Django E-Commerce Web-App

Project Report Submitted in Partial Fulfilment of the Requirements for
the Degree of

Bachelor of Engineering
in
Computer Science & Engineering

Submitted by

Nitin Saini: (Roll No. 19UCSE4010)

Vimarsh Koul: (Roll No. 19UCSE4019)

Under the Mentorship of
Dr. Shrawan Ram
Assistant Professor

&

Under the Guidance of
Mr. Abhishek Gaur
Assistant Professor

Department of Computer Science & Engineering
MBM University, Jodhpur

July, 2022

This page was intentionally left blank.

Django E-Commerce Web-App

Project Report Submitted in Partial Fulfilment of the Requirements for the Degree of

Bachelor of Engineering
in
Computer Science & Engineering

Submitted by

Nitin Saini: (Roll No. 19UCSE4015)

Vimarsh Koul: (Roll No. 19UCSE4019)

Under the Mentorship of
Dr. Shrawan Ram
Assistant Professor

&

Under the Guidance of
Mr. Abhishek Gaur
Assistant Professor

Department of Computer Science & Engineering
MBM University, Jodhpur

July, 2022

This page was intentionally left blank.

Department of Computer Science & Engineering

M.B.M. University,
Ratanada, Jodhpur, Rajasthan, India –342011

CERTIFICATE

This is to certify that the work contained in this report entitled **"Django E-Commerce Web-App**" is submitted by the group members Mr. Nitin Saini (Roll. No: 19UCSE4010) and Mr. Vimarsh Koul, (Roll. No: 19UCSE4019) to the Department of Computer Science & Engineering, M.B.M. University, Jodhpur, for the partial fulfillment of the requirements for the degree of **Bachelor of Engineering** in **Computer Science & Engineering**.

They have carried out their work under my guidance. This work has not been submitted elsewhere for the award of any other degree or diploma.

The project work in our opinion has reached the standard fulfilling the requirements for the degree of Bachelor of Engineering in Computer Science in accordance with the regulations of the Institute.

Mr. Abhishek Gaur
Assistant professor
(Guide)
Dept. of Computer Science & Engg.
M.B.M. University, Jodhpur

Dr. Shrawan Ram
(Mentor)
Dept. of Computer Science & Engg.
M.B.M. University, Jodhpur

This page was intentionally left blank.

DECLARATION

we, ***Nitin Saini and Vimarsh Koul,*** hereby declare that this project titled **"*Django E-Commerce Web-App*"** is a record of original work done by us under the supervision and guidance of ***Mr. Abhishek Gaur***.

we further certify that this work has not formed the basis for the award of the Degree/Diploma/Associateship/Fellowship or similar recognition to any candidate of any university and no part of this report is reproduced as it is from any other source without appropriate reference and permission.

SIGNATURE OF STUDENT

(Nitin Saini)
8th Semester, CSE
Enroll. - 18R/41151
Roll No. - 19UCSE4010

SIGNATURE OF STUDENT

(Vimarsh Koul)
8th Semester, CSE
Enroll. - 18R/33035
Roll No. - 19UCSE4019

This page was intentionally left blank.

ACKNOWLEDGEMENT

We take the opportunity to express our gratitude to all who have provided their immense support and their valuable time in guiding us. It is due to their support, Guidance, Supervision, and Encouragement that We have successfully completed our Project Report on "***Django E-Commerce Web-App***". We are highly indebted to our guide Mr. Abhishek Gaur, Assistant Professor, and mentor Dr. Shrawan Ram, Assistant Professor, for their guidance and constant supervision as well as for providing necessary information regarding this Project.

This page was intentionally left blank.

ABSTRACT

This E-Commerce Web-App project is implemented in the Python-Django framework. The main aim of this project is to create an online shopping application like Amazone, Flipcart, etc. Using this application, users can shop for anything they want, and then the product will be delivered to their home, they don't need to go anywhere.

In this Web-App you can first the user register by entering their simple details or they can simply log in with google in a single click for that we are using google OAuth2.0. And after that users can search for the product they want and then add that item to their cart and further they can buy those products. And users can also share their views on the product.

To create this web app we are using Django which is a Python framework that is used for web development and SQLite database to store all the data. For the front end, we are using HTML, CSS, and Javascript.

This page was intentionally left blank.

Table of Contents

Chapter 1: INTRODUCTION **1**

1.1 Overall Description 1

1.2 Purpose 1

1.2.1 Functionalities 1

1.3 Motivation and Scope 2

1.3.1 Aim 2

1.4 PROBLEM STATEMENT 2

Chapter 2: Technology Used **5**

2.1 Django 5

2.1.1. History Of Django:- 7

2.1.2. What does Django code look like:- 7

2.2. DRF 9

2.2.1. Introduction to Rest API 9

2.2.2. Need of using DRF 10

2.3. Git & GitHub 11

2.3.1. Introduction to Git: 11

2.3.2. Feature of Git: 11

2.3.3. Benefits of Git: 14

2.3.4. Git Command 15

2.4. OAuth 2.0 17

2.4.1 Principles of OAuth2.0 18

2.4.2 Roles 18

2.5 HTML, CSS, Javascript 19

2.5.1 HTML 19

2.5.2 CSS 20

2.5.3 Javascript 20

Chapter 3: Work Done

3.1. Product Listing category-wise

3.2. Register and log in

3.3. Add items to the cart And Place an order

Chapter 4: Conclusion & Future Work **21**

4.1 Conclusion 25

4.2 Future Enhancements 25

References 27

List of Figures

2.1. Api life cycle in Django……8

2.2 A typical Django application that uses React as a front end. It needs an API to allow React to consume data from the database……...10

2.3: Features of Git……..12

2.4: Distributed version control System……...13

2.5: Benefits of Git……...14

2.6. Paytm payment workflow…….22

3.1 E-commerce web app home page……..23

3.2 Items list page……24

3.3. User Registration page…….24

3.4. Login with google……25

3.5. User Cart view…….26

3.6. Buy items…….26

3.7. Pay online using Paytm……27

This page was intentionally left blank.

Chapter 1: INTRODUCTION

The “Django E-Commerce Web Application” has been developed to override the problems prevailing in the practicing manual system. This application is supported to eliminate, and in some cases reduce, the hardships faced by the existing systems. Moreover, this web app is designed for the particular need to carry out operations in a smooth and effective manner. This application is reduced as much as possible to avoid errors while entering the data. It also provides an error message while entering invalid data. No formal knowledge is needed for the user to use this system. Thus, this proves it is user-friendly. Using this can lead to an error-free, secure, reliable, and fast management system.

1.1 Overall Description

The main aim of this application is to an online platform through which users can buy anything that they want, using this web application first users have to register into the web app and by entering some details or simply login into google by a simple click and then users can search for the products that they want to buy and then users can put those products into their cart and then they can simply buy the product.

1.2 Purpose

The main purpose of the project is to provide an online shopping platform this app will also manage the cart history and the order history of the user. Using this app users can simply search for the product they want and also show the product categories wise which also help the users to buy similar kind of items.

1.2.1 Functionalities

Functionalities provided by the Django E-Commerce Web App are as follows:

- User Registration.
- User login and log in with Google.
- Search for the products.
- View product.
- Add product to cart.

