A Project Report on

"GAME STORE WEBSITE"

Submitted to the Bangalore North University in partial fulfillment of

The requirement for the award of the degree in

Bachelor of Computer Applications

Submitted by

RAHUL.M.SALAGUNDI USN: R1919281

Under the guidance of

Ms. DEEPA PURANIK
BCA Department.

NEW HORIZON COLLEGE
New Horizon Knowledge Park, Ring Road, Marathalli
Affiliated to Bangalore North University, Recognized by Govt. of Karnataka
Recognized under section 2(f) of the UGC Act, 1956

STUDENTS DECLARATION

I, **RAHUL.M.SALAGUNDI** hereby declare that the project report entitled **"GAME STORE WEBSITE"** was prepared by me during the year 2021-2022 and was submitted to the Bangalore North University, in partial fulfillment of the requirements for the award of the Degree in **Bachelor of Computer Applications**. I also declare that this project report is original and genuine and it has not been submitted to any other University for the award of any degree, diploma or other similar titles or purposes.

Date: **Signature of the Candidate**

Place: **RAHUL.M.SALAGUNDI**

[USN: R1919281]

NEW HORIZON
COLLEGE
New Horizon Knowledge Park, Ring Road, Marathalli
Affiliated to Bangalore North University, Recognized by Govt. of Karnataka
Recognized under section 2(f) of the UGC Act, 1956

CERTIFICATE

This is to certify that the project titled "**GAME STORE WEBSITE**" has been satisfactorily completed by **RAHUL.M.SALAGUNDI**, **Reg No. R1919281**, in partial fulfillment of the requirements for Project Lab with course code F0250, for the VI Semester BCA course during the academic semester from 2021 - 2022 as prescribed by North Bangalore University.

Faculty In-charge **Head of the Department**

Valued by

Examiner 1: ________________________ Date:

Examiner 2: ________________________ Centre: New Horizon College

ACKNOWLEDGEMENT

First of all, we would like to thank the God Almighty for all the blessings he has showered on us. Our spiritual quotient gave us more strength and motivation that helped immensely

We express our deep sincere thanks to **Dr.Mohan Maghnani, Chairman of New Horizon Educational Institutions** for providing the platform and infrastructure to do this project. This project would not have been possible without the help and co-operation of many. At the outset I wish to record my profound gratitude to our **Principal Dr. R. Bodhisatvan**.

We are extremely thankful **to Mr Nagaraju K, Head of the Department, BCA** for his unending support and encouragement during the development of this project.

We would like to acknowledge the interest, support and the guidance extended by our project guide **Ms. Deepa Puranik, Department of BCA**, to make this project implementation successful.

We thank all other faculty members who helped us a lot in completing this project. The computer lab was always open for us. We thank the Lab administrator and other technical staff for their help and support.

Above all we would thank fellow friends, faculties and parents as well for their valuable suggestions and support in developing the project report.

Finally, we extend our deep sense of gratitude to all those who made this project come alive and encouraged and guided me from the start to finish.

SYNOPSIS

Definition:

This Game Store website is an e-commerce through which we can purchase games for various platforms without any hassle.

Introduction to the system:

The system is a real-time system in the form of web app. It will provide a platform to buy game keys, watch game trailers and check accurate prices of the game.

Need of the software:

The aim of this project is to present a user friendly store to buy online games for various platforms such as Playstation, X Box, PC etc. It maintains a record of all games available to buy and also tags all the bestsellers. The main objective of this project is to remove constrains such as delivery and availability of games, which can be made possible through an e-commerce website.

List of modules:

1. Sign-Up
2. Login
3. Home
4. Cart
5. Product Details
6. Order Details
7. Pending Orders

TABLE OF CONTENTS

1. INTRODUCTION

1.1 PROJECT PROFILE
1.2 PROJECT INTRODUCTION

1. INTRODUCTION

1.1 PROJECT PROFILE

Project Title: Game Store Website

Platform: Microsoft Windows 11

Back-end Used: Django Framework [Python]

Front-end Used: Python, CSS, HTML

Team Size: 1

Project Objective: Objective of this project is to make purchasing of game keys easier through online mode.

Project Duration: 3 months.

Guided BY: Ms. Deepa puranik

Submitted To: New Horizon College, Bangalore.

1.2 PROJECT INTRODUCTION

Over the years, games developed from black and white to coloured, 2D to 3D, from the Television Screens to small, portable little screens. This is because of the never-ending demands of children who get hooked up to mind-challenging and adventurous games. The video game industry is still relatively young compared to others, but in such a short period of time, hardware improvements have made leaps and bounds making video games one of the biggest businesses around. Not only do these improvements make Online Gaming more realistic and fun, but it has impacted sales in many ways.

Therefore, the main objective of this project is to meet the demands of the customer for various games and providing timely delivery of it. It presents a simple user interface for Signing-Up and buying new games for a variety of platforms such as Playstations, Xbox, Nintendo and PC.

It removes constrains such as delivery and availability of games, which can be made possible through an e-commerce website.

2. ENVIRONMENT DESCRIPTION

2.1 HARDWARE AND SOFTWARE REQUIREMENT

2.2 TECHNOLOGY USED

2. ENVIRONMENT DESCRIPTION

2.1 HARDWARE AND SOFTWARE REQUIREMENTS

DEVELOPMENT Side Tools

Processor	Intel core i5
RAM	4 GB minimum
Hard Disk	15 GB of available space required on system drive of available or more.
Operating System	Windows 7, 8, 8.1, 10, 11
Access	CSS, HTML, Visual Studio Code and Django Framework.

Server-Side Tools

Processor	PC with intel i5 or above.
RAM	4 GB
Hard Disk	80GB
Operating System	Windows 7, 8, 8.1, 10, 11
Software	Django [Python]

2.2 TECHNOLOGY USED

HTML

Hypertext Markup Language (**HTML**) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript, it forms a triad of cornerstone technologies for the World Wide Web. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML can embed programs written in a scripting language such as JavaScript , which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), maintainer of both the HTML and the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

CSS

CSS Stands for "Cascading Style Sheet." Cascading style sheets are used to format the layout of Web pages. They can be used to define text styles, table sizes, and other aspects of Web pages that previously could only be defined in a page's HTML.

CSS helps Web developers create a uniform look across several pages of a Web site. Instead of defining the style of each table and each block of text within a page's HTML, commonly used styles need to be defined only once in a CSS document. Once the style is defined in cascading style sheet, it can be used by any page that references the CSS file. Additionally, CSS makes it easy to change styles across several pages at once. For example, a Web developer may want to increase the default text size from 10pt to 12pt for fifty pages of a Web site. If the pages all reference the same style sheet, the text size only needs to be changed on the style sheet and all the pages will show the larger text.

PYTHON

Python is a very popular general-purpose interpreted, interactive, object-oriented, and high-level programming language. Python is dynamically-typed and garbage-collected programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL).

Python supports multiple programming paradigms, including Procedural, Object Oriented and Functional programming language. Python design philosophy emphasizes code readability with the use of significant indentation.

This tutorial gives a complete understanding of Python programming language starting from basic concepts to advanced concepts. This tutorial will take you through simple and practical approaches while learning Python Programming language.

Python is consistently rated as one of the world's most popular programming languages. Python is fairly easy to learn, so if you are starting to learn any programming language then Python could be your great choice. Today various Schools, Colleges and Universities are teaching Python as their primary programming language. There are many other good reasons which makes Python as the top choice of any programmer:

- Python is Open Source which means its available free of cost.

- Python is simple and so easy to learn
- Python is versatile and can be used to create many different things.
- Python has powerful development libraries include AI, ML etc.
- Python is much in demand and ensures high salary

Python is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain. I will list down some of the key advantages of learning Python:

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Characteristics of Python

Following are important characteristics of **Python Programming** −

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Applications of Python

The latest release of Python is 3.x. As mentioned before, Python is one of the most widely used language over the web. I'm going to list few of them here:

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

DJANGO

Django is an MVT web framework that is used to build web applications. The huge Django web-framework comes with so many “batteries included” that developers often get amazed as to how everything manages to work together. The principle behind adding so many batteries is to have common web functionalities in the framework itself instead of adding latter as a separate library.

One of the main reasons behind the popularity of Django framework is the huge Django community. The community is so huge that a separate website was devoted to it where developers from all corners developed third-party packages including authentication, authorization, full-fledged Django powered CMS systems, e-commerce add-ons and so on. There is a high probability that what you are trying to develop is already developed by somebody and you just need to pull that into your project.

Why should you use Django?

Django is designed in such a way that encourages developers to develop websites fast, clean and with practical design. Django’s practical approach to getting things done is where it stands out from the crowd.

If you’re planning to build a highly customizable app, such as social media website, Django is one of the best frameworks to consider. Django strength lies in its interaction between users or its ability to share different types of media. One of the great advantage of django is its ability to utilize large community-based support which gives you highly customizable third-party ready to use plugins in your applications.

Below are the top ten reasons to choose Django for web development −

Python

Python is arguably one of the easiest programming languages to learn because of its simple language constructs, flow structure and easy syntax. It is versatile and runs websites, desktop applications and mobile applications embedded in many devices and is used in other applications as a popular scripting language.

Batteries Included

Django comes with common libraries which are essential to build common functionalities like URL routing, authentication, an object-relational mapper (ORM), a templating system and db-schema migrations.

Built-in admin

Django has an in-built administration interface which lets you handle your models, user/ group permissions and to manage users. With model interface in place, there is no need for a separate database administration program for all but advanced database functions.

Doesn't get in your way

Creating a Django application adds no boilerplate and no unnecessary functions. There's no mandatory imports, third-party libraries and no XML configuration files.

Scalable

Django is based on MVC design pattern. It means that all the entities like db (database), back-end and front-end code are individual entity. Django allows us to separate code from the static media including pictures, files, CSS and JavaScript that make up your site.

Django supports a full list of third-party libraries for web servers, caching, performance management, clustering and balancing. One of the advantages Django provides is the support for major email and messaging applications and services like ReST and OAuth.

Battle tested

Django was first open-sourced in 2005. After 12 years of growth, Django now not only runs news publishing websites but also runs all or part of major global enterprise like Pinterest, Instagram, Disqus, Bitbucket, EventBrite and Zapier. This makes it a robust and reliable web framework to work with.

Huge package support

Because of its large community support and huge developers network, there is a high possibility that whatever you intend to do might have been done before. Large international community of developers contribute to the community by releasing their projects as open-source packages.

One such repository of these projects is Django Package site. Currently, Django packages list over 3400 plus reusable Django apps, sites and tools to use in our Django projects.

Actively developed

One of the biggest risks associated with open source project is its sustainability. We cannot be sure if it lasts long.

There is no such risk with Django as it is 12 years old. Its consistent releases, newer/better versions and active community is growing every-day with a large core team of voluntary contributors who maintains and improve the code base every-day.

Stable releases

Open-source software projects like Django are, in many cases, actively developed and more secure than competing proprietary software as many developers are developing and testing it every day. However, the drawback of an open-source software project is the absence of a stable codebase to commercially viable development.

3. SYSTEM STUDY

3.1 EXISTING SYSTEM

3.2 SCOPE OF THE SYSTEM

3.2.1 PROPOSED SYSTEM

3. SYSTEM STUDY

3.1 EXISTING SYSTEM

The existing system used a walk-in store for the management and maintenance of various games. The current system requires numerous paper forms to store the history of purchases with data spread throughout. Often information is incomplete or does not follow management standards. It requires man power to record the purchases and consumes more time. It is difficult to go back and look for a particular purchase.

3.2 SCOPE OF SYSTEM

Module Description

➢ **User Modules**

- **Sign-Up :** New users can register their accounts with their details in this page.
- **Login :** Existing customers and newly registered customers can login using this page.
- **Home :** This page contains a list of games that can be purchased by any customer. It also has a carousel of most popular games.
- **Cart :** This page contains all the games added to the cart by the customer/user.
- **Product Details :** This page shows details of the game product such as Name, Game Developer, Trailer and Game Reviews.
- **Order Details :** It shows the delivery updates for a games purchased by the registered customer.
- **Pending Orders :** This page contains information about the pending orders of a customer such as Order no., Name, Price and Delivery Status.

- **Admin Modules**

  - **Admin View :** This page contains details such as User Name, Price and Status of all the completed orders.
  - **Admin Dashboard :** it contains details such as Total Earnings, Total pending and delivered items.
  - **Add Item :** To add new games to the homepage.
  - **Item Confirm Delete :** This is a confirmation page which asks for confirmation to delete a game from homepage.

3.3 PROPOSED SYSTEM

The aim of proposed system is to develop a system of improved facilities. The proposed system can overcome all the limitations of the existing system. The system provides proper security and reduces the manual work.

OBJECTIVES OF SYSTEM

- Security of data.
- Ensure data accuracies.
- Minimize manual data entry.
- Minimum time needed for the various processing.
- Greater efficiency.
- Better service.
- User friendliness and interactive.
- Minimum time required.

