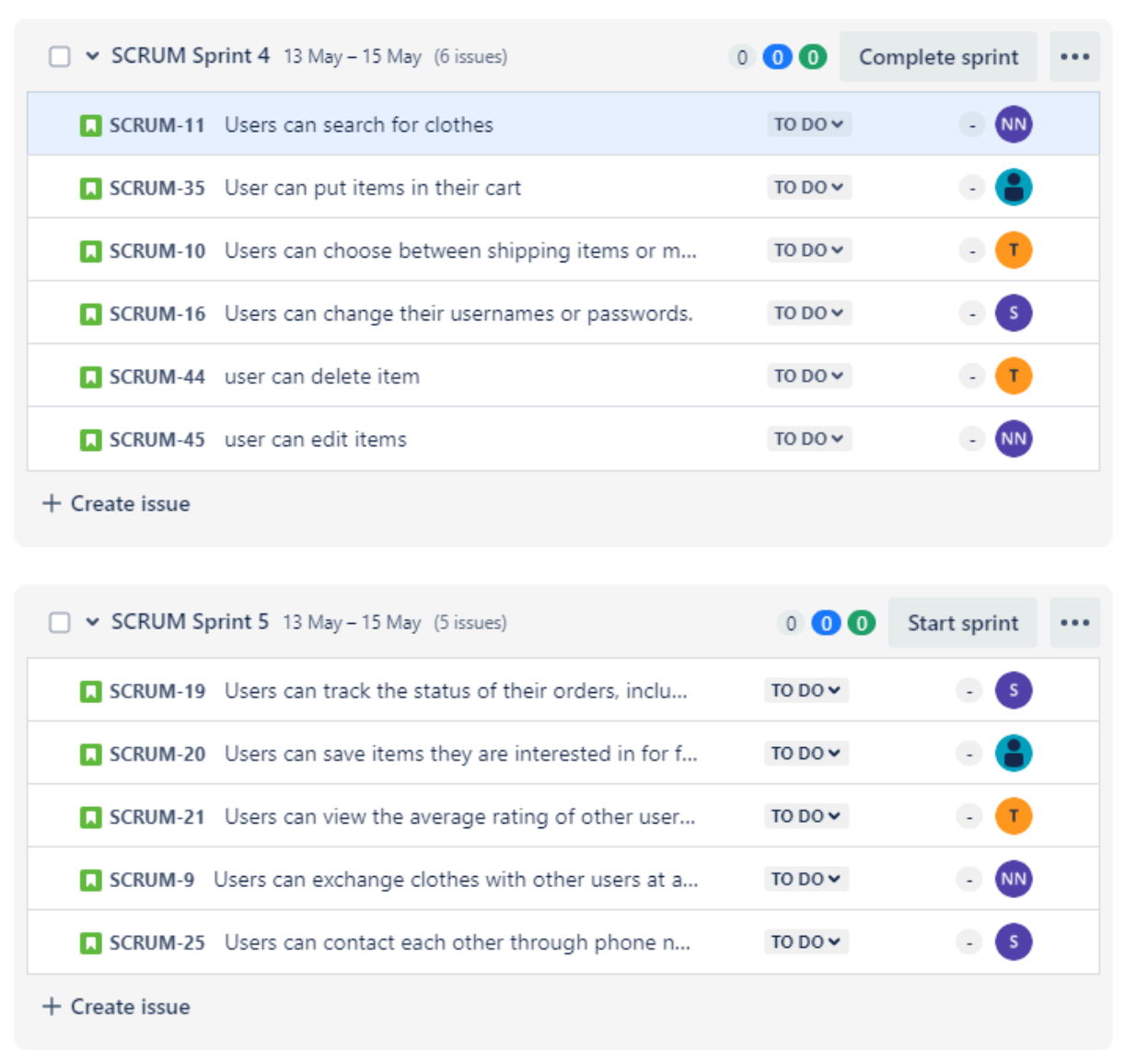
In this Backlog, these user stories are to be implemented in the final phase so we were going to assign it to a sprint first then to members




Here, we were deciding to implement it on one sprint or two sprints, but we will discuss that later in this document.

## 3. The last two sprints :

We assigned the tasks and started sprint 4 which will end on 15 May then we will assign it as completed and start the last sprint.