- Buy products.
- Review product

1.3 Motivation and Scope

The gist of the idea is to digitalize the process of shopping and manually go to the market and buy products. There is some web app that exists in the market which is also doing the same like Amazone, Flipcart, etc. And in this project, we also take some references from these websites as well.

1.3.1 Aim

Our project aims to make an E-commerce web app through which users can buy products easily.

- To utilize resources in an efficient manner by increasing their productivity through automation.
- It satisfies the user requirement.
- Be easy to understand by the user and operator
- Be easy to operate.
- Have a good user interface
- Be expandable
- Delivered on schedule within the budget.

1.4 PROBLEM STATEMENT

The old manual ways were suffering from a series of drawbacks. In the old scenario, users have to go to the market in order to buy something and it will take too much effort users also need to go to the different shops for different products. Also sometimes they have to pay more money on transport as compared to the product they are purchasing.

So we have to create a web-based application through which users can simply register and start shopping. In the app, users can simply search for the product that they want to buy so it also saves a lot of time. Also, they can easily place an order for the products and can also pay the money online or can also choose the pay cash on delivery.

This page was intentionally left Blank

Chapter 2: Technology Used

In the backend development we used several technologies to create RESTful Apis. In our project, We used Django which is a Python-based framework and on top of Django, we also used Django Rest Framework to make the restful APIs and also used the version control tool i.e. git, and a centralized repository i.e. GitHub.

2.1 Django

Django is a high-level Python web framework that enables the rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It is free and open-source, has a thriving and active community, great documentation, and many options for free and paid-for support.

Django helps you write software that is:

- Complete:- Django follows the "Batteries included" philosophy and provides almost everything developers might want to do "out of the box". Because everything you need is part of the one "product", it all works seamlessly together, follows consistent design principles, and has extensive and up-to-date documentation.
- Versatile:- Django can be (and has been) used to build almost any type of website — from content management systems and wikis, through to social networks and news sites. It can work with any client-side framework and can deliver content in almost any format (including HTML, RSS feeds, JSON, XML, etc). Internally, while it provides choices for almost any functionality you might want (e.g. several popular databases, templating engines, etc.), it can also be extended to use other components if needed.
- Secure:- Django helps developers avoid many common security mistakes by providing a framework that has been engineered to "do the right things" to protect the website automatically. For example, Django provides a secure way to

to manage user accounts and passwords, avoiding common mistakes like putting session information in cookies where it is vulnerable (instead cookies just contain a key, and the actual data is stored in the database) or directly storing passwords rather than a password hash. A password hash is a fixed-length value created by sending the password through a cryptographic hash function. Django can check if an entered password is correct by running it through the hash function and comparing the output to the stored hash value. However due to the "one-way" nature of the function, even if a stored hash value is compromised it is hard for an attacker to work out the original password. Django enables protection against many vulnerabilities by default, including SQL injection, cross-site scripting, cross-site request forgery, and clickjacking (see Website security for more details of such attacks).

- Scalable:- Django uses a component-based "shared-nothing" architecture (each part of the architecture is independent of the others, and can hence be replaced or changed if needed). Having a clear separation between the different parts means that it can scale for increased traffic by adding hardware at any level: caching servers, database servers, or application servers. Some of the busiest sites have successfully scaled Django to meet their demands (e.g. Instagram and Disqus, to name just two).
- Maintainable:- Django code is written using design principles and patterns that encourage the creation of maintainable and reusable code. In particular, it makes use of the Don't Repeat Yourself (DRY) principle so there is no unnecessary duplication, reducing the amount of code. Django also promotes the grouping of related functionality into reusable "applications" and, at a lower level, groups related code into modules (along the lines of the Model View Controller (MVC) pattern).
- Portable:- Django is written in Python, which runs on many platforms. That means that you are not tied to any particular server platform, and can run your applications on many flavors of Linux, Windows, and macOS. Furthermore, Django is well-supported by many web-hosting providers, who often provide specific infrastructure and documentation for hosting Django sites.

2.1.1. History Of Django:-

Django was initially developed between 2003 and 2005 by a web team who were responsible for creating and maintaining newspaper websites. After creating a number of sites, the team began to factor out and reuse lots of common code and design patterns. This common code evolved into a generic web development framework, which was open-sourced as the "Django" project in July 2005.

Django has continued to grow and improve, from its first milestone release (1.0) in September 2008 through to the recently-released version 4.0 (2022). Each release has added new functionality and bug fixes, ranging from support for new types of databases, template engines, and caching, through to the addition of "generic" view functions and classes (which reduce the amount of code that developers have to write for a number of programming tasks).

2.1.2. What does Django code look like:-

In a traditional data-driven website, a web application waits for HTTP requests from the web browser (or other clients). When a request is received the application works out what is needed based on the URL and possibly information in POST data or GET data. Depending on what is required it may then read or write information from a database or perform other tasks required to satisfy the request. The application will then return a response to the web browser, often dynamically creating an HTML page for the browser to display by inserting the retrieved data into placeholders in an HTML template.

Django web applications typically group the code that handles each of these steps into separate files:

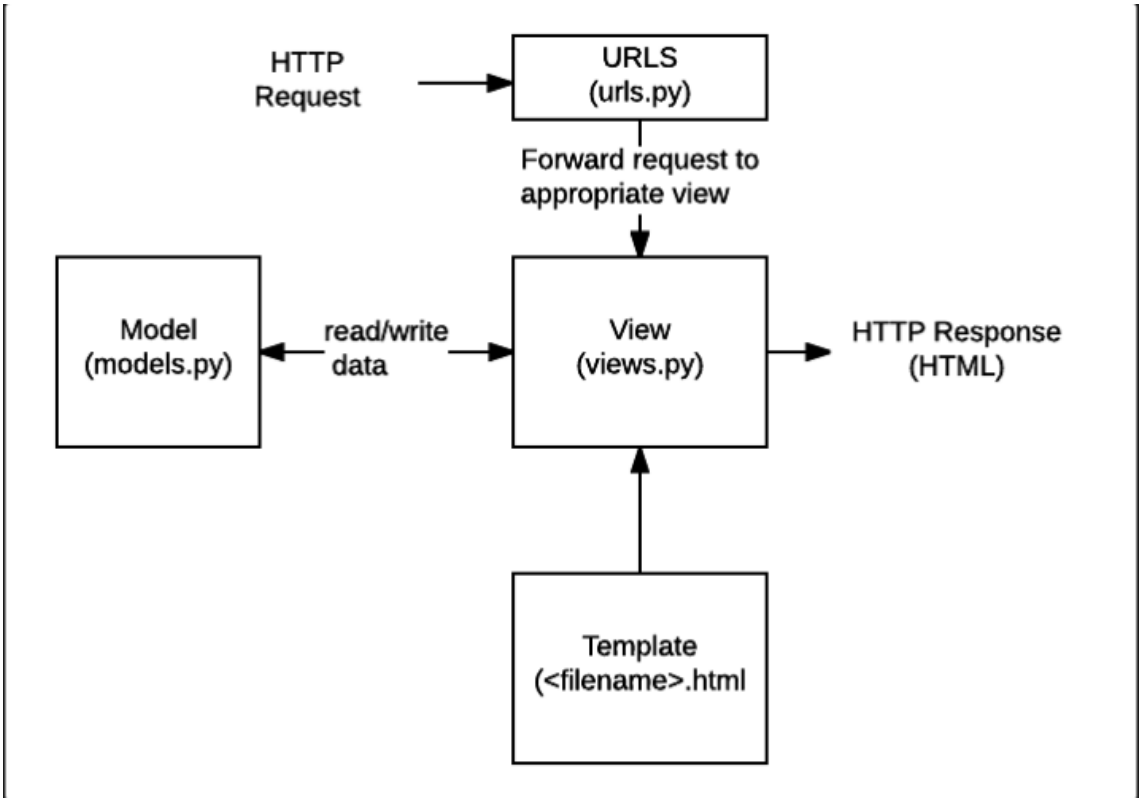


Fig 2.1: Api life cycle in Django

- URLs: While it is possible to process requests from every single URL via a single function, it is much more maintainable to write a separate view function to handle each resource. A URL mapper is used to redirect HTTP requests to the appropriate view based on the request URL. The URL mapper can also match particular patterns of strings or digits that appear in a URL and pass these to a view function as data.
- View: A view is a request handler function, which receives HTTP requests and returns HTTP responses. Views access the data needed to satisfy requests via models and delegate the formatting of the response to templates.
- Models: Models are Python objects that define the structure of an application's data and provide mechanisms to manage (add, modify, delete) and query records in the database.
- Templates: A template is a text file defining the structure or layout of a file (such as an HTML page), with placeholders used to represent actual content. A view

can dynamically create an HTML page using an HTML template, populating it with data from a model. A template can be used to define the structure of any type of file; it doesn't have to be HTML!

2.2. DRF

Django REST framework (DRF) is a powerful and flexible toolkit for building Web APIs. It is a framework that is built on top of Django. Although we can also create the REST Apis just by using Django as well the Django rest framework makes this work very easy for us.

2.2.1. Introduction to Rest API

A REST API is a standardized way to provide data to other applications. Those applications can then use the data however they want. Sometimes, APIs also offer a way for other applications to make changes to the data.

There are a few key options for a REST API request:

- GET — The most common option, returns some data from the API based on the endpoint you visit and any parameters you provide
- POST — Creates a new record that gets appended to the database
- PUT — Looks for a record at the given URI you provide. If it exists, update the existing record. If not, create a new record
- DELETE — Deletes the record at the given URI
- PATCH — Update individual fields of a record

Typically, an API is a window into a database. The API backend handles querying the database and formatting the response. What you receive is a static response, usually in JSON format, of whatever resource you requested.

REST APIs are so commonplace in software development, that it’s an essential skill for a developer to know how they work. APIs are how applications communicate with one another or even within themselves.

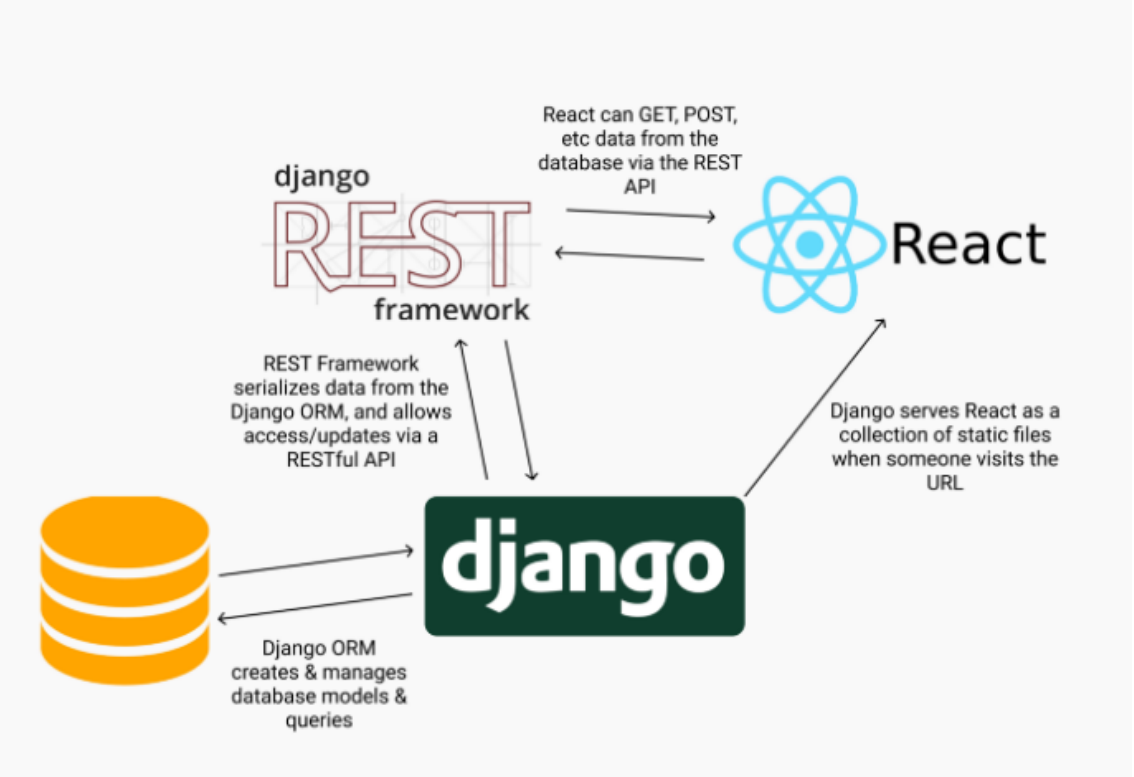


Fig 2.2: A typical Django application that uses React as a front end. It needs an API to allow React to consume data from the database.

2.2.2. Need of using DRF

Many frameworks allow you to easily build APIs for blog applications, but we will use only one – the Django REST framework (DRF). It's convenient in many ways and offers the following advantages:

- Its Web-browsable API is a huge usability win for your developers.
- Authentication policies include packages for OAuth1 and OAuth2.
- Serialization supports both ORM and non-ORM data sources.
- It's customizable all the way down. Just use regular function-based views if you don't need the more powerful features.
- It has extensive documentation and great community support.
- It's used and trusted by internationally recognized companies including Mozilla, Red Hat, Heroku, and Eventbrite.

2.3. Git & GitHub

2.3.1. Introduction to Git:

Git is an open-source distributed version control system. It is designed to handle minor to major projects with high speed and efficiency. It is developed to coordinate the work among the developers. The version control allows us to track and work together with our team members in the same workspace.

Git is the foundation of many services like GitHub and GitLab, but we can use Git without using any other Git services. Git can be used privately and publicly.

Git was created by Linus Torvalds in 2005 to develop Linux Kernel. It is also used as an important distributed version-control tool for DevOps. Git is easy to learn and has fast performance. It is superior to other SCM tools like Subversion, CVS, Perforce, and ClearCase.

2.3.2. Feature of Git:

Git is the most popular Version Control System nowadays because it is open-source software that is easy to handle and perform work on various projects.

Git allows a team of people to work together, all using the same files. And it helps the team cope with the confusion that tends to happen when multiple people are editing the same files. Git provides each developer a local copy of the entire development history, and changes are copied from one such repository to another.