4. SYSTEM PLANNING

4.1 REQUIREMENT SPECIFICATION

4.2 FEASIBILITY STUDY

4.3 LIFE CYCLE MODEL

4. SYSTEM PLANNING

4.1 REQUIREMENT SPECIFICATION

Before building an elegant application, best way to identify basic needs and functionality is to visit systems relevant to that specific area addressed by proposed application.

For gathering basic needs and functionalities, we had meeting with organization personal to understand requires functionalities also noted down some useful feature to be included. We also referred to some articles related to application to understand various functionalities.

- Requirement gathering for project “Event Management” initiated by the study of a past few projects.
- We also took guidance from our college faculties to clear about requirement analysis.
  We held several meetings with our project guide to decide the process model and basic system architecture.
- The current system satisfies all users requirements.

4.2 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are:

Economic Feasibility

This study is carried out to check the economic impact will have on the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products have to be purchased.

Technical Feasibility

This study is carried out to check the technical feasibility, that is,the technical requirements of the system. Any system developed must not have a high demand on the available available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes for the implementing this system.

Operational Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system

4.3 LIFE CYCLE MODEL

Types of Life Cycle Model:

- Prototype Model
- Waterfall Model
- Iterative Enhancement Model
- Spiral Model
- Dynamic System Development Model

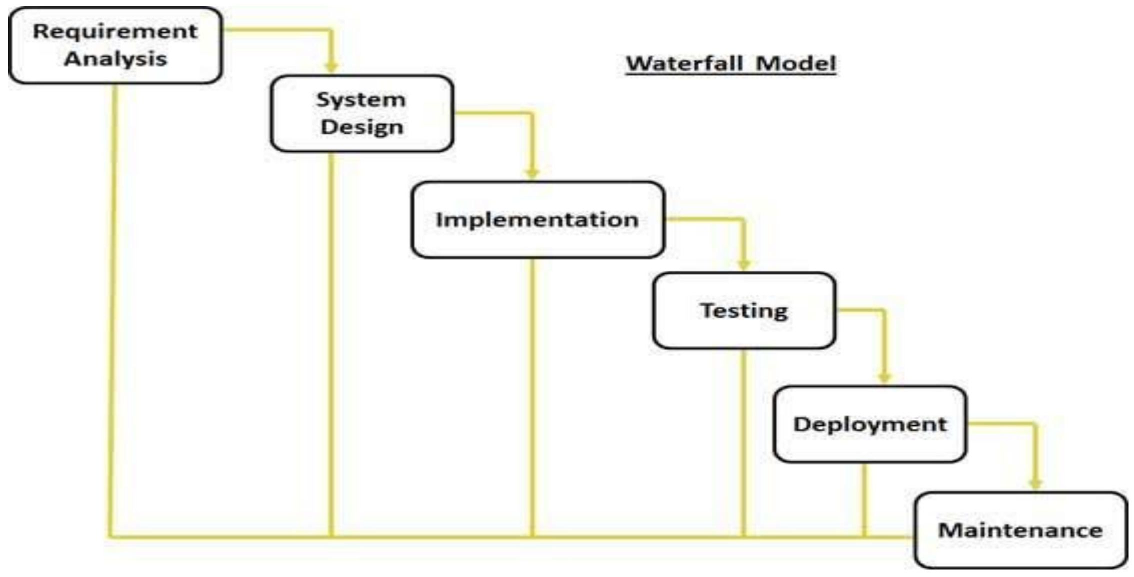
PROTOTYPE MODEL

The Prototyping Model is a systems development method (SDM) in which a prototype (an early approximation of a final system or product) is built, tested, and then reworked as necessary until an acceptable prototype is finally achieved from which the complete system or product can now be developed. This model works best in scenarios where not all of the project requirements are known in detail ahead of time. It is an iterative, trial-and-error process that takes place between the developers and the users.

WATERFALL MODEL

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

The advantage of waterfall development is that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.



ITERATIVE ENHANCEMENT MODEL

Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. At each iteration, design modifications are made and new functional capabilities are added. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).

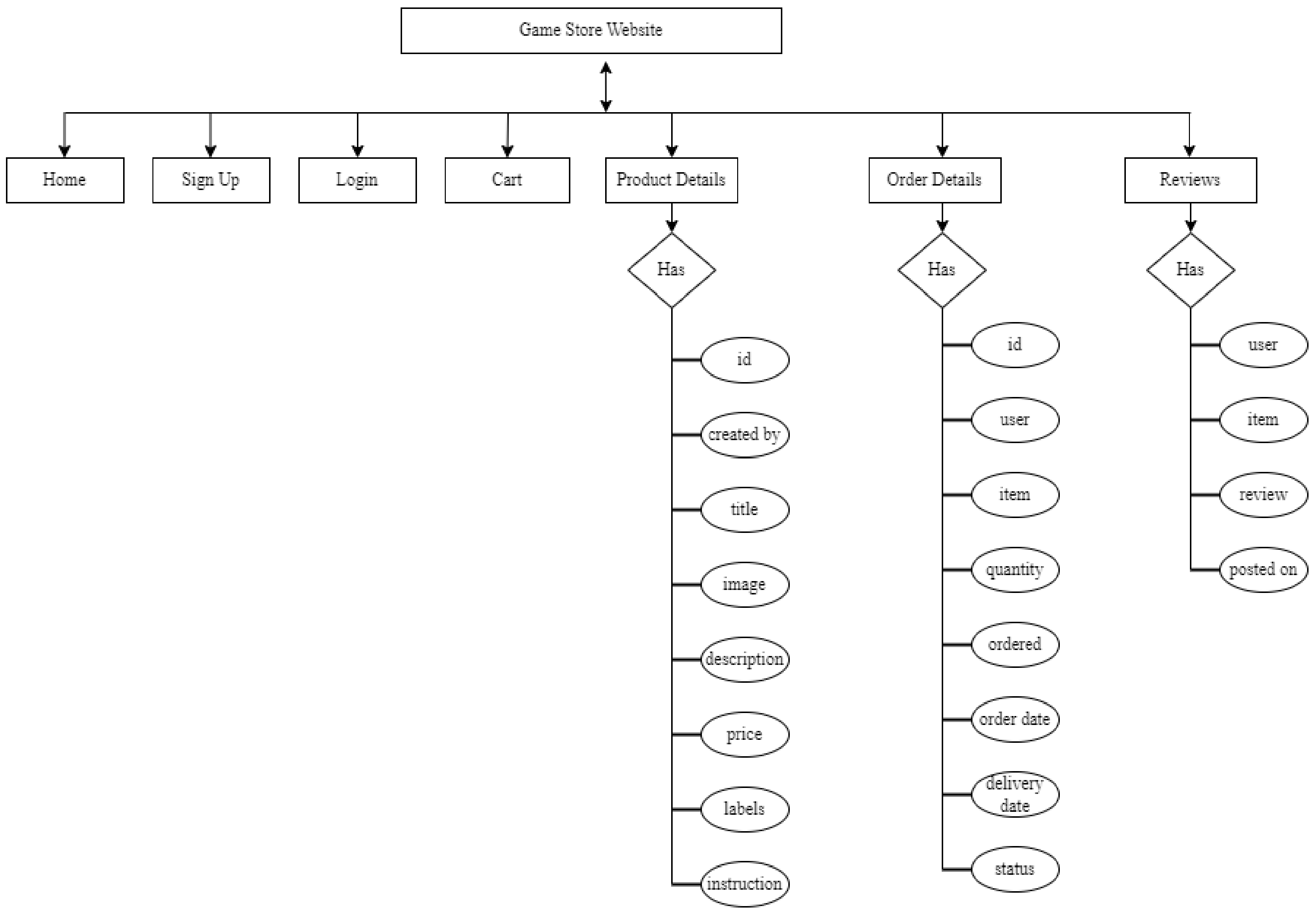
The advantage of this model is that there is a working model of the system at a very early stage of development which makes it easier to find functional or design flaws. Finding issues at an early stage of development enables to take corrective measures in a limited budget.

5. SYSTEM MODEL ARCHITECTURE

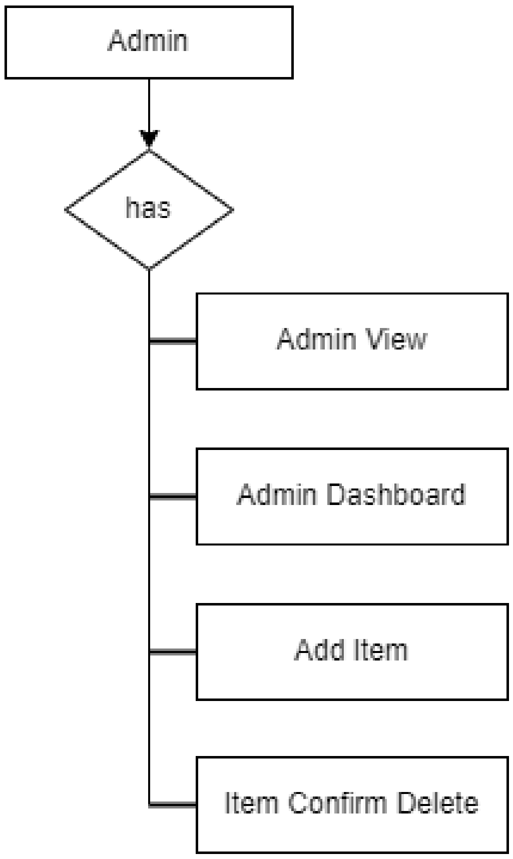
5. SYSTEM ARCHITECTURE

5.1 E-R DIAGRAM

USER:

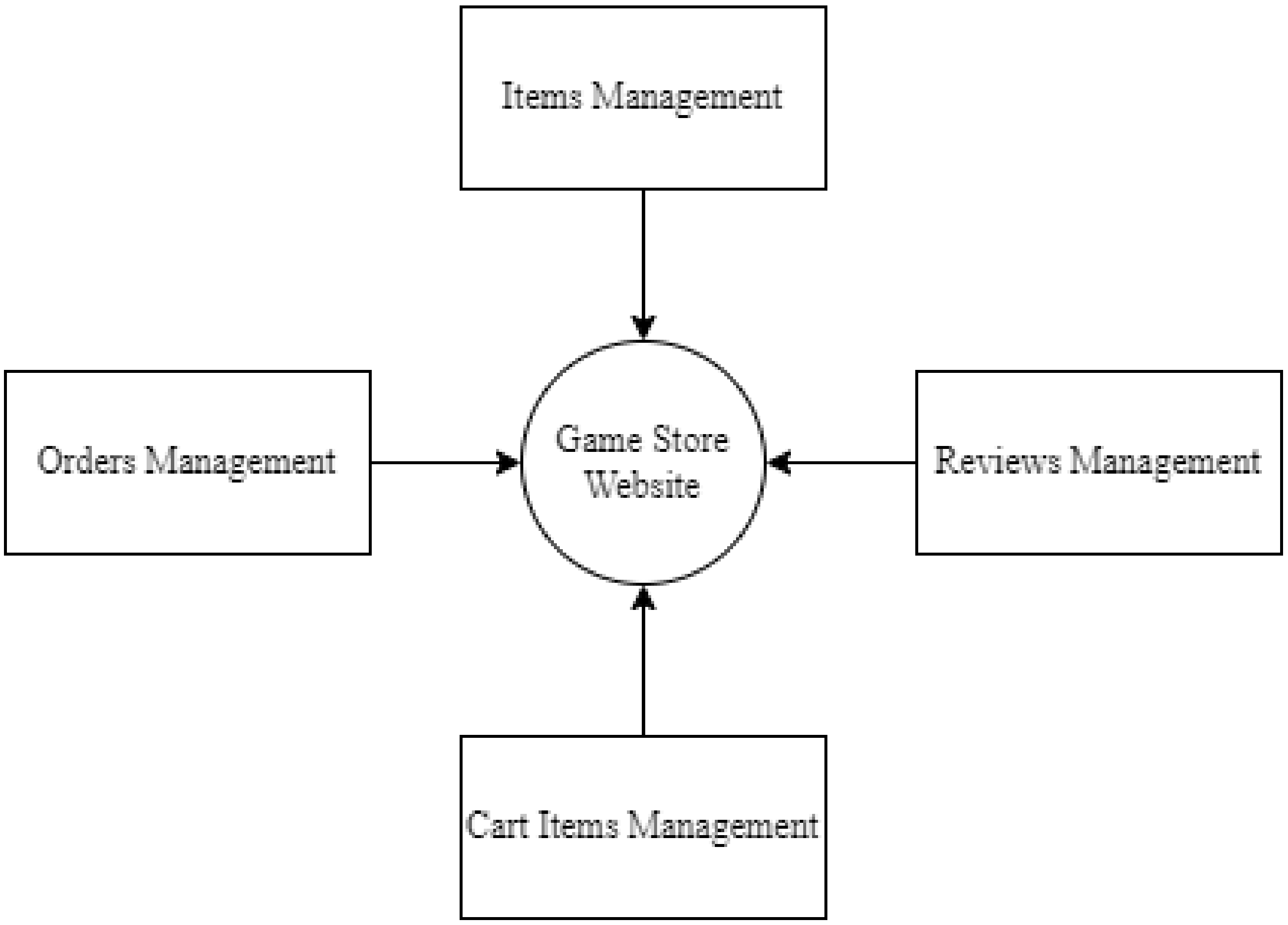


ADMIN:

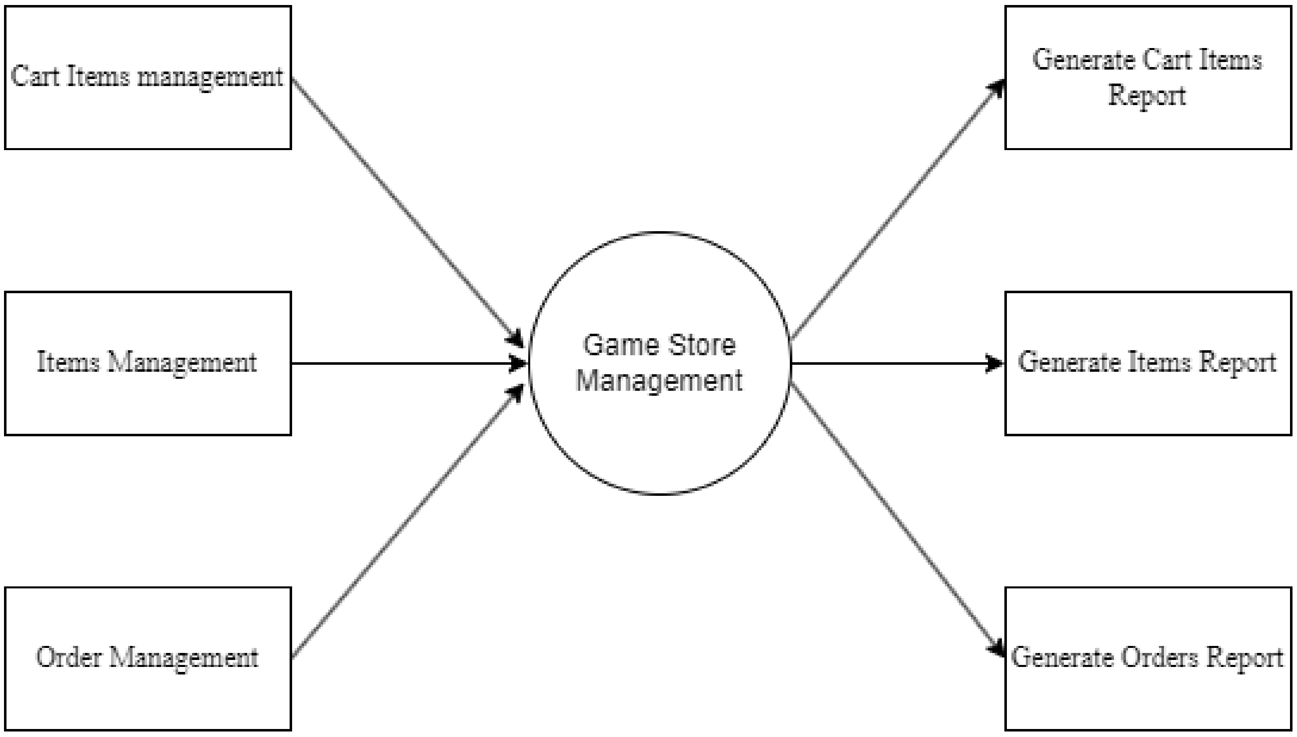

Admin
has
Admin View
Admin Dashboard
Add Item
Item Confirm Delete

5.2 DATAFLOW DIAGRAM

LEVEL 0:



LEVEL 1:


Cart Items management
Items Management
Order Management
Game Store Management
Generate Cart Items Report
Generate Items Report
Generate Orders Report

5.3 TABLES

Product Details

id	number
created by	text
title	text
image	text(Specialized)
description	text
price	number
labels	text
instructions	text

Order Details

id	number
user	text
item	text
quantity	number
ordered	text
Order date	date/time
Delivery date	date/time
status	text

Review

user	text
item	text
review	text
posted on	date/time

6. SOFTWARE TESTING

6.1 TESTING INTRODUCTION

6.2 TEST CASE

6. SOFWARE TESTING

6.1 TESTING INTRODUCTION

Software testing is an investigation conducted to provide stakeholders with information about the quality of the software product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding software bugs (errors or other defects), and verifying that the software product is fit for use. Software testing involves the execution of a software component or system component to evaluate one or more properties of interest.

The general testing process is the creation of a testing strategy (which sometimes includes the execution of the testing case), creation of a test plan/design (which usually includes test case and test procedure) and the execution of tests. Test are inputs that have been devised to test the system.

Types of Testing

1. Black-Box Testing

2. White Box Testing

3. Alpha Testing

4. Beta Testing

5. Clean Room Testing

Black-Box Testing

Black box Testing is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional. It focuses on the functional requirements of the software. In black-box testing, the cases are derived from the specification of the system, the implementation details of the system are not taken into considerations.

White Box Testing

White-box testing, sometimes called Glass box testing or structural testing or clear box testing. White box testing is used to test areas that cannot be reached from a black box testing. White-box texting user an internal perspective of the system to design the test cases based on the internal structure.

Alpha Testing

Alpha testing is simulated for actual operational testing by potential users/customers or an independent test team at the developer's site. Alpha testing is often employed for off-the-self software as a form of internal acceptance testing before the

software goes to beta testing.

Beta Testing

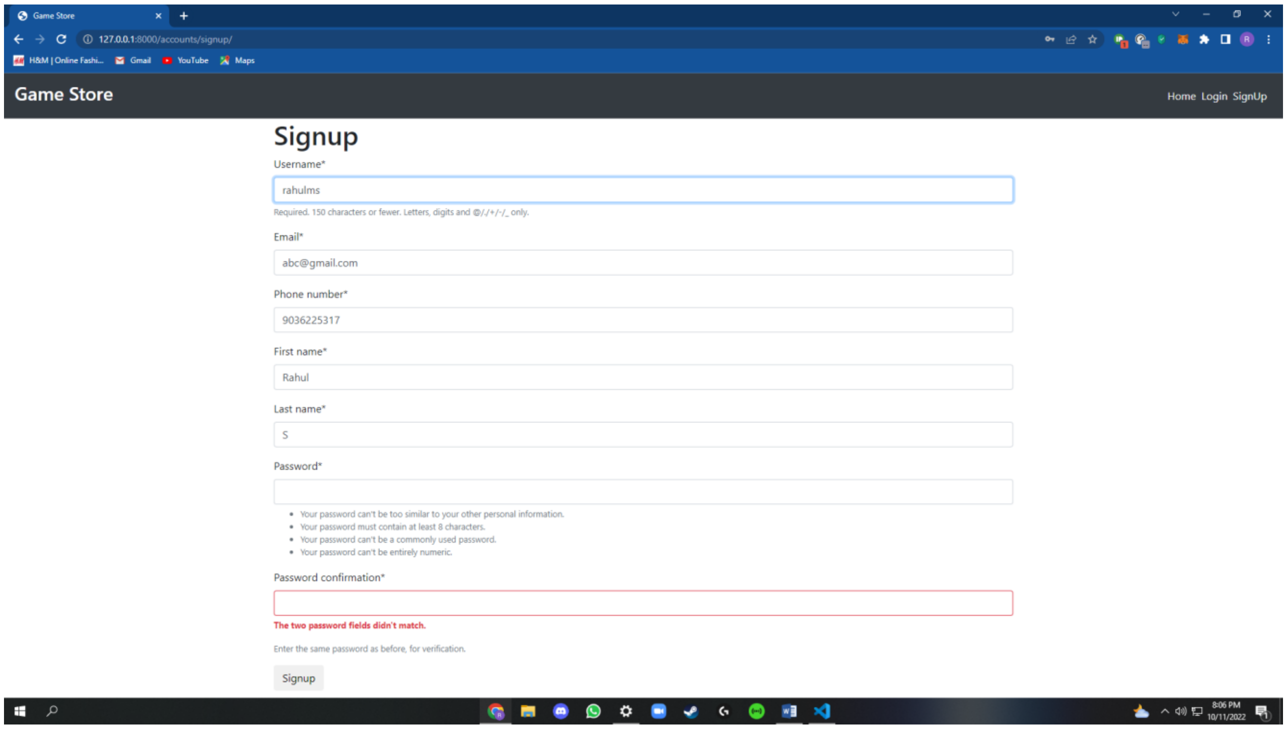
Beta testing comes after alpha testing. Version of the software, known as beta versions, are related to a limited audience outside of the programming team. The software is released to groups of people so that further testing can ensure the products has few faults or bugs.

Clean Room Testing

Clean room testing makes use of incremental development model. The initial increment delivered to the customer with the critical functionalities and less important features are added in later increments.

6.2 TEST CASES

- If the two field i.e , password and password confirmation do not match ,then an error message will be displayed.



Username*

rahulms

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email*

abc@gmail.com

Phone number*

9036225317

First name*

Rahul

Last name*

S

Password*

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

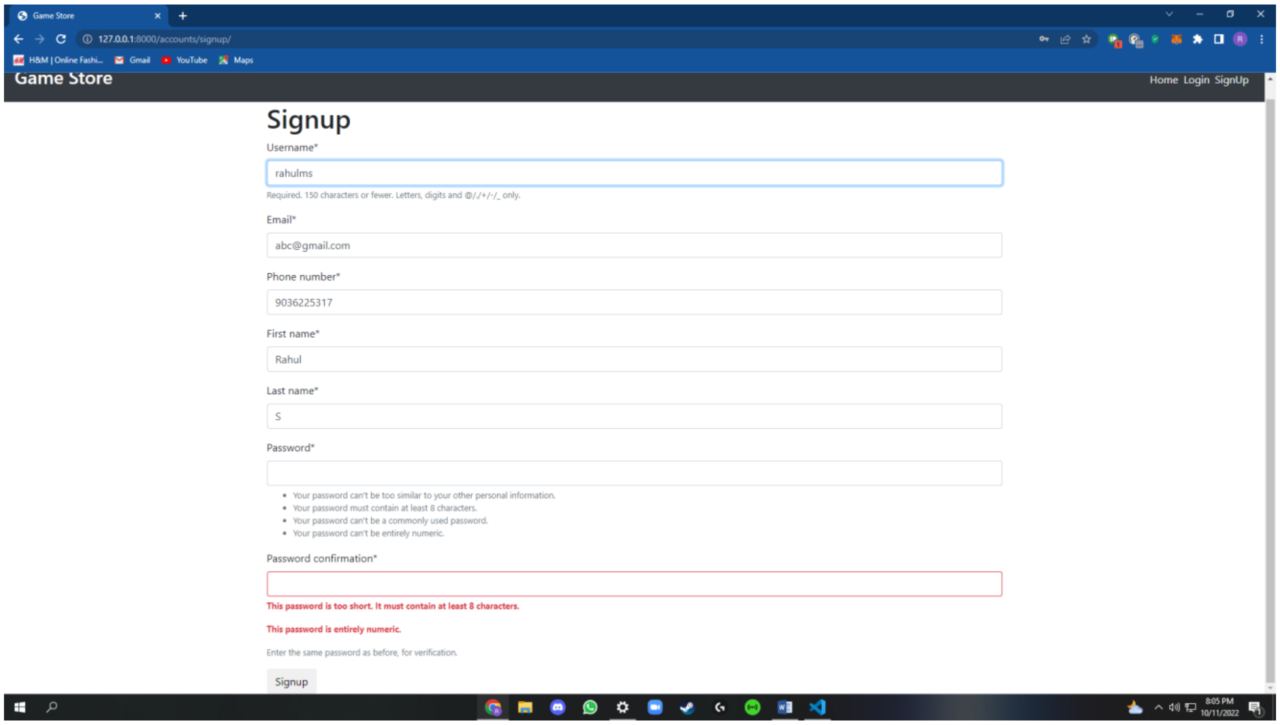
Password confirmation*

The two password fields didn't match.

Enter the same password as before, for verification.

- The password field should meet the following criteria:
    - The password can't be too similar to your other personal information.
    - The password must contain at least 8 characters.
    - The password can't be a commonly used password.
    - The password can't be entirely numeric.

If the password does not match this criteria , then an error message is displayed.