## 4. Done tasks before the last phase (More than 70% of the requirements):There are 20 Tasks done here before Phase 4.

List

Give feedback

Search list

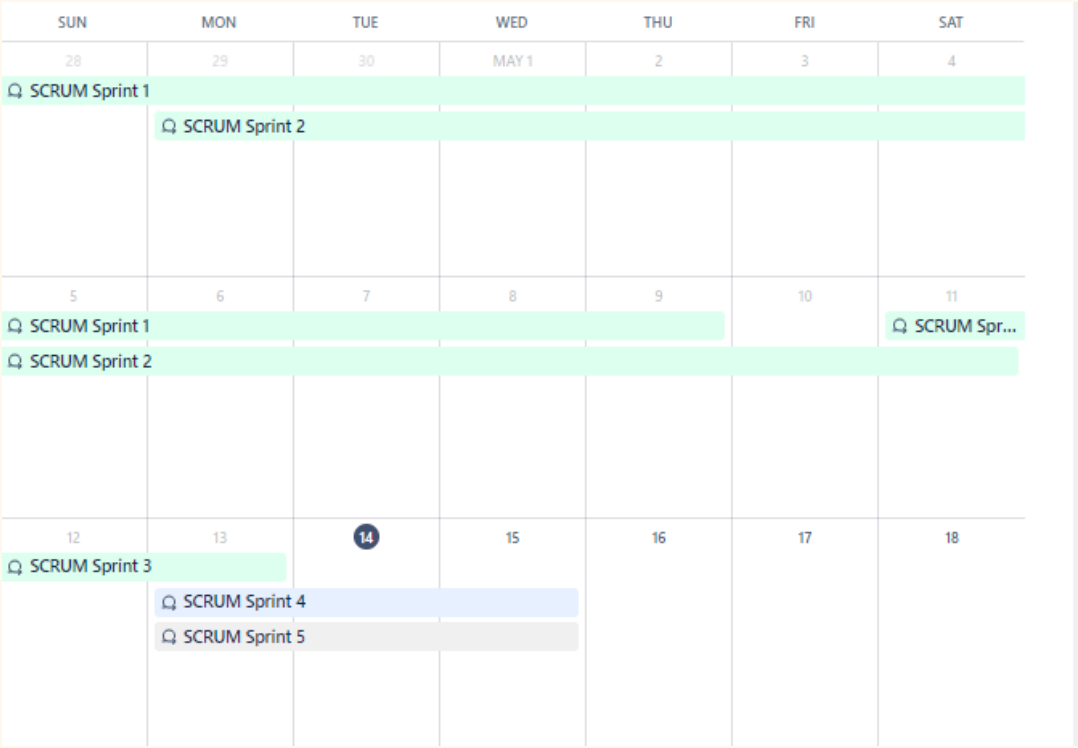
Share | 1 filter applied | Group | More

| Type | Key | Summary | Status | Sprint | Assignee | Due date | Labels | Created | Updated | Reporter |
|---|---|---|---|---|---|---|---|---|---|---|
| | SCRUM-12 | Users can log in and manage their profiles, including person... | DONE | SCRUM Sprin... | s-rghda.ahmed | | | May 4, 2024 | May 11, 2024 | Tasneem |
| | SCRUM-6 | Users can create an account on the website. | DONE | SCRUM Sprin... | s-rghda.ahmed | | | May 4, 2024 | May 11, 2024 | Tasneem |
| | SCRUM-17 | Users should be able to follow the website on social media p... | DONE | SCRUM Sprin... | Nada Nabil | | | May 4, 2024 | May 11, 2024 | Tasneem |
| | SCRUM-22 | Users can browse the website as guests without creating an ... | DONE | SCRUM Sprin... | Nourhan Mohamed ... | | | May 4, 2024 | May 10, 2024 | Tasneem |
| | SCRUM-27 | Transferring business requirements to technical one | DONE | SCRUM Sprin... | Nourhan Mohamed ... | | | May 4, 2024 | May 4, 2024 | Tasneem |
| | SCRUM-28 | choose the theme, visualized identity, colors and style for the... | DONE | SCRUM Sprin... | Nada Nabil | | | May 4, 2024 | May 11, 2024 | Tasneem |
| | SCRUM-29 | choose between templates | DONE | SCRUM Sprin... | s-rghda.ahmed | | | May 4, 2024 | May 11, 2024 | Tasneem |
| | SCRUM-31 | start the database | DONE | SCRUM Sprin... | Nada Nabil | | | May 4, 2024 | May 5, 2024 | Tasneem |
| | SCRUM-30 | start the frontend | DONE | SCRUM Sprin... | Tasneem | | | May 4, 2024 | May 9, 2024 | Tasneem |
| | SCRUM-24 | Users can contact the website managers through emails for i... | DONE | SCRUM Sprin... | Nada Nabil | | | May 4, 2024 | May 10, 2024 | Tasneem |
| | SCRUM-32 | design a logo | DONE | SCRUM Sprin... | Tasneem | | | May 4, 2024 | May 10, 2024 | Tasneem |
| | SCRUM-8 | Users can provide a description of their clothes | DONE | SCRUM Sprin... | s-rghda.ahmed | | | May 4, 2024 | May 11, 2024 | Tasneem |
| | SCRUM-13 | Users can rate the state of clothes on a scale of 1-5. | DONE | SCRUM Sprin... | s-rghda.ahmed | | | May 4, 2024 | May 11, 2024 | Tasneem |
| | SCRUM-37 | All the product data for add item must be saved in the datab... | DONE | SCRUM Sprin... | Nourhan Mohamed ... | | | May 10, 2024 | May 11, 2024 | Tasneem |
| | SCRUM-15 | The website should provide advanced search options such as... | DONE | SCRUM Sprin... | Nourhan Mohamed ... | | | May 4, 2024 | May 11, 2024 | Tasneem |
| | SCRUM-33 | Users can add elements to their carts. | DONE | SCRUM Sprin... | Nourhan Mohamed ... | | | May 9, 2024 | May 11, 2024 | Tasneem |
| | SCRUM-16 | Users can change their usernames or passwords. | DONE | SCRUM Sprint 4 | Tasneem | | | May 4, 2024 | May 14, 2024 | Tasneem |
| | SCRUM-36 | User Can Browse all Items | DONE | SCRUM Sprin... | Nourhan Mohamed ... | | | May 10, 2024 | May 10, 2024 | Nourhan Mohamed ... |
| | SCRUM-38 | adding product to the database by the basic info with no co... | DONE | SCRUM Sprin... | Tasneem | | | May 10, 2024 | May 10, 2024 | Tasneem |
| | SCRUM-39 | adding the advanced options for the product | DONE | SCRUM Sprin... | Tasneem | | | May 10, 2024 | May 11, 2024 | Tasneem |