Fig 2.3: Features of Git

1. Open-Source: Git is a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. It is called open-source because it provides the flexibility to modify its source code according to the user's needs. Unlike other Version Control Systems which provide paid functionalities like the repository space, privacy of codes, accuracy, speed, etc. Git is all open-source software that provides these functionalities for free and even in a better way than others.

Being open-source Git allows multiple people to work on the same project at the same time and collaborate with each other very easily and efficiently. Hence, Git is considered to be the best Version Control System available nowadays.

2. Distributed: Distributed systems are those which allow the users to perform work on a project from all over the world. A distributed system holds a Central repository that can be accessed by many remote collaborators by using a Version Control System. Git is one of the most popular Versions Control systems that is being used nowadays. Having a Central Server results in a problem of Data Loss or Data disconnectivity in case of a system failure of the central server. To tackle such kind of a situation, Git mirrors the whole repository on each snapshot of the version that is being pulled by the user. In this case, if the central server crashes, then the copy of repositories can be gained back from the users who have downloaded the latest snapshot of the project.

Having a distributed system, Git allows the users to work simultaneously on the same project, without interfering with others' work. When a particular user gets done with

their part of the code, they push the changes to the repository, and these changes get updated in the local copy of every other remote user who pulls the latest copy of the project.

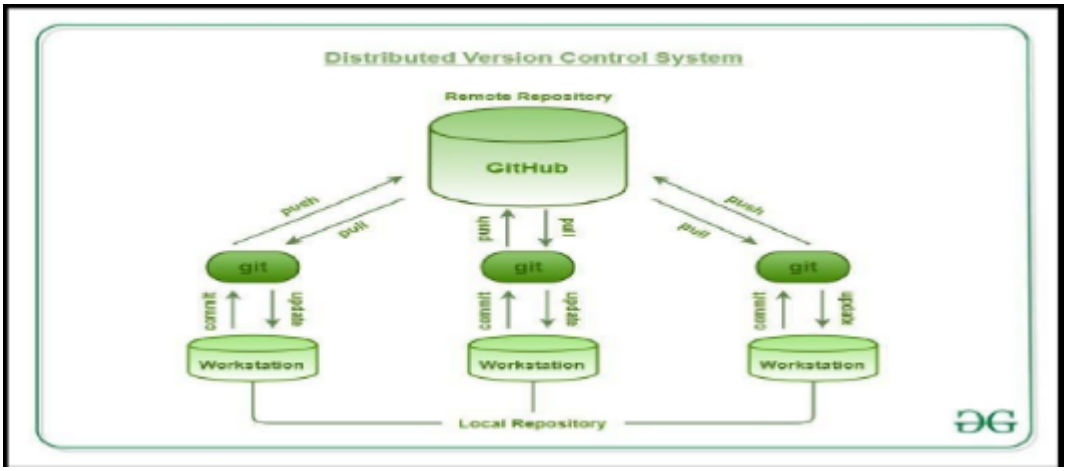


Fig 2.4: Distributed version control System

3. Security: Git is secure. It uses the SHA1 (Secure Hash Function) to name and identify objects within its repository. Files and commits are checked and retrieved by its checksum at the time of checkout. It stores its history in such a way that the ID of particular commits depends upon the complete development history leading up to that commit. Once it is published, one cannot make changes to its old version.

4. Speed: Since Git stores all the data related to a project in the local repository by the process of cloning, it is very much efficient to fetch data from the local repository instead of doing the same from the remote repository. Git is very fast and scalable compared to other version control systems which result in the handling of large projects efficiently.

The fetching power from a local repository is about 100 times faster than what is possible with the remote server. According to a test conducted by Mozilla, Git is one order of magnitude faster, which is about 10 times faster than other VCS tools. This is because Git is actually written in C language which is unlike other languages, very close to machine language and hence it makes processing very fast.

2.3.3. Benefits of Git:

A version control application allows us to keep track of all the changes that we make in the files of our project. Every time we make changes in files of an existing project, we can push those changes to a repository. Other developers are allowed to pull your changes from the repository and continue to work with the updates that you added to the project files.

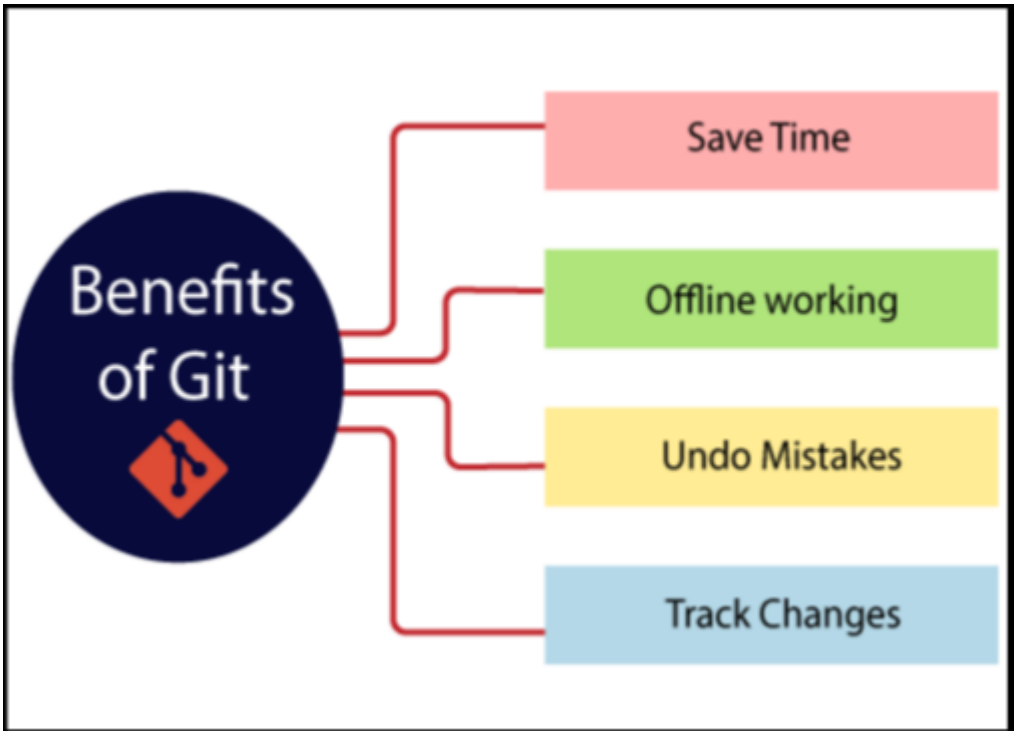


Fig 2.5: Benefits of Git

- **SavesTime:** Git is lightning-fast technology. Each command takes only a few seconds to execute so we can save a lot of time as compared to login into a GitHub account and finding out its features.
- **Offline:** Working One of the most important benefits of Git is that it supports offline working. If we are facing internet connectivity issues, it will not affect our work. In Git, we can do almost everything locally. Comparatively, other CVS like SVN is limited and prefer the connection with the central repository.
- **Undo:** MistakesOne additional benefit of Git is we can Undo mistakes. Sometimes the undo can be a savior option for us. Git provides the undo option for almost everything.