Game Store

Home Login SignUp

Signup

Username*

rahulms

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email*

abc@gmail.com

Phone number*

9036225317

First name*

Rahul

Last name*

S

Password*

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation*

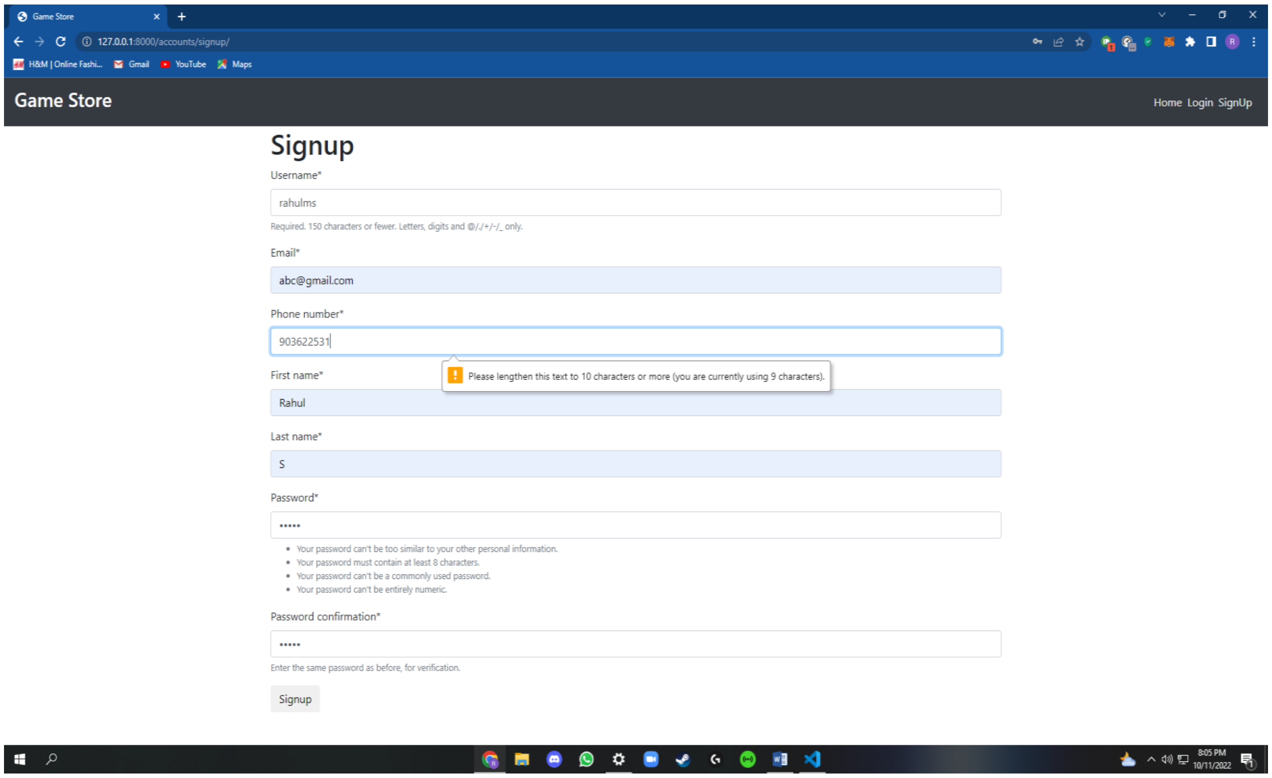
This password is too short. It must contain at least 8 characters.

This password is entirely numeric.

Enter the same password as before, for verification.

Signup

- The mobile number field must contain 10 digits or else an error message is displayed.



Game Store

Home Login SignUp

Signup

Username*

rahulms

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email*

abc@gmail.com

Phone number*

903622531

Please lengthen this text to 10 characters or more (you are currently using 9 characters).

First name*

Rahul

Last name*

S

Password*

•••••

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

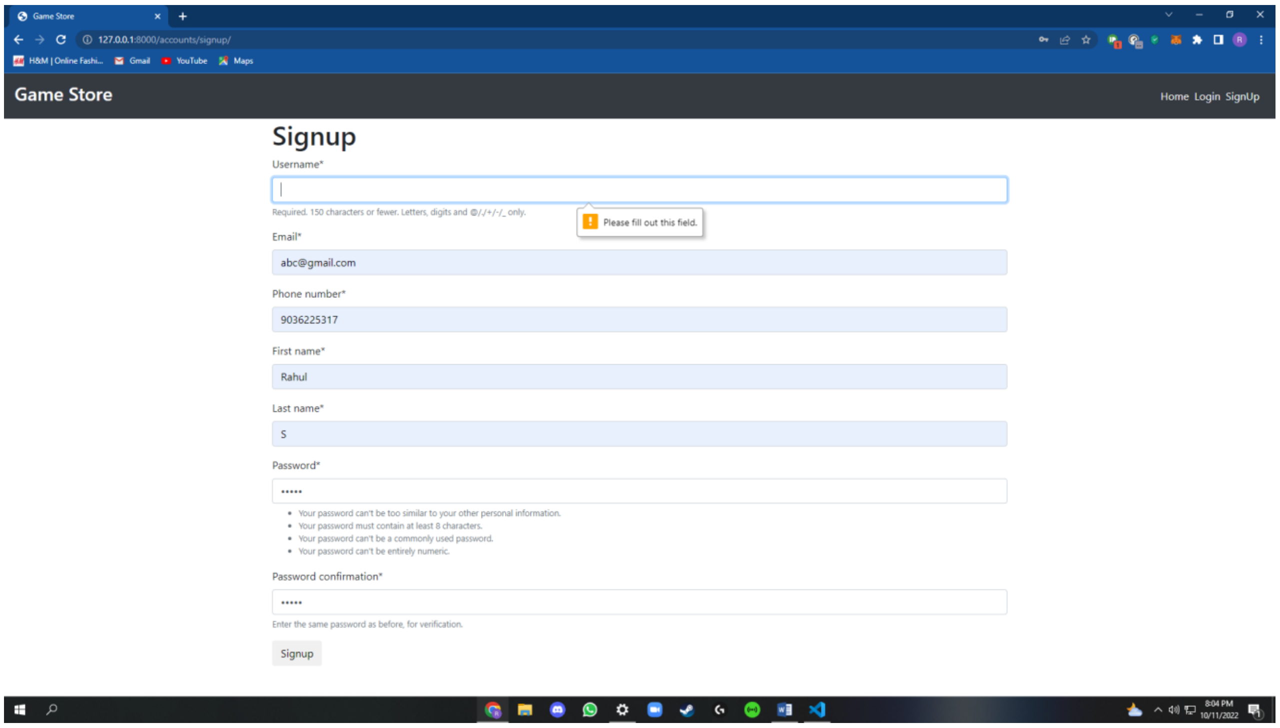
Password confirmation*

•••••

Enter the same password as before, for verification.

Signup

- Fields marked with asterisk (*) cannot be empty.



Signup

Username*

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Please fill out this field.

Email*

abc@gmail.com

Phone number*

9036225317

First name*

Rahul

Last name*

S

Password*

•••••

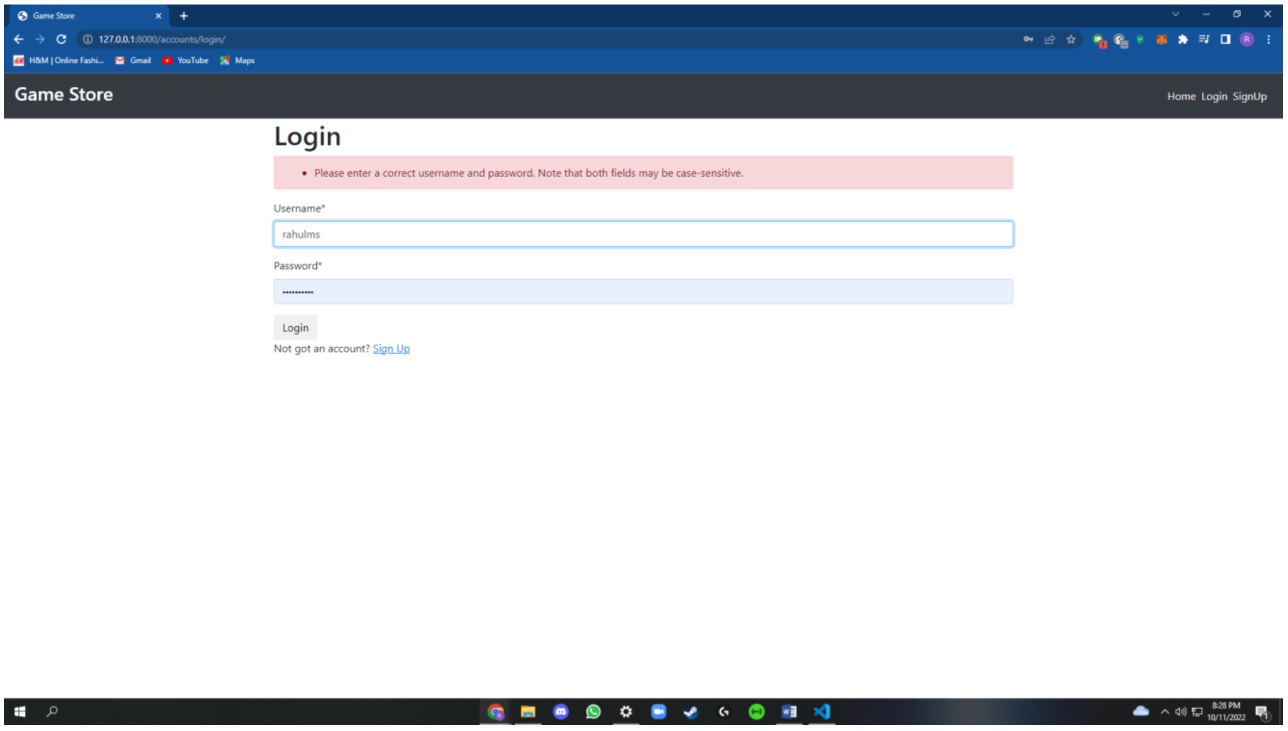
- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation*

•••••

Enter the same password as before, for verification.

- Wrong username or password entry displays a message.



Login

- Please enter a correct username and password. Note that both fields may be case-sensitive.

Username*

rahulms

Password*

••••••••••

Not got an account? Sign Up

7. CODING

7. CODING

accounts/forms.py

```
from django import forms

from django.contrib.auth.forms import UserCreationForm

from django.contrib.auth.models import User


class NewUSerForm(UserCreationForm):

    email = forms.EmailField(required=True)

    phone_number = forms.CharField(required=True,min_length=10)

    first_name = forms.CharField(required=True)

    last_name = forms.CharField(required=True)


    class Meta:

        model = User

        fields = ("username", "email", "phone_number", "first_name", "last_name", "password1",
"password2")


    def save(self,commit=True):

        user = super(NewUSerForm, self).save(commit=False)

        user.email = self.cleaned_data["email"]

        user.phone_number = self.cleaned_data["phone_number"]

        user.first_name = self.cleaned_data["first_name"]

        user.last_name = self.cleaned_data["last_name"]

        if commit:

            user.save()

        return user
```

accounts/urls.py

```
from django.urls import path

from . import views


app_name = 'accounts'


urlpatterns = [

    path('signup/', views.signup_view, name="signup"),

    path('login/', views.login_view, name="login"),

    path('logout/', views.logout_view, name="logout"),

]
```

accounts/views.py

```
from django.shortcuts import render, redirect

from django.contrib.auth.forms import AuthenticationForm

from django.contrib.auth import login, logout

from .forms import NewUSerForm


def signup_view(request):

    if request.method == 'POST':

        form = NewUSerForm(request.POST)

        if form.is_valid():

            user = form.save()

            login(request, user)

            return redirect('main:home')

    else:

        form = NewUSerForm()
    return render(request, 'accounts/signup.html', { 'form': form })


def login_view(request):
    if request.method == 'POST':
        form = AuthenticationForm(data=request.POST)
        if form.is_valid():
            user = form.get_user()
            login(request, user)
            if 'next' in request.POST:
                return redirect(request.POST.get('next'))
            return redirect('main:home')
    else:
        form = AuthenticationForm()
    return render(request, 'accounts/login.html', { 'form': form })


def logout_view(request):
    if request.method == 'POST':
            logout(request)
            return redirect('/')
```

game/settings.py

```
import os


# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
```

```
# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'v7)ouo@=dhswvnfs!@o$v07^m#+2l^1+c#n%bel2eqn!5f^fx^'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'main.apps.MainConfig',
    'accounts.apps.AccountsConfig',
    'crispy_forms',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'game.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

```
WSGI_APPLICATION = 'e_food.wsgi.application'



# Database
# https://docs.djangoproject.com/en/3.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
# Password validation
# https://docs.djangoproject.com/en/3.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]



# Internationalization
# https://docs.djangoproject.com/en/3.0/topics/i18n/

LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'Asia/Kolkata'
USE_I18N = True
USE_L10N = True
USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.0/howto/static-files/