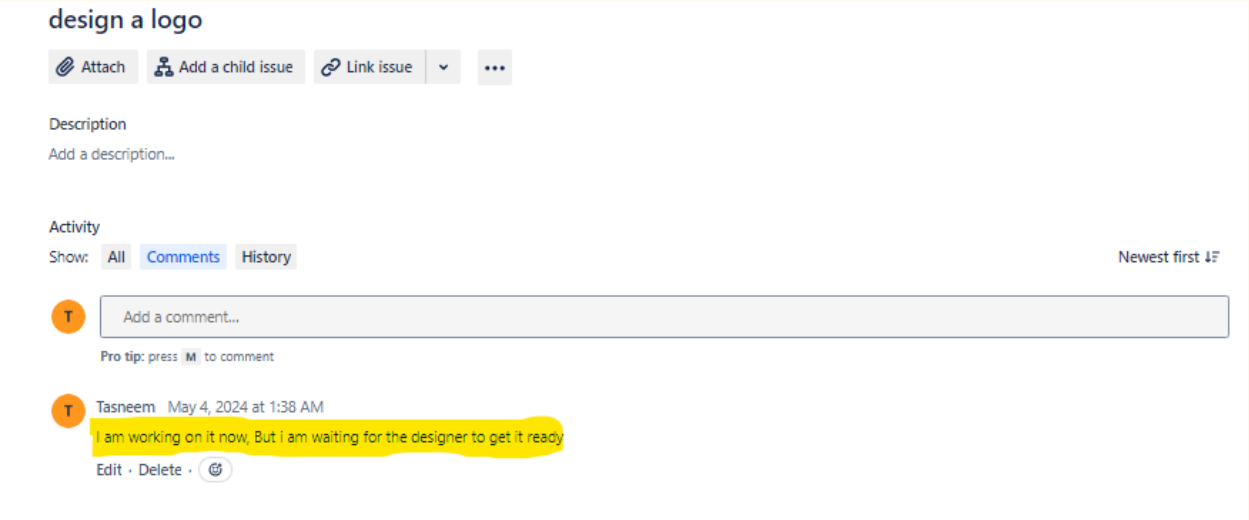
+ Create

## 5. Calendar and Time line from sprint 1 to 5:

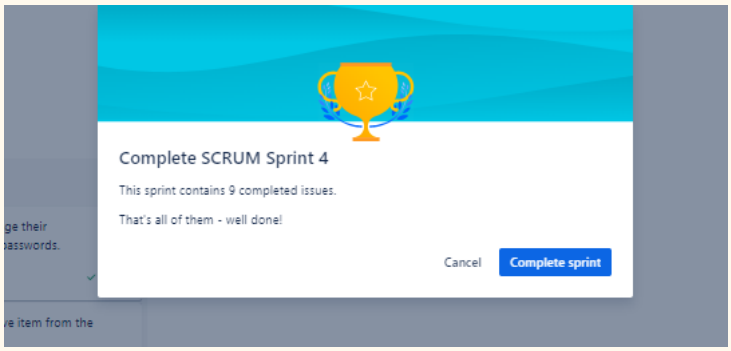
Starts from 28 April to 18 Mat

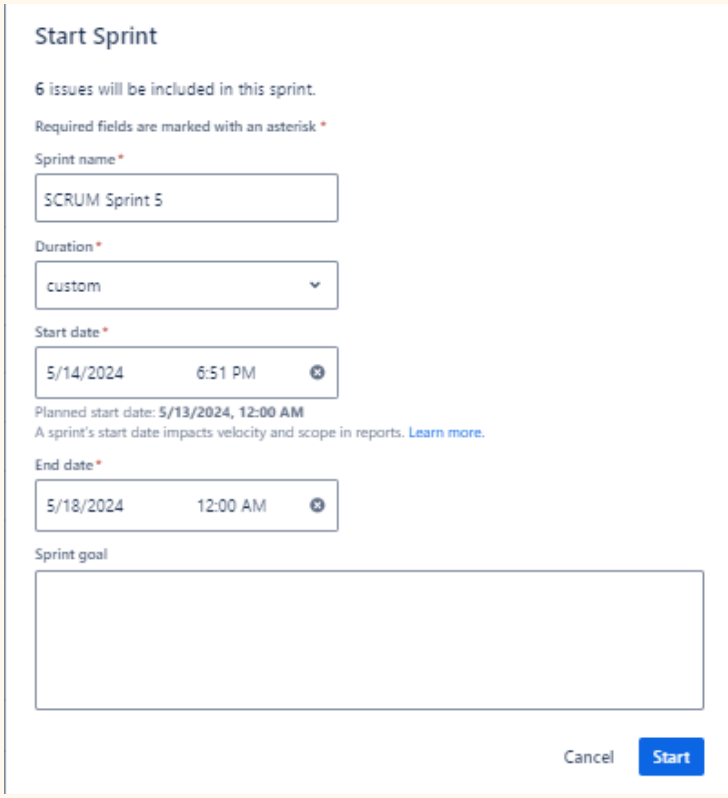


## 6. Communication and Connection in a formal way:




## 7. Start sprint when Completing the previous one:



# OO Design, Architecture and Design specification

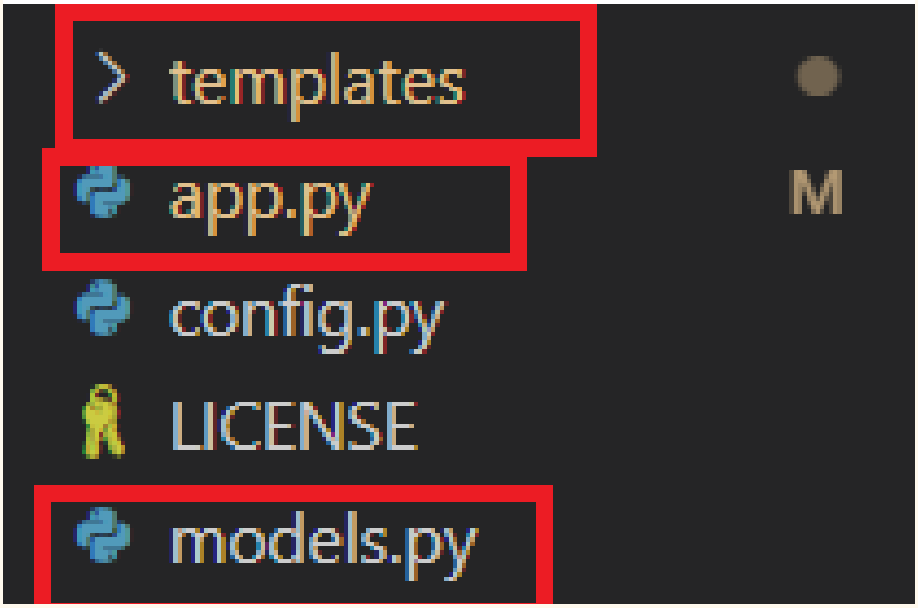
## 1. Architecture: We used MVC (Model-View-Controller )

- **Model:**Represents the data and the business logic of the application, and The models interact with the database using SQLAlchemy, they are all found in **Model.py** file.
- **View:**Represents the presentation layer, responsible for displaying the data, and they are all the html files that represent the interface.
- **Controller**:Manages the user input, interacts with the model, and renders the appropriate view, they are all the pages functions found in **app.py** file

## 2. What is the rationale for our architecture

We choose this architecture especially as it achieves **Separation of Concerns**, how?

- Database operations are handled by models
- Request handling and business logic are managed by controllers.
- The user interface is rendered through templates.



### 3. Session management:

User sessions are managed to keep track of logged-in users and their interactions with the system (for example: adding items to the cart, applying promo codes).

## Design Patterns

### 1. Factory Method Pattern:

**We created two factories: ItemFactory and MessageFactory classes:**

- These classes are the factories for creating instances of Item and ContactMessage objects. This pattern encapsulates the object creation logic within the factory so we can ensure that the object creation process is consistent and can easily be modified or extended without changing the code that uses the objects.

```
#1. Factory design Pattern

class ItemFactory:
    @staticmethod
    def create_item(name, description,
                    price, image,
                    quantity, rate,
                    category_id,
                    user_id):
        return Item(name=name,
                    description=description,
                    price=price,
                    image=image,
                    quantity=quantity,
                     rate=rate,