- **Track:** TheChangesGit facilitates some exciting features such as Diff, Log, and Status, which allows us to track changes so we can check the status, and compare our files or branches.

2.3.4. Git Command

1. git config :

   Usage: git config –global user.name “[name]”

   Usage: git config –global user.email “[email address]”

   This command sets the author name and email address respectively to be used with your commits.

2. git init :

   Usage: git init [repository name]

   This command is used to start a new repository.

3. git clone:

   Usage: git clone [url]

   This command is used to obtain a repository from an existing URL.

4. git add:

   Usage: git add [file]

   This command adds a file to the staging area.

   Usage: git add *

   This command adds one or more to the staging area.

5. git commit:

   Usage: git commit -m “[Type in the commit message]”

   This command records or snapshots the file permanently in the version history.

   Usage: git commit -a

This command commits any files you've added with the git add command and also commits any files you've changed since then.

6. git status :

   Usage: git status

   This command lists all the files that have to be committed.

7. git show :

   Usage: git show [commit]

   This command shows the metadata and content changes of the specified commit.

8. git branch :

   Usage: git branch

   This command lists all the local branches in the current repository.

   Usage: git branch[branch name]

   This command creates a new branch

   Usage: git branch -d [branch name]

   This command deletes the feature branch.

9. git checkout :

   Usage: git checkout [branch name]

   This command is used to switch from one branch to another.

   Usage: git checkout -b [branch name]

   This command creates a new branch and also switches to it.

10. git merge

    Usage: git merge [branch name]

    This command merges the specified branch's history into the current one

11. git push :

Usage: git push [variable name] master

This command sends the committed changes of master branch to your remote repository.

Usage: git push [variable name] [branch]

This command sends the branch commits to your remote repository.

Usage: git push –all [variable name]

This command pushes all branches to your remote repository.

Usage: git push [variable name] :[branch name]

This command deletes a branch on your remote repository.

12. git pull :

Usage: git pull [Repository Link]

This command fetches and merges changes on the remote server to your working directory.

2.4. OAuth 2.0

OAuth 2.0, which stands for “Open Authorization”, is a standard designed to allow a website or application to access resources hosted by other web apps on behalf of a user.

OAuth 2 is an authorization framework that enables applications to obtain limited access to user accounts on an HTTP service. It works by delegating user authentication to the service that hosts a user account and authorizing third-party applications to access that user account. OAuth 2 provides authorization flows for web and desktop applications, as well as mobile devices.

2.4.1 Principles of OAuth2.0

OAuth 2.0 is an authorization protocol and NOT an authentication protocol. As such, it is designed primarily as a means of granting access to a set of resources, for example, remote APIs or user data.

OAuth 2.0 uses Access Tokens. An **Access Token** is a piece of data that represents the authorization to access resources on behalf of the end-user. OAuth 2.0 doesn't define a specific format for Access Tokens. However, in some contexts, the JSON Web Token (JWT) format is often used. This enables token issuers to include data in the token itself. Also, for security reasons, Access Tokens may have an expiration date.

2.4.2 Roles

The idea of roles is part of the core specification of the OAuth2.0 authorization framework. These define the essential components of an OAuth 2.0 system and are as follows:

- **Resource Owner**: The user or system that owns the protected resources and can grant access to them.
- **Client**: The client is the system that requires access to the protected resources. To access resources, the Client must hold the appropriate Access Token.
- **Authorization Server**: This server receives requests from the Client for Access Tokens and issues them upon successful authentication and consent by the Resource Owner. The authorization server exposes two endpoints: the Authorization endpoint, which handles the interactive authentication and consent of the user, and the Token endpoint, which is involved in a machine-to-machine interaction.
- **Resource Server**: A server that protects the user's resources and receives access requests from the Client. It accepts and validates an Access Token from the Client and returns the appropriate resources to it.

2.5 HTML, CSS, Javascript

2.5.1 HTML

HTML stands for Hyper Text Markup Language. It's one of the fundamental technologies required for web development. It provides the base structure for a web page. HTML code ensures that all the content on a website is properly formatted. This is so our Internet browser can display the content as intended. Without HTML, a browser couldn't display text or load images and other elements.

HTML5, the most current version of HTML, specifies a large number of Application Programming Interfaces (API integration services) that can be used with JavaScript for a more interactive and dynamic website:

- Canvas: Canvas is an HTML5 element used to draw images and shapes and manipulate them. It can also be used for more complex cases such as game graphics and animations.
- Web Storage: Web Storage is used to store information right in the browser. Some examples of this would be storing user login information and saving user preferences for a website.
- Service workers: Service workers enable a script that keeps running in the background when a web page is opened and is mainly used in websites with offline capabilities. It makes pages available offline and allows for the use of web push notifications. It can send these notifications even when our browser isn't open.
- WebSocket: WebSocket allows for a persistent two-way connection between the user and the server. The most common use cases include chats and notifications in web apps.

2.5.2 CSS

Cascading Style Sheets, abbreviated as CSS, define the style and aesthetics of a web page. While HTML is used to structure a web page, CSS specifies the appearance of that structure. This includes page layouts, colors, fonts, and element positioning. If

HTML is the bones of the web page, CSS is the skin. It makes the Internet, and the website, look good.

2.5.3 Javascript

JavaScript (often shortened to JS) is a lightweight, interpreted, object-oriented language with first-class functions, and is best known as the scripting language for Web pages, but it's used in many non-browser environments as well.

JavaScript runs on the client-side of the web, which can be used to design / program how the web pages behave on the occurrence of an event. JavaScript is an easy-to-learn and also powerful scripting language, widely used for controlling web page behavior.

The Mozilla project provides two JavaScript implementations. The first-ever JavaScript was created by Brendan Eich at Netscape and has since been updated to conform to ECMA-262 Edition 5 and later versions. This engine, code-named SpiderMonkey, is implemented in C/C++. The Rhino engine, created primarily by Norris Boyd (also at Netscape) is a JavaScript implementation written in Java. Like SpiderMonkey, Rhino is ECMA-262 Edition 5 compliant.

Google's V8, which is used in the Google Chrome browser and recent versions of Opera browser. This is also the engine used by Node.js

2.6. Paytm Payment Integration:-

In this project, we have also used the Paytm payment through which users can pay online.

JS Checkout is an ideal solution for businesses who would like to collect payment on their platform with minimal coding. It offers a high level of customization to the merchant with no redirection to bank pages and enhanced merchant visibility. You can do this by simply integrating a few lines of the code snippet, prepared by Paytm exclusively for you. This enables you to integrate and start accepting payments within an hour.