STATIC_URL = '/static/'

MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media/')

CRISPY_TEMPLATE_PACK = 'bootstrap4'
```

game/urls.py

```
from django.contrib import admin

from django.urls import path, include

from django.conf import settings

from django.conf.urls.static import static



admin.site.site_header = "E-Food Admin"

admin.site.site_title = "E-Food"

admin.site.index_title = "E-Food Administration Panel"


urlpatterns = [

    path('admin/', admin.site.urls),

    path('',include('main.urls')),

    path('accounts/', include('accounts.urls')),

]


if settings.DEBUG:

    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

game/wsgi.py

```
import os
```

```
from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'e_food.settings')

application = get_wsgi_application()
```

style.css

```
.product .main .container .row img{
    max-width: 540px;
    max-height: 450px;
    width: fit-content;
}
.product .main .container .row .content{
    display: flex;
    flex-direction: column;
    text-align: center;
    justify-content: center;
}
.product .main .container .row .information{
    display: flex;
    flex-direction: column;
    text-align: center;
    justify-content: center;
}
```

admin_dashboard.html

```
{% load static %}

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="utf-8">

  <meta http-equiv="X-UA-Compatible" content="IE=edge">

  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

  <meta name="description" content="">

  <meta name="author" content="">


  <title>Admin Dashboard</title>


  <link
href="https://fonts.googleapis.com/css?family=Nunito:200,200i,300,300i,400,400i,600,600i,700,
700i,800,800i,900,900i" rel="stylesheet">

  <link href="css/sb-admin-2.min.css" rel="stylesheet">

  <link rel="stylesheet" type="text/css" href="{% static 'static/main/sb-admin-2.min.css' %}">

  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">

    <script
src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.3/Chart.bundle.min.js"></script>

</head>
```

```
<body id="page-top">

  <div id="wrapper">

    <ul class="navbar-nav bg-gradient-primary sidebar sidebar-dark accordion"
id="accordionSidebar">

      <a class="sidebar-brand d-flex align-items-center justify-content-center" href="{% url
'main:admin_dashboard' %}">

        <div class="sidebar-brand-text mx-3">Welcome</div>

      </a>

      <hr class="sidebar-divider my-0">

      <li class="nav-item active">

        <a class="nav-link" href="{% url 'main:admin_dashboard' %}">

          <i class="fa fa-bandcamp"></i>

          <span>Dashboard</span></a>

      </li>

      <hr class="sidebar-divider">

      <div class="sidebar-heading">

        Admin Interface

      </div>

      <li class="nav-item">

        <a class="nav-link" href="{% url 'main:item-create' %}">

          <i class="fa fa-plus"></i>

          <span>Add Products</span>

        </a>

      </li>

      <li class="nav-item">
        <a class="nav-link" href="{% url 'main:item_list' %}">
          <i class="fa fa-plus"></i>
          <span>View Items</span>
        </a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="{% url 'main:admin_view' %}">
          <i class="fa fa-check"></i>
          <span>Completed Orders</span>
        </a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="{% url 'main:pending_orders' %}">
          <i class="fa fa-bell"></i>
          <span>Pending Orders</span>
        </a>
      </li>
      <hr class="sidebar-divider">
      <li class="nav-item">
        <a class="nav-link">
          <i class="fas fa-fw fa-chart-area"></i>
          <span>Profile</span></a>
      </li>
      <li class="nav-item">
```

```
        <a class="nav-link" href="{% url 'main:home' %}">
          <i class="fa fa-home"></i>
          <span>Home</span></a>
      </li>
      <li class="nav-item ml-3 mb-3">
        <form class="logout-link" action="{% url 'accounts:logout' %}" method="post">
            {% csrf_token %}
            <button type="submit" class="mr-2">Logout</button>
        </form>
      </li>
      <hr class="sidebar-divider d-none d-md-block">
    </ul>
    <div id="content-wrapper" class="d-flex flex-column">
      <div id="content">
        <nav class="navbar navbar-expand navbar-light bg-white topbar mb-4 static-top shadow">
          <h2 class="text-center">AdminPage</h3>
        </nav>
        <div class="container-fluid">

          <div class="row">
            <div class="col-xl-4 col-md-4 mb-4">
              <div class="card border-left-success shadow h-100 py-2">
                <div class="card-body">
                  <div class="row no-gutters align-items-center">
                    <div class="col mr-2">
                      <div class="text-xs font-weight-bold text-success text-uppercase mb-1">Total
          Earnings</div>
                      <div class="h5 mb-0 font-weight-bold text-gray-800">₹ {{ income }}</div>
                    </div>
                    <div class="col-auto">
                      <i class="fas fa-dollar-sign fa-2x text-gray-300"></i>
                    </div>
                  </div>
                </div>
              </div>
            </div>


            <div class="col-xl-4 col-md-4 mb-4">
              <div class="card border-left-primary shadow h-100 py-2">
                <div class="card-body">
                  <div class="row no-gutters align-items-center">
                    <div class="col mr-2">
                      <div class="text-xs font-weight-bold text-primary text-uppercase mb-1">Total
          Delivered</div>
                      <div class="h5 mb-0 font-weight-bold text-gray-800">{{ completed_total
          }}</div>
                    </div>
                    <div class="col-auto">
                      <i class="fas fa-comments fa-2x text-gray-300"></i>
                    </div>
```

```
              </div>
            </div>
          </div>
        </div>

        <div class="col-xl-4 col-md-4 mb-4">
          <div class="card border-left-danger shadow h-100 py-2">
            <div class="card-body">
              <div class="row no-gutters align-items-center">
                <div class="col mr-2">
                  <div class="text-xs font-weight-bold text-danger text-uppercase mb-1">Pending
Orders</div>
                  <div class="h5 mb-0 font-weight-bold text-gray-800">{{ pending_total }}</div>
                </div>
                <div class="col-auto">
                  <i class="fas fa-comments fa-2x text-gray-300"></i>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>

      <div class="row">
        <div class="col-xl-6 col-lg-6">
          <div class="card shadow mb-4">
            <div class="card-header py-3 d-flex flex-row align-items-center justify-content-
    between">
              <h6 class="m-0 font-weight-bold text-primary">Categories Bar Chart</h6>
            </div>
            <div class="card-body">
              <div class="chart-pie pt-4 pb-2">
                <canvas id="myChart"></canvas>
              </div>
            </div>
          </div>
        </div>
        <div class="col-xl-6 col-lg-6">
          <div class="card shadow mb-4">
            <div class="card-header py-3 d-flex flex-row align-items-center justify-content-
    between">
              <h6 class="m-0 font-weight-bold text-primary">Categories Doughnut Chart</h6>
            </div>
            <div class="card-body">
                <canvas id="myAreaChart"></canvas>
            </div>
          </div>
        </div>
      </div>
```

```
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>

    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>

    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous"></script>

    <script>

      let myChart = document.getElementById('myChart').getContext('2d');

      let myAreaChart = document.getElementById('myAreaChart').getContext('2d');

      let count1 = {{ count1 }};

      let count2 = {{ count2 }};

      let count3 = {{ count3 }};


      let orderChart1 = new Chart(myChart, {

        type:'bar',

        data:{

          labels:['Spicy Avocado', 'Tempura', 'Maki'],

          datasets:[{

            label:'Items',

            data:[

              count1,

              count2,
              count3,
            ],
            backgroundColor:[
              '#007BFF',
              '#28A745',
              '#FFC107',
            ],
          }]
        },
        options:{
          layout:{
            padding:{
              left:0,
              right:0,
              bottom:40,
              top:0
            }
          },
        }
      });
      let orderChart2 = new Chart(myAreaChart, {
        type:'doughnut',
        data:{
          labels:['Spicy Avocado', 'Tempura', 'Maki'],
          datasets:[{
            label:'Items',
            data:[
              count1,
              count2,
              count3,
            ],
            backgroundColor:[
              '#007BFF',
              '#28A745',
              '#FFC107',
            ],
            borderWidth:1,
            borderColor:'#777',
            hoverBorderWidth:1,
            hoverBorderColor:'#000'
          }]
        },
        options:{
          layout:{
            padding:{
              left:0,
              right:0,
              bottom:40,
              top:0
            }
          },
        }
      });
  </script>
</body>
</html>
```

Home.html

```
{% extends "main/layout.html" %}
{% block content %}
  <div class="home bg-dark">
    <div id="carouselExampleCaptions" class="carousel slide mb-5" data-ride="carousel"
style="height: 500px;">
        <ol class="carousel-indicators">
          <li data-target="#carouselExampleCaptions" data-slide-to="0" class="active"></li>
          <li data-target="#carouselExampleCaptions" data-slide-to="1"></li>
          <li data-target="#carouselExampleCaptions" data-slide-to="2"></li>
        </ol>
        <div class="carousel-inner">
          <div class="carousel-item active">
            <img src="../media/images/Banner4.webp" class="d-block w-100" alt="banner1"
height="500px">
            <div class="carousel-caption d-none d-md-block text-light">
              <h1>Game Store</h1>
            </div>
          </div>
          <div class="carousel-item">
            <img src="../media/images/Banner1.jpg" class="d-block w-100" alt="sushi2"
height="500px">
          </div>
          <div class="carousel-item">
            <img src="../media/images/Banner3.jpg" class="d-block w-100" alt="sushi3"
height="500px">
          </div>
        </div>
        <a class="carousel-control-prev" href="#carouselExampleCaptions" role="button" data-
slide="prev">
          <span class="carousel-control-prev-icon" aria-hidden="true"></span>
          <span class="sr-only">Previous</span>
        </a>
        <a class="carousel-control-next" href="#carouselExampleCaptions" role="button" data-
slide="next">
          <span class="carousel-control-next-icon" aria-hidden="true"></span>
          <span class="sr-only">Next</span>
        </a>
    </div>
    <div class="container">
      <h1 class="text-center text-light mb-3">Our Products</h1>
      <div class="row">
```

```
        {% for item in card_items %}
            <div class="col-sm-4">
                <div class="card mb-3" style="height: fit-content; overflow: auto;">
                  <img src="{{ item.image.url }}" style="height: 300px;">
                    <div class="card-body text-center">
                      <button class="btn btn-warning">By:{{ item.created_by }}</button>
                      <h5 class="text-center"><a href="{% url 'main:product' item.slug %}">{{
item.title }} </a><span class="badge badge-{{ item.label_colour }}">{{ item.labels
}}</span></h5>
                      <p class="text-center">{{ item.description }}</p>
                      <h5 class="text-center">₹ {{ item.price }}</h5>
                      <a href="{% url 'main:product' item.slug %}" class="btn btn-info float-
right">View Details</a>
                    </div>
                </div>
            </div>
        {% endfor %}
      </div>
    </div>
  </div>
{% endblock content %}
```

Cart.html

```
{% extends "main/layout.html" %}
{% block content %}
```

```
<div class="cart">
    <div class="container">
        <div class="row">
            <div class="col-sm-9 mt-5">
                <h3 class="text-center">Your Cart</h3>
                {% if cart_items %}
                <div class="table-responsive">
                    <table class="table table-light">
                        <thead>
                        <tr>
                            <th scope="col">Name</th>
                            <th scope="col">Description</th>
                            <th scope="col">Specification</th>
                            <th scope="col">Price</th>
                            <th scope="col">Total keys</th>
                            <th scope="col"></th>
                        </tr>
                        </thead>
                        <tbody>
                            {% for cart in cart_items %}
                        <tr>
                            <td>{{ cart.item.title }} <span class="badge badge-{{ cart.item.label_colour
}}">{{ cart.item.labels }}</span></td>
                            <td>{{ cart.item.description }}</td>
                            <td>{{ cart.item.instructions }}</td>
                    <td>{{ cart.item.price }}</td>
                    <td>{{ cart.item.keys }} keys</td>
                    <td><a class="btn btn-danger" href="{% url 'main:remove-from-cart' cart.id
%}">Delete😥</a></td>
                  </tr>
                    {% endfor %}
                  </tbody>
                </table>
              </div>
              {% else %}
              <h4 class="text-center">Sorry, your cart is empty..Please add items!!!</h4>
              {% endif %}
            </div>
            <div class="col-sm-3 mt-5 mb-3">
              <div class="content-section">
                <h3 class="text-center">Order Summary</h3>
                  <ul class="list-group">
                    <li class="list-group-item list-group-item-light text-center text-dark">Total: {{
total }}</li>
                    <li class="list-group-item list-group-item-light text-center text-dark">Number of
Boxes: {{ count }}</li>
                    <li class="list-group-item list-group-item-light text-center text-dark">Total keys:
{{ total_keys }}</li>
                    <li class="list-group-item list-group-item-light text-center text-dark"><a
class="btn btn-info text-dark" href="{% url 'main:ordered' %}">Proceed To
Checkout🤝</a></li>
```

```
                        <li class="list-group-item list-group-item-light text-center"><a class="btn btn-
warning text-dark" href="/">Continue Buying ✌ </a></li>

                      </ul>

                </div>

            </div>

        </div>

    </div>

</div>

{% endblock content %}
```