                     category_id=category_id,
                     user_id=user_id)

class MessageFactory:
    @staticmethod
    def create_message(name,
                       email,
                       message):
        return ContactMessage(name=name,
                              email=email,
                               message=message)
```

## 2. Strategy Pattern:

**We created three strategies :CheckoutStrategy, DefaultCheckoutStrategy, and DiscountCheckoutStrategy**

- We used  this pattern because it allows the client code to select a specific strategy at runtime without directly implementing it, ex: CheckoutStrategy: This abstract class defines a strategy interface for applying discounts to the total price during checkout, DefaultCheckoutStrategy: A concrete strategy that applies no discount, DiscountCheckoutStrategy: A concrete strategy that applies a discount to the total price based on a given percentage. This allows the system to easily switch between different discount strategies without modifying the checkout logic.

```
#2. Strategy design Pattern

class CheckoutStrategy:
    def apply_strategy(self, total_price):
        raise NotImplementedError

class DefaultCheckoutStrategy(CheckoutStrategy):
    def apply_strategy(self, total_price):
        return total_price

class DiscountCheckoutStrategy(CheckoutStrategy):
    def __init__(self, discount_percentage):
        self.discount_percentage = discount_percentage

    def apply_strategy(self, total_price):
        # Calculate the discounted price
        discount_amount = total_price * (self.discount_percentage / 100)
        discounted_price = total_price - discount_amount
        return discounted_price
```

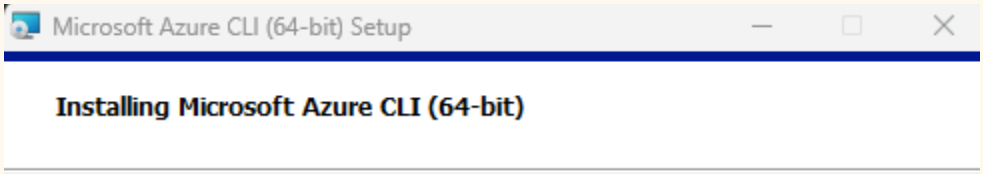
## 3. Singleton Pattern:

This  ensures that only one instance of the Flask application is created, providing global access to request handling and routing

```
#Singlton
app = Flask(__name__) #application instance.
```

# Deployment model using Docker

We choose Docker because it allows us to package our application and its dependencies into a standardized unit called a container, ensuring consistency across different environments and simplifying deployment so that the application can be runned along any device without too many dependencies and installations.