With this solution, you'll be able to:

- Start accepting payments within minutes by integrating a few lines of the code snippet, prepared by Paytm
- Design the payments screen as per your theme - Adjust header, background, and text color.
- Improve success rate by removing redirection altogether - Complete payment on your own page
- Enhance business visibility - Display your logo on payment's page
- Control pay mode sequencing - Want to promote Debit Card over Net Banking transactions on your payment page? JS Checkout solution gives you the option to do so

The workflow while creating payment through Paytm is:-

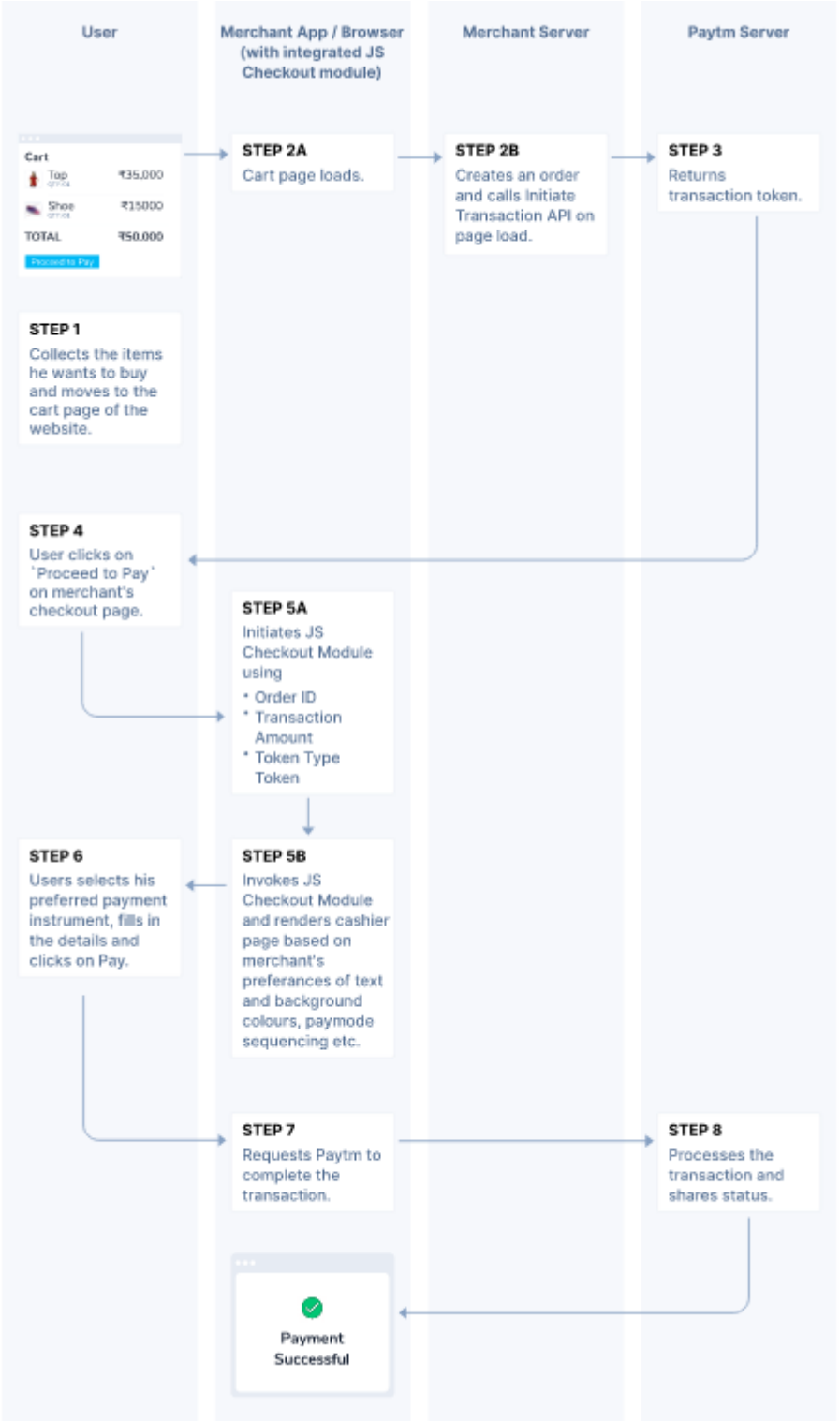
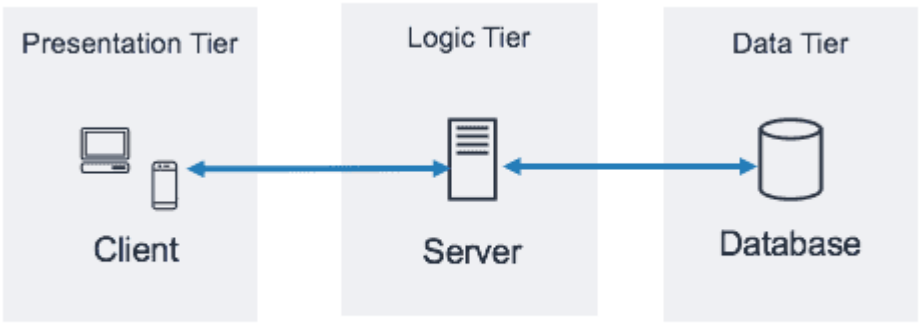


Fig:- 2.6. Paytm payment workflow.

Chapter 3: Work Done

Our E-Commerce web app is based on a three-tier architecture having a presentation tier, Logical tier, and Data tier.



3.1. Product Listing category-wise:-

This is also the home page of our application in this page users can see all the categories of the products and after clicking on a particular category they can see all the products which are lies inside that category.

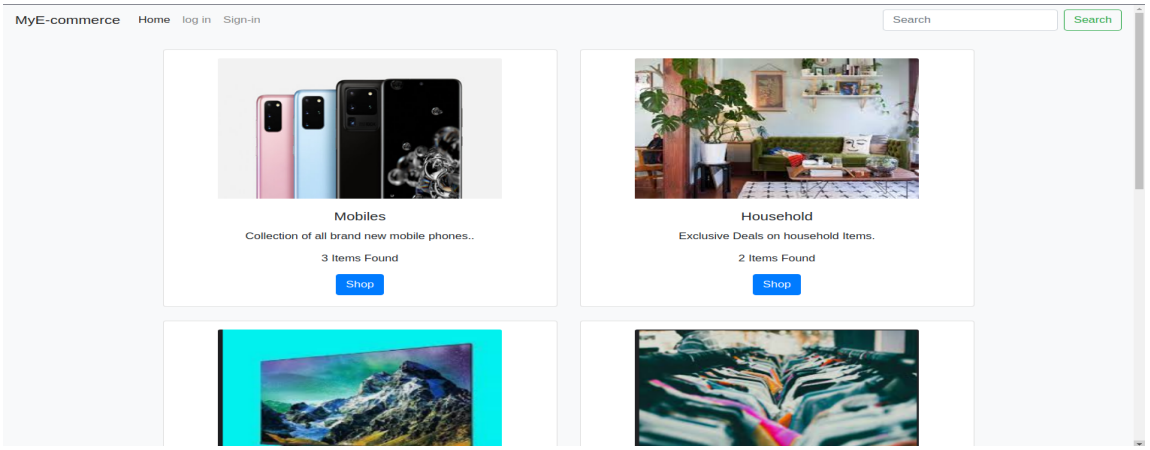


Fig: 3.1 E-commerce web app home page

once clicked on the shop button users can see all the items of that category.