Product.html

```
{% extends "main/layout.html" %}

{% block content %}


<div class="product">

    <!--Main layout-->

  <main class="mt-3 pt-4 main">

    <div class="container dark-grey-text">


      <div class="row wow fadeIn">

        <object style="height: 300px; width: 500px"><param name="movie"
value="{{item.instructions}}"><param name="allowFullScreen" value="true"><param
name="allowScriptAccess" value="always"><embed src="{{item.instructions}}"
type="application/x-shockwave-flash" allowfullscreen="true" allowScriptAccess="always"
width="500" height="300"></object>

        <div class="col-md-6 mb-4">
```

```
     <img src="{{ item.image.url }}" class="img-fluid mt-4" alt="">
    </div>
    <div class="col-md-6 mb-4 information">
     <!--Content-->
     <div class="container"
     <div class="p-4 content">
      <div class="mt-0">
        <h2><u>{{ item.title }}</u><span class="badge badge-{{ item.label_colour }} ml-
2">{{ item.labels }}</span></h2>
      </div>
      <div class="lead">
       {% if item.description %}
       <h5>{{ item.description }}</h5>
       {% endif %}
       <h3>₹ {{ item.price }} per key</h3>
      </div>
      <a href="{% url 'main:add-to-cart' item.slug %}" class="btn btn-primary btn-md my-0
p">Add to cart</a>
     </div>
    </div>
   </div>
   <hr>
  </div>
 </main>
 <div class="container">
```

```
    <h1 class="text-center">Reviews</h1>


    <form action="{% url 'main:add_reviews' %}" method="POST" class="mb-3">
      {% csrf_token %}
      <div class="row">
        <div class="col-sm-11">
          <input type="text" name="review" class="form-control" placeholder="Enter Your
Review">
          <input type="hidden" name="rslug" class="form-control" value="{{item.slug}}">
        </div>
        <div class="col-sm-1">
          <button type="submit" class="btn btn-primary">Submit</button>
        </div>
      </div>
    </form>


    {% for review in reviews %}
    <div class="main-reviews card bg-success mb-3">
      <div class="card-body">
        <div class="user_details" style="display: flex; border-bottom: 1px solid rgb(31, 30, 30);">
          <h3 class="mr-3">{{ review.user.username }} </h3>
          <p class="mt-2">{{ review.posted_on }}</p>
        </div>
          <h4>{{ review.review }}</h4>
      </div>
    </div>
    {% endfor %}

  </div>
</div>
{% endblock content %}
```

Main/admin.py

```
from django.contrib import admin
from .models import Item, CartItems, Reviews
from django.db import models

class ItemAdmin(admin.ModelAdmin):
    fieldsets = [
        ("Created By", {'fields': ["created_by"]}),
        ("Title", {'fields': ["title"]}),
        ("Image", {'fields': ["image"]}),
        ("Description", {'fields': ["description"]}),
        ("Price", {'fields': ["price"]}),
        ("keys", {'fields': ["keys"]}),
        ("Instructions", {'fields': ["instructions"]}),
        ("Labels", {'fields': ["labels"]}),
        ("Label Colour", {'fields': ["label_colour"]}),
        ("Slug", {'fields': ["slug"]}),
    ]
    list_display = ('id','created_by','title','description','price','labels')


class CartItemsAdmin(admin.ModelAdmin):
    fieldsets = [
        ("Order Status", {'fields' : ["status"]}),
        ("Delivery Date", {'fields' : ["delivery_date"]})


    ]
    list_display = ('id','user','item','quantity','ordered','ordered_date','delivery_date','status')
    list_filter = ('ordered','ordered_date','status')


class ReviewsAdmin(admin.ModelAdmin):
    list_display = ('user','item','review','posted_on')


admin.site.register(Item,ItemAdmin)
admin.site.register(CartItems,CartItemsAdmin)
admin.site.register(Reviews,ReviewsAdmin)
```

main/models.py

```
from django.db import models
from django.conf import settings
from django.shortcuts import reverse
from django.utils import timezone
from django.contrib.auth.models import User
```

```
class Item(models.Model):
    LABELS = (
        ('BestSeller', 'BestSeller'),
        ('New', 'New'),
        ('Coming Soon', 'Coming Soon'),
    )

    LABEL_COLOUR = (
        ('danger', 'danger'),
        ('success', 'success'),
        ('primary', 'primary'),
        ('info', 'info')
    )
    title = models.CharField(max_length=150)
    description = models.CharField(max_length=250,blank=True)
    price = models.FloatField()
    keys = models.IntegerField(default=6)
    instructions = models.CharField(max_length=250)
    image = models.ImageField(default='default.png', upload_to='images/')
    labels = models.CharField(max_length=25, choices=LABELS, blank=True)
    label_colour = models.CharField(max_length=15, choices=LABEL_COLOUR, blank=True)
    slug = models.SlugField()
    created_by = models.ForeignKey(User, on_delete=models.CASCADE)

    def __str__(self):
        return self.title

    def get_absolute_url(self):
        return reverse("main:product", kwargs={
            'slug': self.slug
        })

    def get_add_to_cart_url(self):
        return reverse("main:add-to-cart", kwargs={
            'slug': self.slug
        })

    def get_item_delete_url(self):
        return reverse("main:item-delete", kwargs={
            'slug': self.slug
        })

    def get_update_item_url(self):
        return reverse("main:item-update", kwargs={
            'slug': self.slug
        })

class Reviews(models.Model):
    user = models.ForeignKey(User, on_delete = models.CASCADE)
    item = models.ForeignKey(Item, on_delete = models.CASCADE)
    rslug = models.SlugField()
    review = models.TextField()
    posted_on = models.DateField(default=timezone.now)

    class Meta:
        verbose_name = 'Review'
        verbose_name_plural = 'Reviews'

    def __str__(self):
        return self.review

class CartItems(models.Model):
    ORDER_STATUS = (
        ('Active', 'Active'),
        ('Delivered', 'Delivered')
    )
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    item = models.ForeignKey(Item, on_delete=models.CASCADE)
    ordered = models.BooleanField(default=False)
    quantity = models.IntegerField(default=1)
    ordered_date = models.DateField(default=timezone.now)
    status = models.CharField(max_length=20, choices=ORDER_STATUS, default='Active')
    delivery_date = models.DateField(default=timezone.now)

    class Meta:
        verbose_name = 'Cart Item'
        verbose_name_plural = 'Cart Items'
    def __str__(self):
        return self.item.title

    def get_remove_from_cart_url(self):
        return reverse("main:remove-from-cart", kwargs={
            'pk' : self.pk
        })

    def update_status_url(self):
        return reverse("main:update_status", kwargs={
            'pk' : self.pk
        })
```

Main/urls.py

```
from django.urls import path
from . import views
from .views import (
    cardListView,
    cardDetail,
    add_to_cart,
    get_cart_items,
    order_item,
    CartDeleteView,
    order_details,
    admin_view,
    item_list,
    pending_orders,
    ItemCreateView,
    ItemUpdateView,
    ItemDeleteView,
    update_status,
    add_reviews,
)

app_name = "main"

urlpatterns = [
    path('', cardListView.as_view(), name='home'),
    path('product/<slug>', views.cardDetail, name='product'),
    path('item_list/', views.item_list, name='item_list'),
    path('item/new/', ItemCreateView.as_view(), name='item-create'),
    path('item-update/<slug>/', ItemUpdateView.as_view(), name='item-update'),
    path('item-delete/<slug>/', ItemDeleteView.as_view(), name='item-delete'),
    path('add-to-cart/<slug>/', views.add_to_cart, name='add-to-cart'),
    path('cart/', views.get_cart_items, name='cart'),
    path('remove-from-cart/<int:pk>/', CartDeleteView.as_view(), name='remove-from-cart'),
    path('ordered/', views.order_item, name='ordered'),
    path('order_details/', views.order_details, name='order_details'),
    path('admin_view/', views.admin_view, name='admin_view'),
    path('pending_orders/', views.pending_orders, name='pending_orders'),
    path('admin_dashboard/', views.admin_dashboard, name='admin_dashboard'),
    path('update_status/<int:pk>', views.update_status, name='update_status'),
    path('postReview', views.add_reviews, name='add_reviews'),
]
```

Main/views.py

```
from django.shortcuts import render, get_object_or_404, redirect
from .models import Item, CartItems, Reviews
from django.contrib import messages
from django.views.generic import (
    ListView,
    DetailView,
    CreateView,
    UpdateView,
    DeleteView,
)
from django.utils import timezone
from django.contrib.auth.decorators import login_required
from django.contrib.auth.mixins import LoginRequiredMixin, UserPassesTestMixin
from .decorators import *
from django.db.models import Sum


class cardListView(ListView):
    model = Item
    template_name = 'main/home.html'
    context_object_name = 'card_items'
```

```
def cardDetail(request, slug):
    item = Item.objects.filter(slug=slug).first()
    reviews = Reviews.objects.filter(rslug=slug).order_by('-id')[:7]
    context = {
        'item' : item,
        'reviews' : reviews,
    }
    return render(request, 'main/product.html', context)


@login_required
def add_reviews(request):
    if request.method == "POST":
        user = request.user
        rslug = request.POST.get("rslug")
        item = Item.objects.get(slug=rslug)
        review = request.POST.get("review")


        reviews = Reviews(user=user, item=item, review=review, rslug=rslug)
        reviews.save()
        messages.success(request, "Thankyou for reviewing this product!!")
    return redirect(f"/product/{item.slug}")


class ItemCreateView(LoginRequiredMixin, CreateView):
    model = Item
    fields = ['title', 'image', 'description', 'price', 'keys', 'instructions', 'labels', 'label_colour', 'slug']
    def form_valid(self, form):
        form.instance.created_by = self.request.user
        return super().form_valid(form)

class ItemUpdateView(LoginRequiredMixin, UserPassesTestMixin, UpdateView):
    model = Item
    fields = ['title', 'image', 'description', 'price', 'keys', 'instructions', 'labels', 'label_colour', 'slug']

    def form_valid(self, form):
        form.instance.created_by = self.request.user
        return super().form_valid(form)

    def test_func(self):
        item = self.get_object()
        if self.request.user == item.created_by:
            return True
        return False

class ItemDeleteView(LoginRequiredMixin, UserPassesTestMixin, DeleteView):
    model = Item
    success_url = '/item_list'

    def test_func(self):
        item = self.get_object()
        if self.request.user == item.created_by:
            return True
        return False


@login_required
def add_to_cart(request, slug):
    item = get_object_or_404(Item, slug=slug)
    cart_item = CartItems.objects.create(
        item=item,
        user=request.user,
        ordered=False,
    )
    messages.info(request, "Added to Cart!!Continue Shopping!!")
    return redirect("main:cart")


@login_required
def get_cart_items(request):
    cart_items = CartItems.objects.filter(user=request.user,ordered=False)
    bill = cart_items.aggregate(Sum('item__price'))
    number = cart_items.aggregate(Sum('quantity'))
    keys = cart_items.aggregate(Sum('item__keys'))
    total = bill.get("item__price__sum")
    count = number.get("quantity__sum")
    total_keys = keys.get("item__keys__sum")
    context = {
        'cart_items':cart_items,
        'total': total,
        'count': count,
        'total_keys': total_keys
    }
    return render(request, 'main/cart.html', context)


class CartDeleteView(LoginRequiredMixin, UserPassesTestMixin, DeleteView):
    model = CartItems
    success_url = '/cart'


    def test_func(self):
        cart = self.get_object()
        if self.request.user == cart.user:
            return True
        return False


@login_required
def order_item(request):
    cart_items = CartItems.objects.filter(user=request.user,ordered=False)
    ordered_date=timezone.now()
    cart_items.update(ordered=True,ordered_date=ordered_date)
    messages.info(request, "Item Ordered")
    return redirect("main:order_details")


@login_required
def order_details(request):
    items = CartItems.objects.filter(user=request.user, ordered=True,status="Active").order_by('-
ordered_date')
```

```
    cart_items = CartItems.objects.filter(user=request.user,
ordered=True,status="Delivered").order_by('-ordered_date')

    bill = items.aggregate(Sum('item__price'))

    number = items.aggregate(Sum('quantity'))

    keys = items.aggregate(Sum('item__keys'))

    total = bill.get("item__price__sum")

    count = number.get("quantity__sum")

    total_keys = keys.get("item__keys__sum")

    context = {

        'items':items,

        'cart_items':cart_items,

        'total': total,

        'count': count,

        'total_keys': total_keys

    }

    return render(request, 'main/order_details.html', context)


@login_required(login_url='/accounts/login/')

@admin_required

def admin_view(request):

    cart_items = CartItems.objects.filter(item__created_by=request.user,
ordered=True,status="Delivered").order_by('-ordered_date')

    context = {

        'cart_items':cart_items,

    }

    return render(request, 'main/admin_view.html', context)
```

```
@login_required(login_url='/accounts/login/')
@admin_required
def item_list(request):
    items = Item.objects.filter(created_by=request.user)
    context = {
        'items':items
    }
    return render(request, 'main/item_list.html', context)


@login_required
@admin_required
def update_status(request,pk):
    if request.method == 'POST':
        status = request.POST['status']
    cart_items = CartItems.objects.filter(item__created_by=request.user,
ordered=True,status="Active",pk=pk)
    delivery_date=timezone.now()
    if status == 'Delivered':
        cart_items.update(status=status, delivery_date=delivery_date)
    return render(request, 'main/pending_orders.html')


@login_required(login_url='/accounts/login/')
@admin_required
def pending_orders(request):
    items = CartItems.objects.filter(item__created_by=request.user,
ordered=True,status="Active").order_by('-ordered_date')
    context = {
        'items':items,
    }
    return render(request, 'main/pending_orders.html', context)


@login_required(login_url='/accounts/login/')
@admin_required
def admin_dashboard(request):
    cart_items = CartItems.objects.filter(item__created_by=request.user, ordered=True)
    pending_total = CartItems.objects.filter(item__created_by=request.user,
ordered=True,status="Active").count()
    completed_total = CartItems.objects.filter(item__created_by=request.user,
ordered=True,status="Delivered").count()
    count1 = CartItems.objects.filter(item__created_by=request.user,
ordered=True,item="3").count()
    count2 = CartItems.objects.filter(item__created_by=request.user,
ordered=True,item="4").count()
    count3 = CartItems.objects.filter(item__created_by=request.user,
ordered=True,item="5").count()
    total = CartItems.objects.filter(item__created_by=request.user,
ordered=True).aggregate(Sum('item__price'))
    income = total.get("item__price__sum")
    context = {
        'pending_total' : pending_total,
        'completed_total' : completed_total,
        'income' : income,
        'count1' : count1,
        'count2' : count2,
        'count3' : count3,
    }
    return render(request, 'main/admin_dashboard.html', context)
```