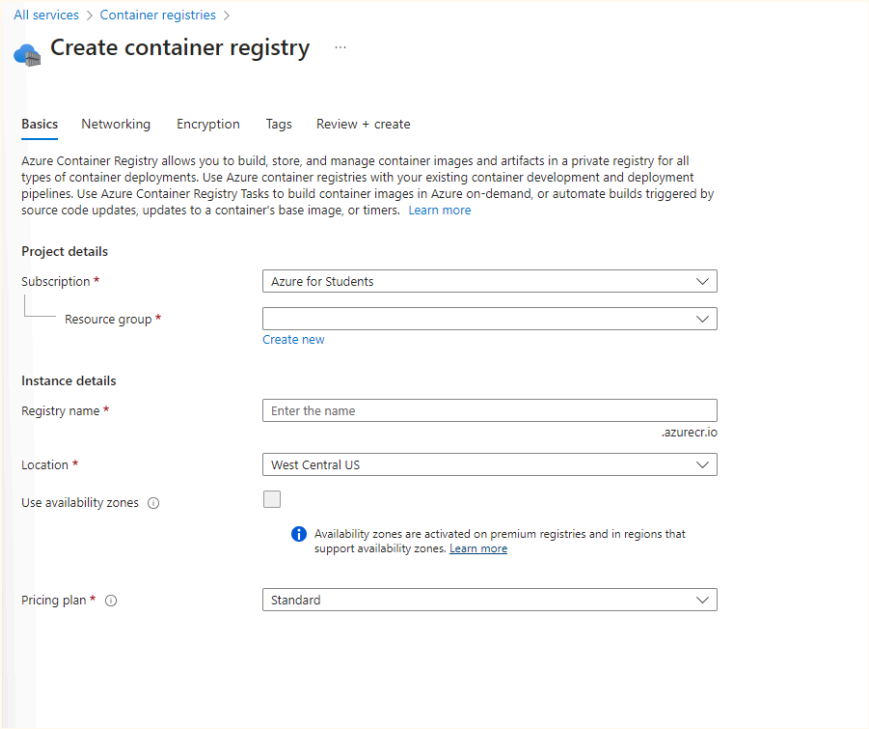

**Dockerfile Configuration:** Our Dockerfile defines the steps for building the Docker image. It starts with a base Python image, installs the necessary dependencies using apt-get, sets up the working directory, copies the application code, and exposes port 5000 for communication.

```
FROM python:3.12-slim

# Install dependencies
RUN apt-get update && \
    apt-get install -y pkg-config build-essential libmariadb-dev-compat libmariadb-dev && \
    rm -rf /var/lib/apt/lists/*

# Set the working directory
WORKDIR /app

# Copy requirements.txt and install Python dependencies
COPY requirements.txt .
RUN pip install -r requirements.txt

# Copy the rest of the application code
COPY . .

# Expose the port the app runs on
EXPOSE 5000

# Command to run the Flask app
CMD ["python", "app.py"]
```