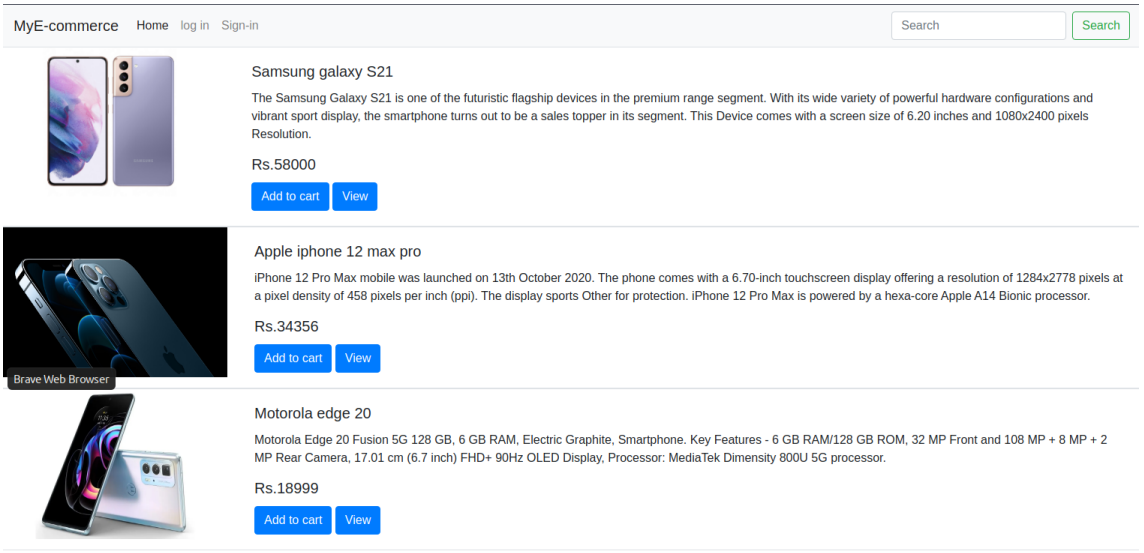


Fig 3.2 Items list page

3.2. Register and Login:-

In our application users can register and log in with google for that we are using OAuth2.0.

So for registration, users can simply go to the Sign-In Page and register.

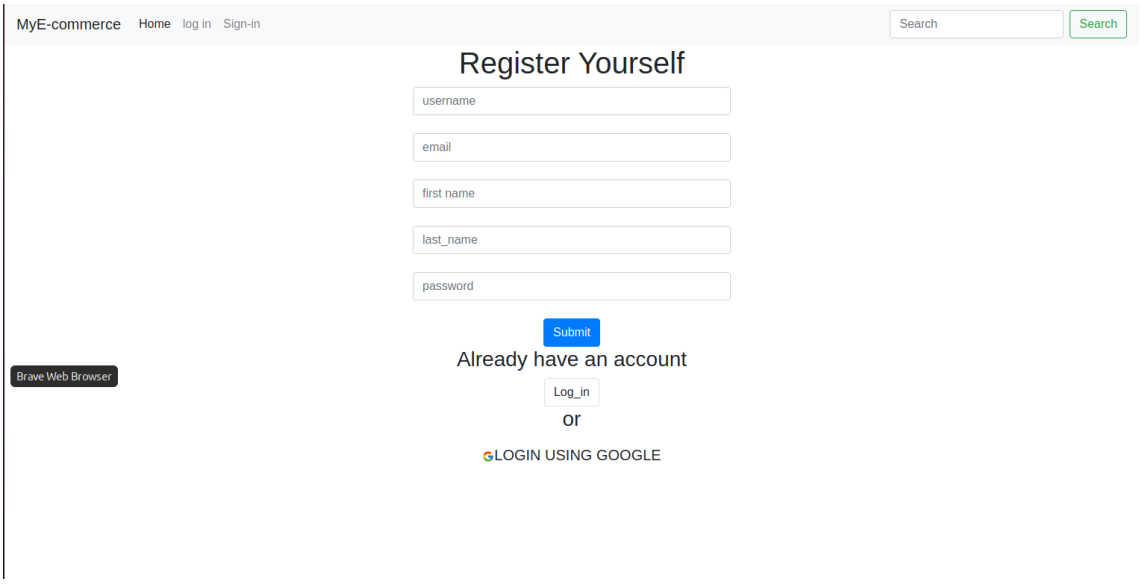


Fig: 3.3. User Registration page

Or simply log in with Google:-

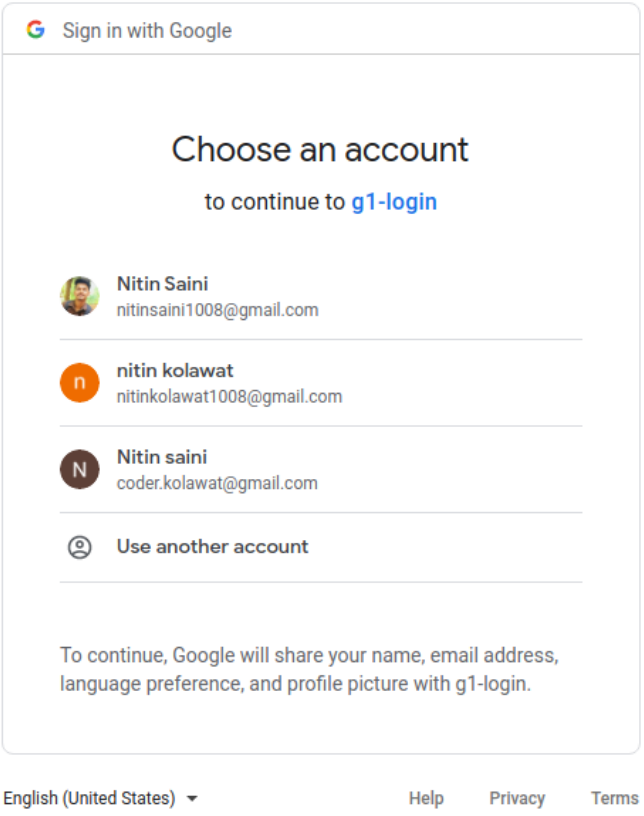


Fig: 3.4. Login with google

3.3. Add items to cart And Place an order:-

After login users can add items to their cart.