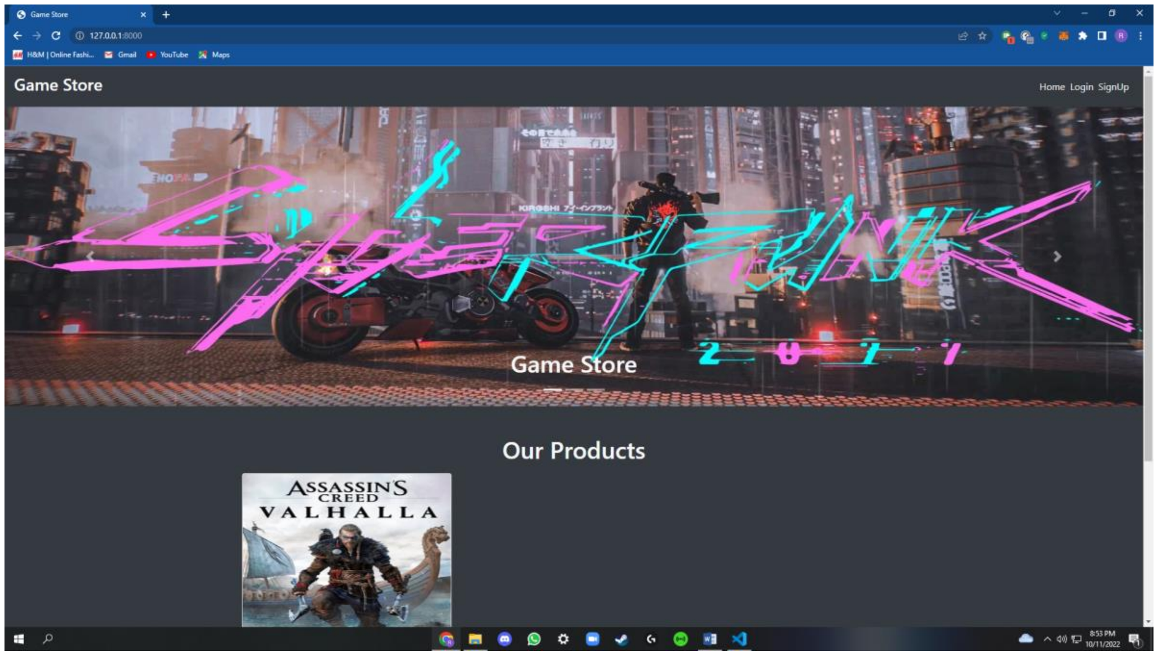
manage.py

```
import os
import sys



def main():
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'e_food.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)



if __name__ == '__main__':
    main()
```

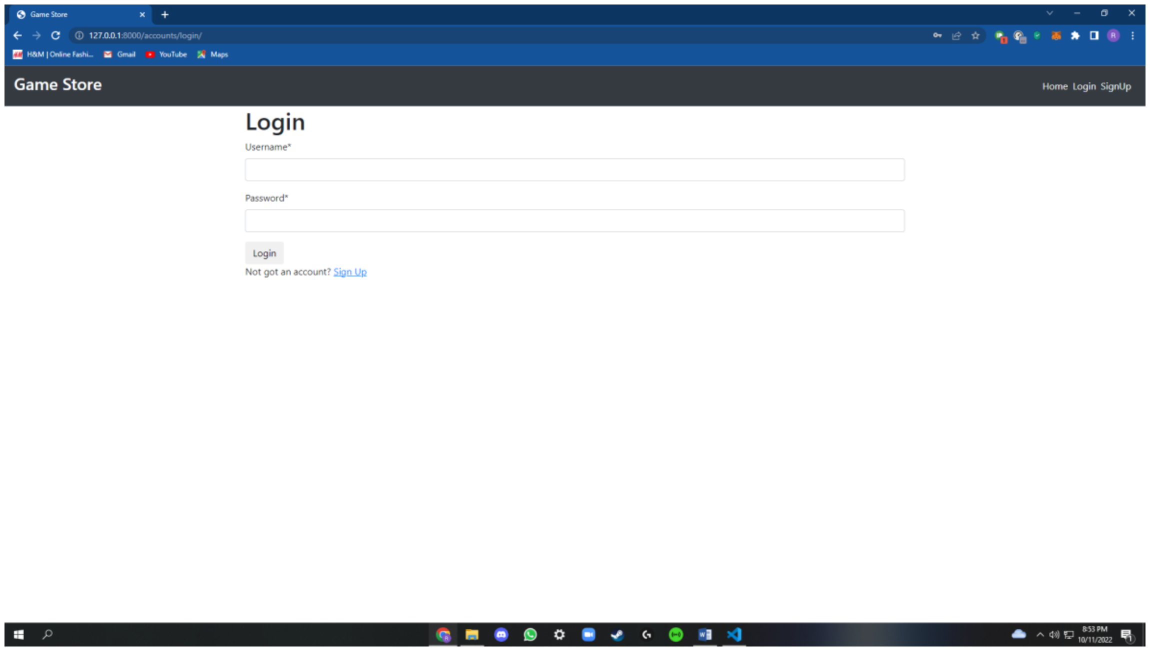
8. SCREEN SHOTS

8. SCREENSHOTS

- **Home**



- **Login**



Login

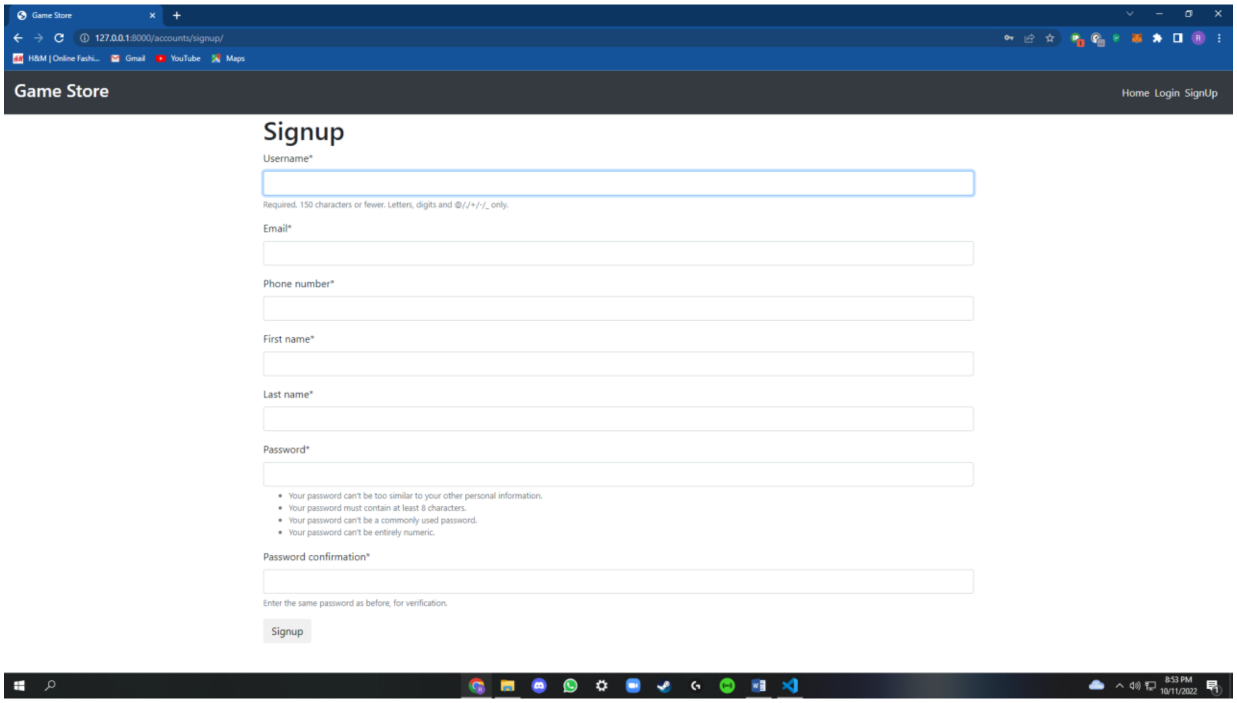
Username*

Password*

Login

Not got an account? Sign Up

- **Sign-Up**



Game Store

Home Login SignUp

Signup

Username*

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email*

Phone number*

First name*

Last name*

Password*

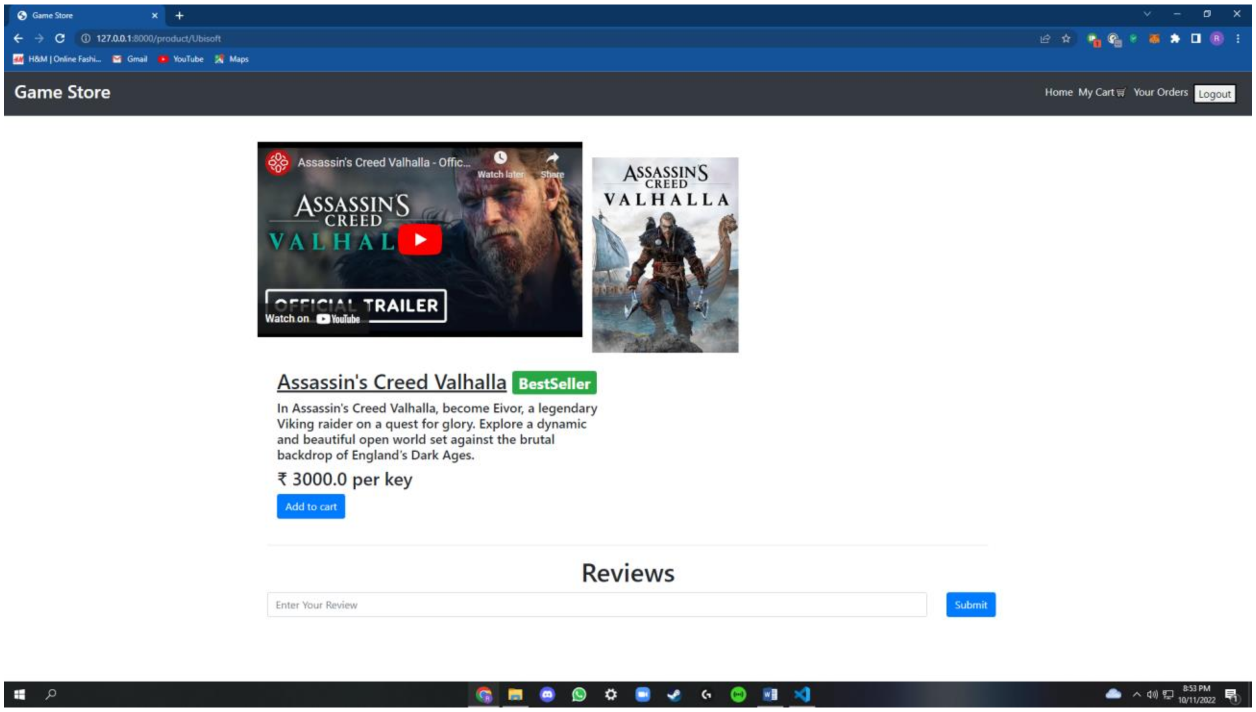
- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation*

Enter the same password as before, for verification.