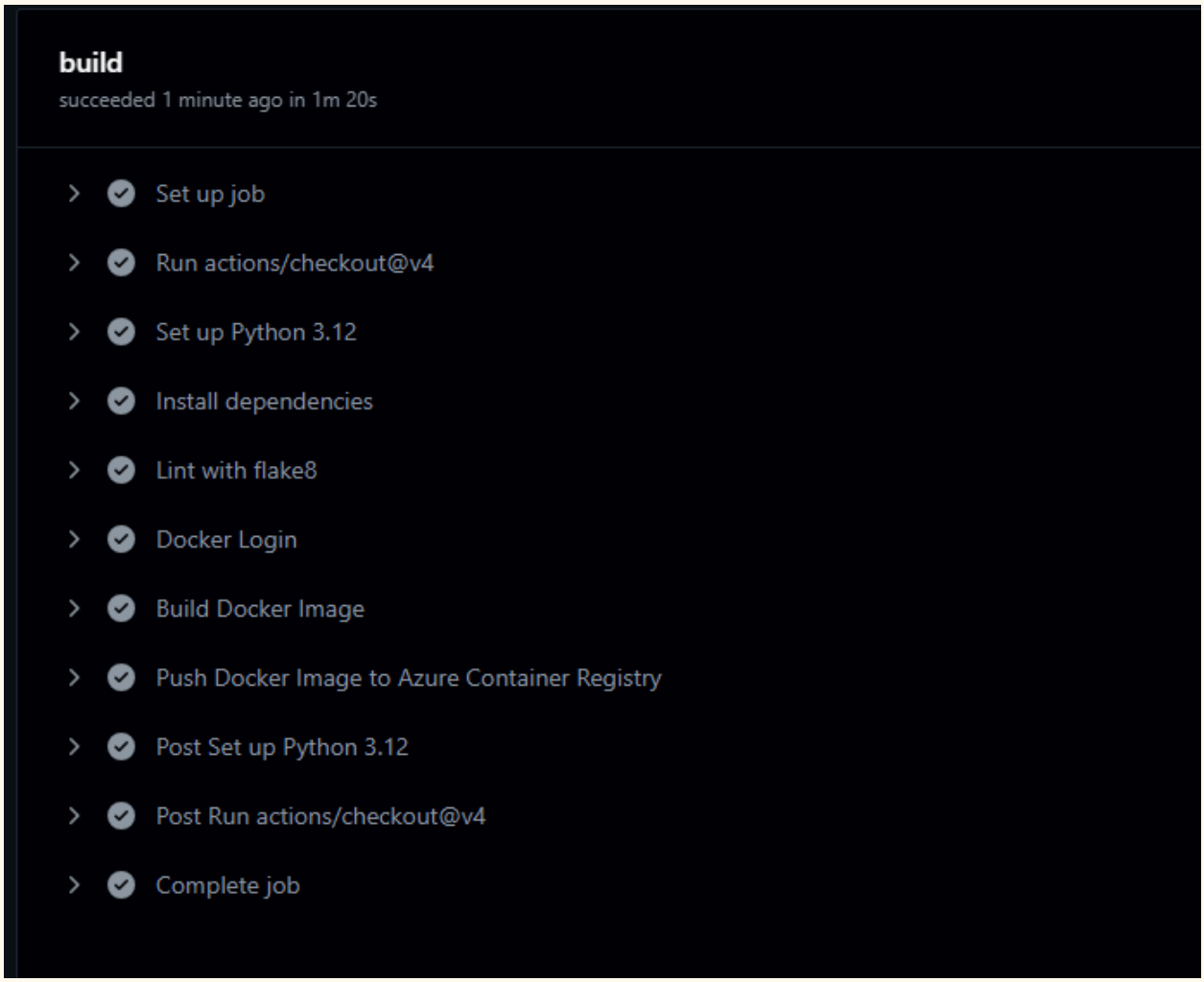
**Building and Pushing Docker Images:** We use the docker build command to build the Docker image locally. Once built, the image is pushed to our Azure Container Registry using the docker push command, making it available for deployment.

```
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
newdocker    latest    2a3d3c7b7626   55 minutes ago   792MB
PS C:\Users\Tassneem> docker tag ^C
PS C:\Users\Tassneem> docker tag newdocker badelhadocker.azurecr.io/v1
PS C:\Users\Tassneem> docker push badelhadocker.azurecr.io/v1
Using default tag: latest
The push refers to repository [badelhadocker.azurecr.io/v1]
42ff22a94d5e: Preparing
a19f2aa46c6c: Preparing
ff98a0867a14: Preparing
7712f0cacbd7: Preparing
4e6f99c1668f: Preparing
2eb7c5261c95: Waiting
a9435642a9e3: Waiting
b3232f6d8e2e: Waiting
146826fa3ca0: Waiting
5d4427064ecc: Waiting
unauthorized: {"errors":[{"code":"UNAUTHORIZED","message":"authentication required, visit https://aka.
ms/acr/authorization for more information."}]}
PS C:\Users\Tassneem> docker login badelhadocker.azurecr.io
Username: BadelhaDocker
Password:

Error response from daemon: Get "https://badelhadocker.azurecr.io/v2/": unauthorized: {"errors":[{"cod
e":"UNAUTHORIZED","message":"authentication required, visit https://aka.ms/acr/authorization for more
information."}]}
PS C:\Users\Tassneem>
PS C:\Users\Tassneem> docker login badelhadocker.azurecr.io
Username: BadelhaDocker
Password:

Login Succeeded
PS C:\Users\Tassneem> docker image list
REPOSITORY                    TAG       IMAGE ID       CREATED             SIZE
badelhadocker.azurecr.io/v1   latest    2a3d3c7b7626   About an hour ago   792MB
newdocker                     latest    2a3d3c7b7626   About an hour ago   792MB
PS C:\Users\Tassneem> docker tag newdocker badelhadocker.azurecr.io/v1
PS C:\Users\Tassneem> docker push badelhadocker.azurecr.io/v1
Using default tag: latest
The push refers to repository [badelhadocker.azurecr.io/v1]
42ff22a94d5e: Pushing  4.923MB/166MB
a19f2aa46c6c: Pushing  4.976MB/133.7MB
ff98a0867a14: Pushed
7712f0cacbd7: Pushed
4e6f99c1668f: Pushing  13.62MB/361.7MB
2eb7c5261c95: Pushing  6.026MB/11.64MB
a9435642a9e3: Pushed
b3232f6d8e2e: Pushing  3.236MB/34.55MB
146826fa3ca0: Waiting
5d4427064ecc: Waiting
```

**Deployment to Azure App Service:** We deploy the Docker image to Azure

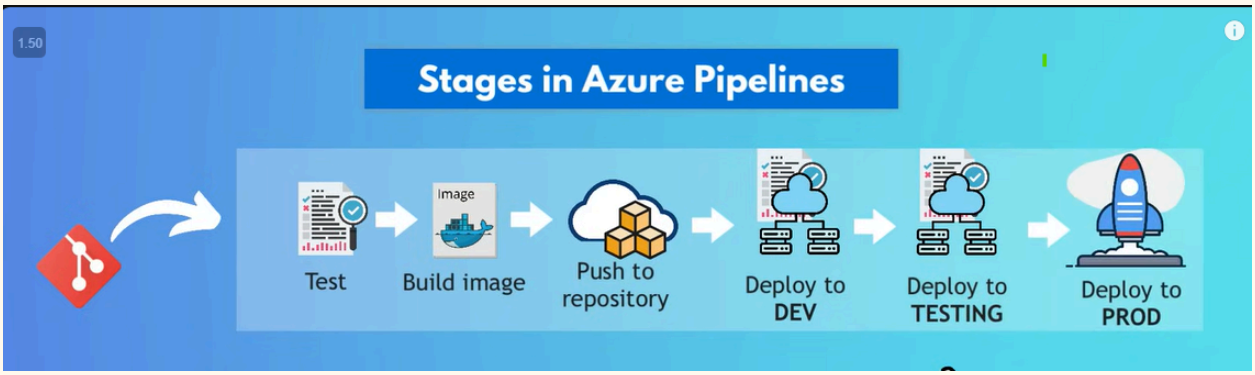


**Integration with CI/CD Workflow:** Docker deployment is integrated into our CI/CD workflow, to ensure automated deployment processes. With each code commit, our CI/CD pipeline triggers a Docker image build and pushes it to the container registry.

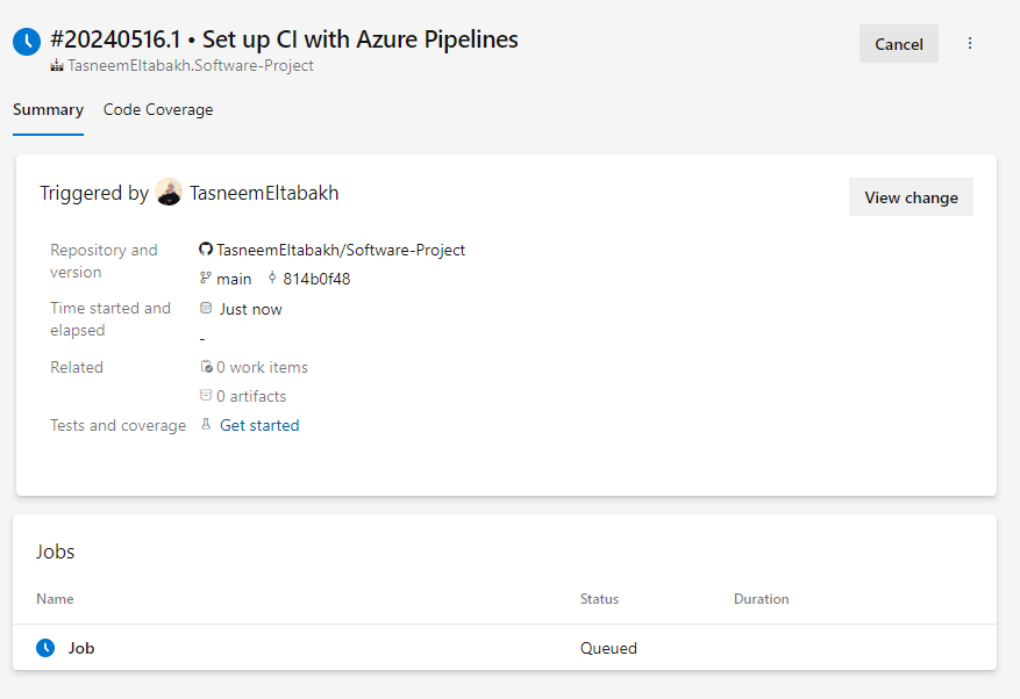


# Workflow for CI/CD

**Overview:**The workflow is triggered by commits to the main branch and is configured to build the application, push the Docker image to Azure Container Registry, and deploy it to an Azure Web App.




At the first, we tried to work out using Azure pipelines but we didn't want to work on a self hosted pool so we get to Github actions



**Prerequisites :** before we dive in to github action we must do the following

- GitHub repository for the application code.
- Azure Container Registry (ACR) setup.
- Azure Web App created.
- GitHub secrets configured for Azure credentials

## Workflow Description

- **Triggering Events**: The workflow is triggered on push events to the main branch
- **Build Job:**
    1. Runs on an Ubuntu environment.Check out the code from the repository.
    2. Sets up Docker Builds for multi-platform builds.
    3. Logs into the Azure Container Registry.
    4. Converts the GitHub username to lowercase to ensure compatibility with Docker tagging conventions.
    5. Builds the Docker image and pushes it to the Azure Container Registry.
- **Deploy Job**
    1. Runs on an Ubuntu environment.
    2. Depends on the successful completion of the build job.
    3. Logs into the Azure Web App using the publish profile.
    4. Deploys the Docker image from the Azure Container Registry to the Azure Web App.

# Testing

Functional Requirements Testing:

Requirements:

1) Login Authentication
2) Applying Promo Codes

1) Login Authentication

**Description:**

This test case verifies that a user is able to successfully login to the website using valid credentials.

**Preconditions:**

- Chrome WebDriver is installed and accessible.
- The target website ("https://badelha.azurewebsites.net/Login") is functional.
- Valid login credentials

**Input Data:**

- Email: nmnm@gmail.com.
- Password: nmnm.
- Click on the input button

**Test steps:**

- Open the target website ("https://badelha.azurewebsites.net/Login") in Chrome browser.
- Maximize the browser window.
- Click on the email input field.

- Enter the email address "nmnm@gmail.com" in the email field.
- Click on the password input field.
- Enter the password "nmnm" in the password field.
- Click on the login button.

**Expected result:**

The user should be successfully logged in to the website.