Signup

- **Product Details**



Game Store

Home My Cart Your Orders Logout

Assassin's Creed Valhalla BestSeller

In Assassin's Creed Valhalla, become Eivor, a legendary Viking raider on a quest for glory. Explore a dynamic and beautiful open world set against the brutal backdrop of England's Dark Ages.

₹ 3000.0 per key

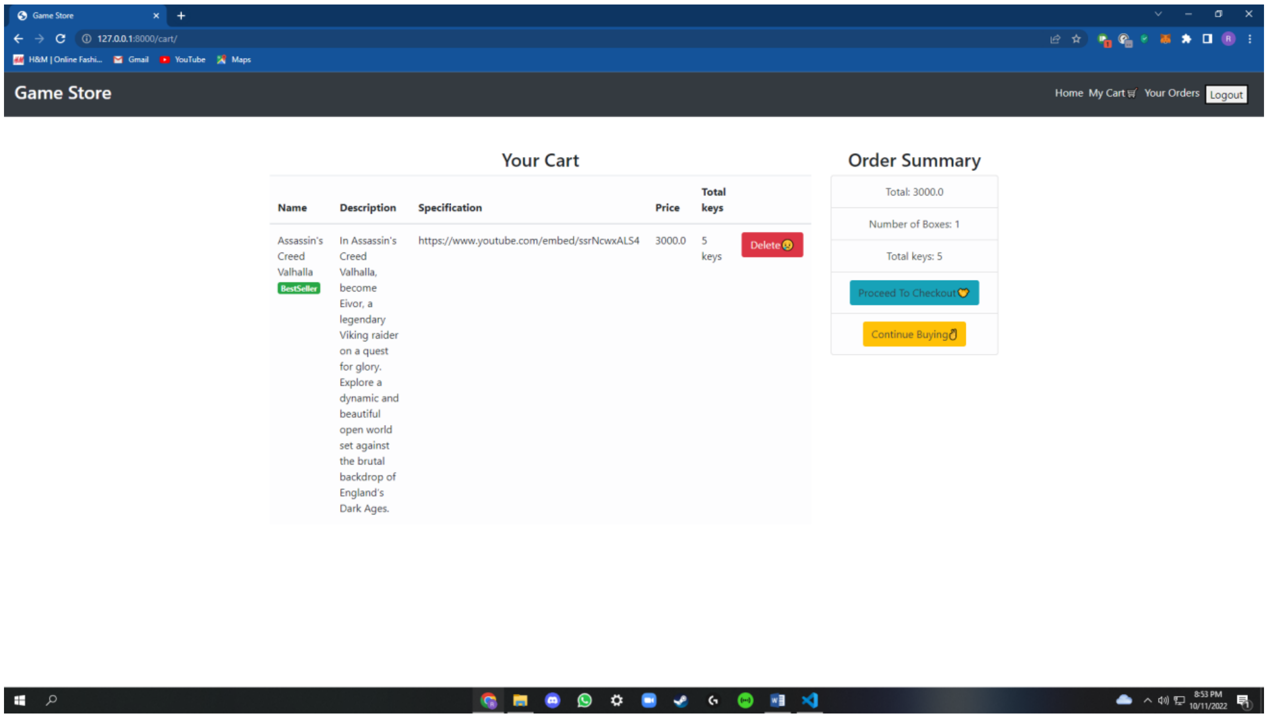
Add to cart

Reviews

Enter Your Review

Submit

- **Cart**



Your Cart

Name	Description	Specification	Price	Total keys	
Assassin's Creed Valhalla BestSeller	In Assassin's Creed Valhalla, become Eivor, a legendary Viking raider on a quest for glory. Explore a dynamic and beautiful open world set against the brutal backdrop of England's Dark Ages.	https://www.youtube.com/embed/ssrNcwxALS4	3000.0	5 keys	Delete

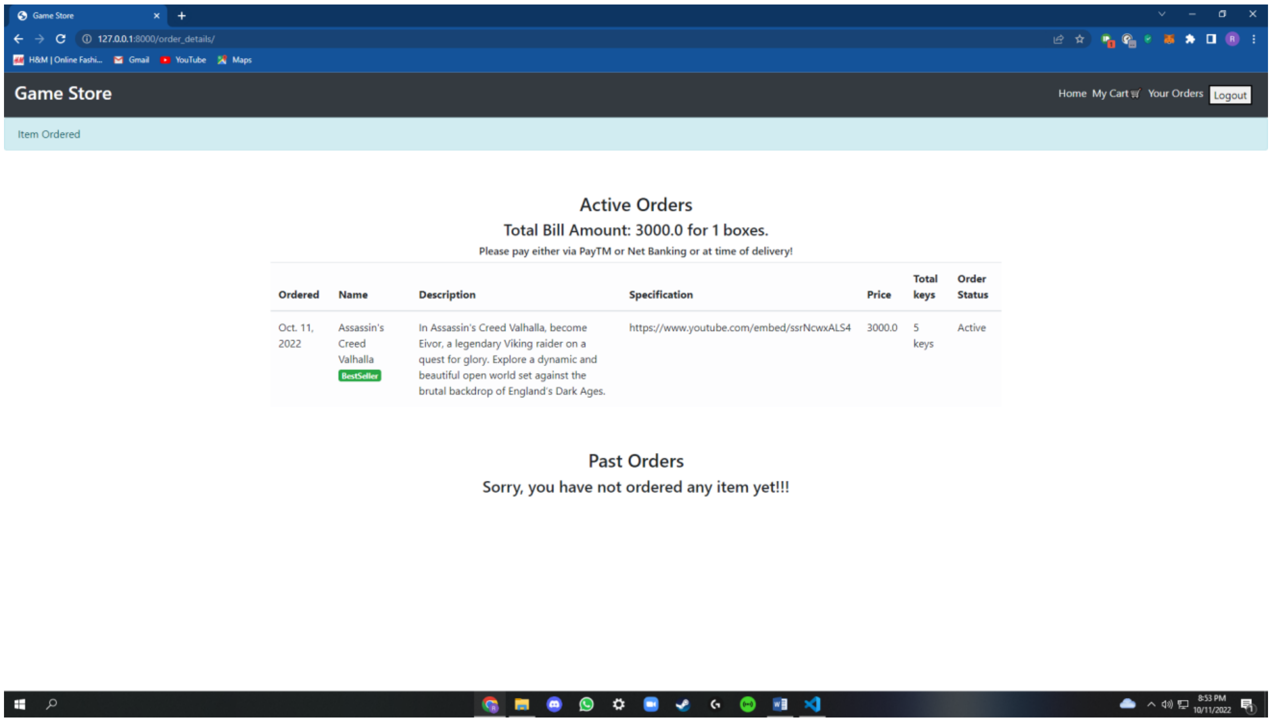
Order Summary

Total: 3000.0

Number of Boxes: 1

Total keys: 5

- **Order Details**


Game Store
127.0.0.1:8000/order_details/
Home My Cart Your Orders Logout

Item Ordered

Active Orders

Total Bill Amount: 3000.0 for 1 boxes.

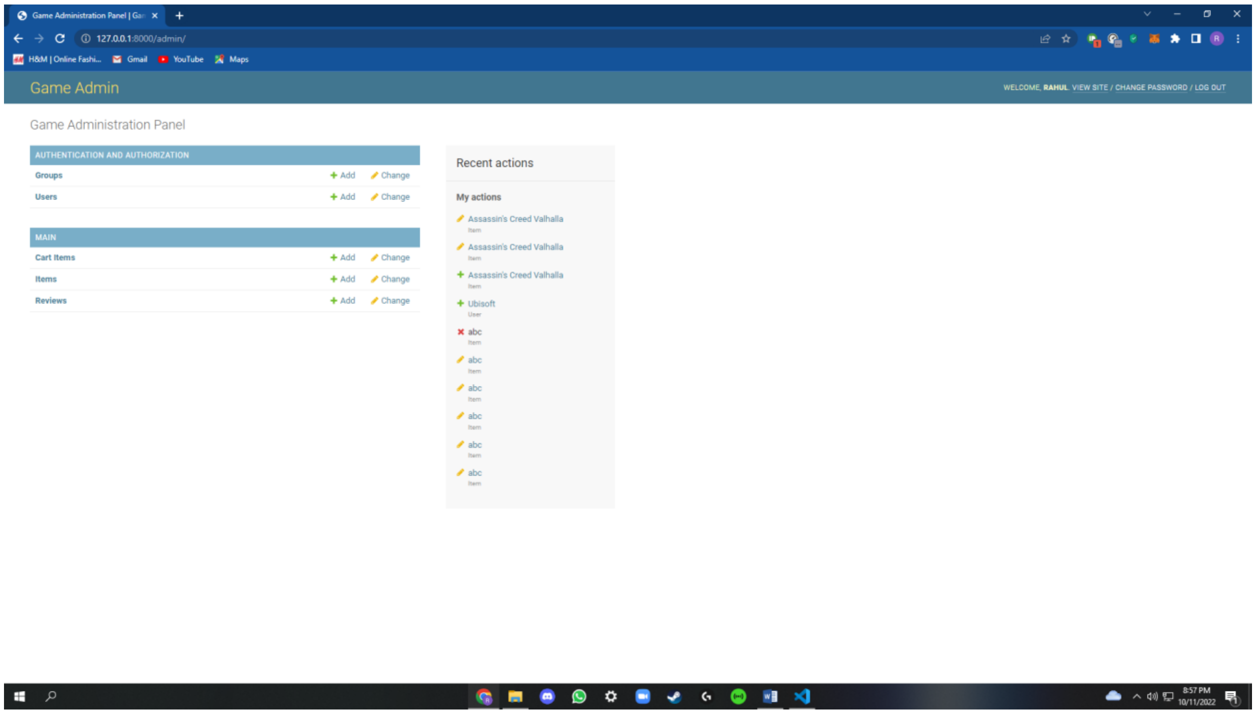
Please pay either via PayTM or Net Banking or at time of delivery!

Ordered	Name	Description	Specification	Price	Total keys	Order Status
Oct. 11, 2022	Assassin's Creed Valhalla BestSeller	In Assassin's Creed Valhalla, become Eivor, a legendary Viking raider on a quest for glory. Explore a dynamic and beautiful open world set against the brutal backdrop of England's Dark Ages.	https://www.youtube.com/embed/ssrNcwxALS4	3000.0	5 keys	Active

Past Orders

Sorry, you have not ordered any item yet!!!

- **Admin Page**



Game Admin

WELCOME, RAHUL. VIEW SITE / CHANGE PASSWORD / LOG OUT

Game Administration Panel

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

MAIN		
Cart Items	+ Add	Change
Items	+ Add	Change
Reviews	+ Add	Change

Recent actions

My actions

- Assassin's Creed Valhalla
  Item
- Assassin's Creed Valhalla
  Item
- + Assassin's Creed Valhalla
  Item
- + Ubisoft
  User
- ✖ abc
  Item
- abc
  Item
- abc
  Item
- abc
  Item
- abc
  Item
- abc
  Item

8:57 PM
10/11/2022

9. LIMITATION AND SYSTEM ENVIRONMENT

9. LIMITATION AND SYSTEM ENVIRONMENT

Limitations:

Each and every system has some limitations that can restrict them to work in a Particular environment. Though our system is provided high amount of accuracy and quick data generation, there is still some limitations that led us to in greater deficiency on that system. Some of the limitations that is observed during the process is:

All work is done manually; there is a very high possibility of human errors

- More number of people has to be employed to write data in the registers.
- As everything is done manually it turns out to be a very time-consuming process.
- Queries are made by making a lot of manual checking from the registers.

Enhancements:

Since we are entering details of the games electronically in the" Game Store Website", data will be secured. Using this application, we can retrieve game orders with a single click. Thus, processing information will be faster. It guarantees accurate maintenance of game details. It easily reduces the record maintenance task and thus reduces the human effort and increases accuracy speed.

- Reduced entry work.
- Easy retrieval of information.
- Reduced errors due to human intervention.
- User friendly screens to enter the data.
- Portable and flexible for further enhancement
- Web enabled.
- Fast finding of information request.

10. BIBLIOGRAPHY

10. BIBIOGRAPHY

Websites:

www.google.com
www.codemiles.com
www.wikipedia.org
www.w3schools.org
www.codeguru.com
www.tutorialspoint.com