**Pass/Fail Criteria:**

- The test case will PASS if the user can log in successfully.
- The test case will FAIL if any errors occur during the login process (e.g., invalid data ”mail, pass”, login button not clickable).

Test Script for Login Authentication:

```
class TestDefaultSuite():

  def setup_method(self, method):

    self.driver = webdriver.Chrome()

    self.vars = {}


  def teardown_method(self, method):

    self.driver.quit()


  def test_testpro1(self):

    self.driver.get("https://badelha.azurewebsites.net/Login")
```

```
    self.driver.set_window_size(820, 663)

    self.driver.find_element(By.NAME, "email").click()

    self.driver.find_element(By.NAME, "email").send_keys("nmnm@gmail.com")

    self.driver.find_element(By.NAME, "pass").click()

    self.driver.find_element(By.NAME, "pass").send_keys("nmnm")

    self.driver.find_element(By.CSS_SELECTOR,
".login100-form-btn").click()
```

Results of running the tests:

```
============================================================ test session starts ============================================================
platform win32 -- Python 3.12.2, pytest-8.2.0, pluggy-1.5.0
rootdir: E:\Third year semester2\SoftWare\project\Project_From_Git\Software-Project\tests
collected 1 item

test_Login.py
DevTools listening on ws://127.0.0.1:62662/devtools/browser/d8bc5df9-ec12-4442-b600-ee6a1b8b377f
.                                                                                                                                    [100%]

============================================================= 1 passed in 76.52s (0:01:16) =============================================================
```

## 2) Applying Promo Codes

**Description:**

This test case verifies that the total price in the user's cart is discounted by a certain percentage (the percentage of the promo code)

**Preconditions:**

- The website is up and running.
- The user has valid login credentials.
- The user should have items in his/her cart

**Input Data:**

- The user's email address and password to login

Ex: username: 'nmnm@zewailcity.edu.eg'

password:'nmnm'

- The promo code's value ('zewail')

**Test steps:**

- Navigate to the website login page.
- Enter valid username and password in the login form.
- Click on the "Login" button.
- Redirect to the shopping cart page
- Check and store the total price of the cart
- Apply the Promo Code
- Click on the Apply Button
- Check the total price after the discount
- Assert that discounted price < total price

**Expected result:**

The user should be able to go to the cart and apply a promo code to get a discounted price.

**Pass/Fail Criteria:**

The test cases pass if the discounted price is less than the original total price and fails otherwise.

Test Script for Promo Code Test Case:

```
class TestCartPRomocode():
  def setup_method(self, method):
    self.driver = webdriver.Chrome()
    self.vars = {}
```

```
  def teardown_method(self, method):
    self.driver.quit()


  def test_cartPRomocode(self):
    self.driver.get("https://badelha.azurewebsites.net/Login")
    self.driver.set_window_size(820, 663)
    self.driver.find_element(By.NAME, "email").click()
    self.driver.find_element(By.NAME, "email").send_keys("")
    self.driver.find_element(By.NAME, "pass").click()
    self.driver.find_element(By.NAME, "pass").send_keys("2263")
    self.driver.find_element(By.CSS_SELECTOR,
".login100-form-btn").click()
    self.driver.get("https://badelha.azurewebsites.net/shop-cart")
    self.driver.set_window_size(787, 864)
    total_Pice= self.driver.find_element(By.NAME, "total_price").text
    self.vars["total_price"] = self.driver.find_element(By.NAME,
"total_price").text
    print("{}".format(self.vars["total_price"]))
    self.driver.find_element(By.NAME, "promo_code").click()
    self.driver.find_element(By.NAME, "promo_code").send_keys("zewail")
    WebDriverWait(self.driver,
30).until(EC.presence_of_element_located((By.NAME, "total_price")))
```

```
    discounted_price= self.driver.find_element(By.NAME,
"total_price").text

    self.vars["discounted_price"] = self.driver.find_element(By.NAME,
"total_price").text

    print("{}".format(self.vars["discounted_price"]))

    assert discounted_price<total_Pice
```

Results of running the tests:

```
platform win32 -- Python 3.12.2, pytest-8.2.0, pluggy-1.5.0
rootdir: C:\Users\Dell\Documents\GitHub\Software-Project
collected 2 items

tests\test_Login.py [13864:27904:0518/180755.699:ERROR:sandbox_win.cc(910)] Sandbox cannot access executable. Check filesystem permissions are valid. S
ee https://bit.ly/31yqMJR.: Access is denied. (0x5)

DevTools listening on ws://127.0.0.1:50327/devtools/browser/f65e089e-367d-4416-9b64-9b8ffcd960b3
[13864:28596:0518/180755.901:ERROR:network_service_instance_impl.cc(600)] Network service crashed, restarting service.
.                                                                                                                    [ 50%]
tests\test_shop-cart.py [25124:12524:0518/180806.395:ERROR:sandbox_win.cc(910)] Sandbox cannot access executable. Check filesystem permissions are vali
d. See https://bit.ly/31yqMJR.: Access is denied. (0x5)

DevTools listening on ws://127.0.0.1:50365/devtools/browser/303fc93e-1b82-4a69-b463-ee496a332e0a
[25124:22532:0518/180806.655:ERROR:network_service_instance_impl.cc(600)] Network service crashed, restarting service.
.                                                                                                                    [100%]

============================================================ 2 passed in 21.79s ============================================================
```