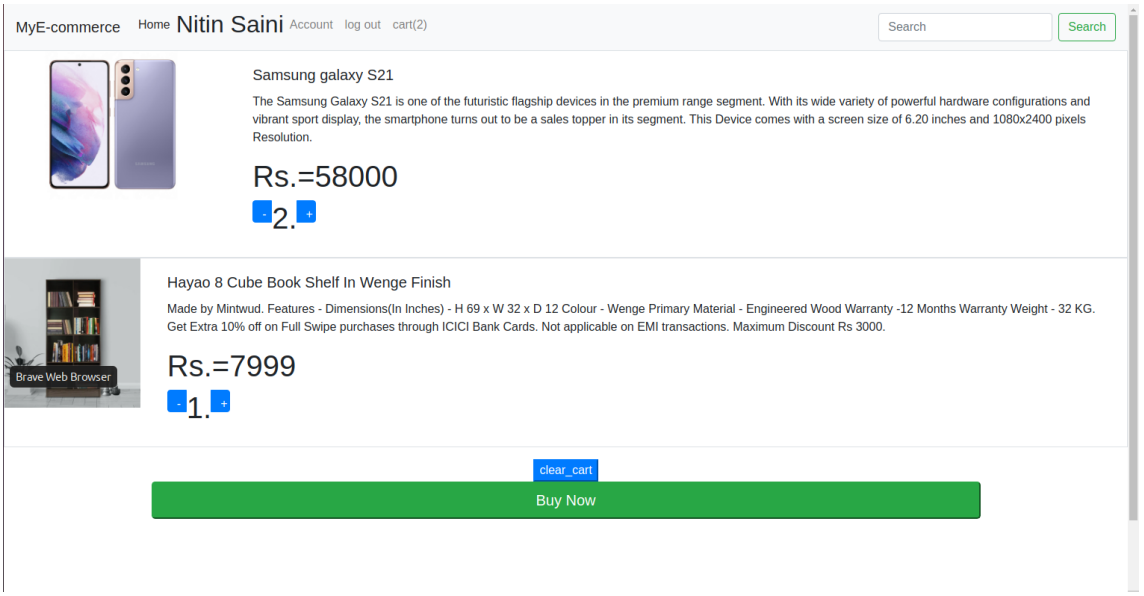


Fig: 3.5. User Cart view

After adding items to their cart they can buy those items after entering some simple details. And can also choose the payment option i.e. Cash on delivery or pay online.

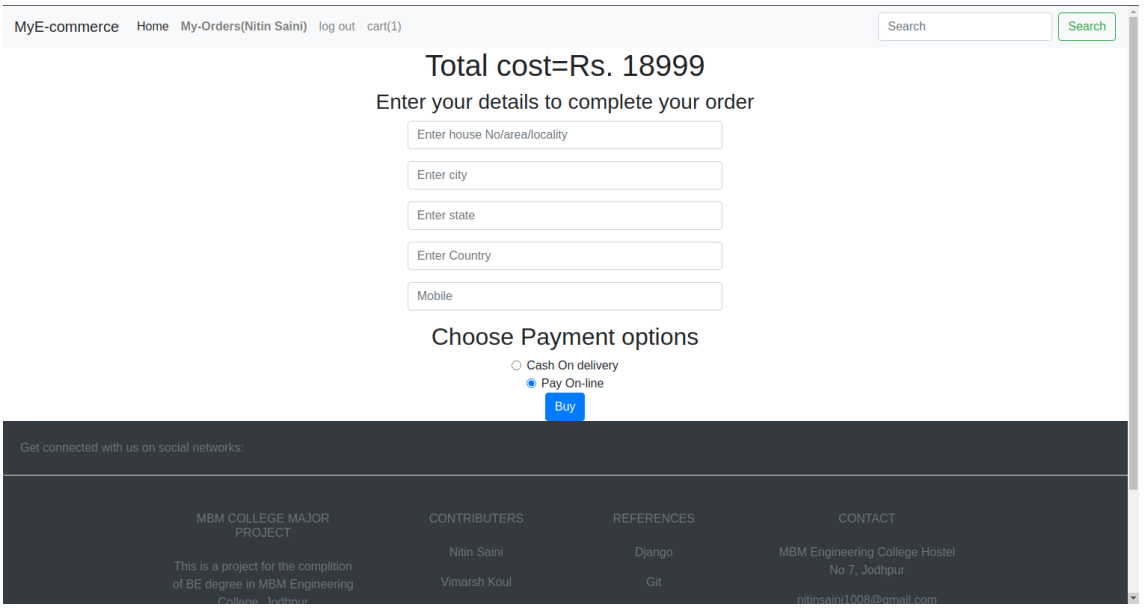


Fig: 3.6. Buy items

If users choose to Pay online then they will redirect to the Paytm payment gateway where they can make their payment.

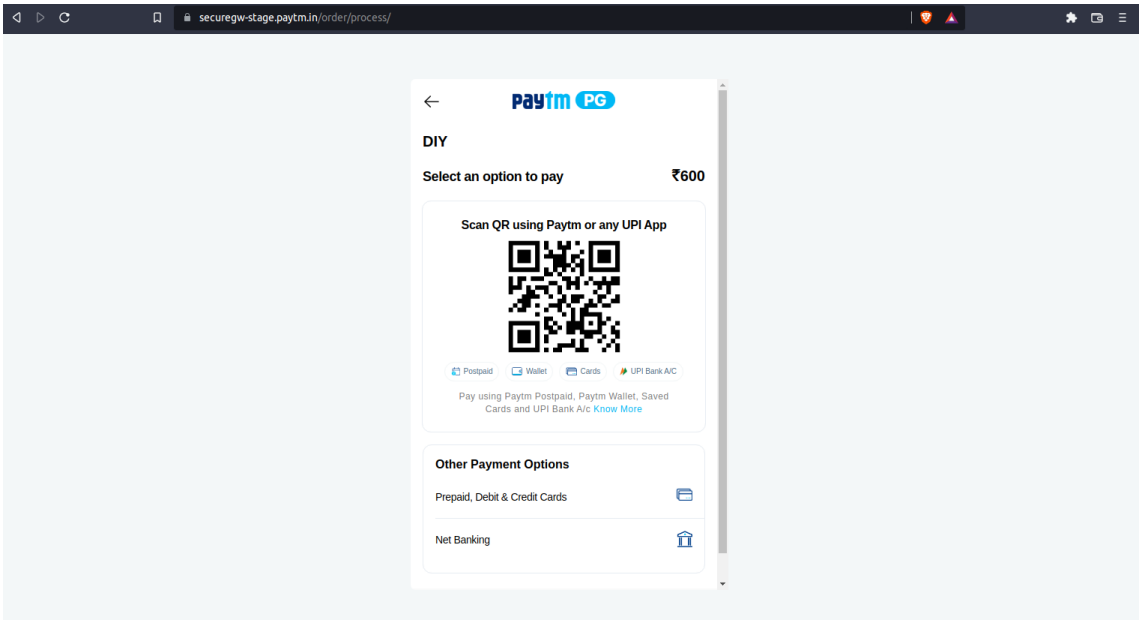


Fig: 3.7. Pay online using Paytm.

This page was intentionally left blank

Chapter 4: Conclusion & Future Work

4.1 Conclusion

We finally created an E-Commerce web application in which users can shop online easily. In our project users first register or just directly log in with google and our app will fetch some details of the user like name, email, etc. once the user is logged in then they can search for a product in the app, and then they can add that product into their cart and further they can also buy their product added into cart.

They can also see their purchase history. And they can also write their reviews on a product. Using our app users don't need to go anywhere for shopping they can order anything from their phone/laptop.

4.2 Future Enhancements

This is the first version of our app but still it required some features and some enhancements like-

Creating the Rest APIs:- Will create the rest APIs so that we can use those APIs for the web apps as well as in mobile apps.

React js with Frontend:- Integrating react with the frontend for a better UI/UX experience.

ML Model for Product Recommendation:- We can include a product recommendation feature for more user retention ability.

This page was intentionally left blank.

References

[1] Git official documentation: Written and maintained by Linus Torvalds and team, https://git-scm.com/docs, accessed by 14th June 2022.

[2] Paytm Payment Gateway Integration in Django: By Ashutosh Chauhan for IOTA, IIITS, https://dev.to/iiits-iota/paytm-payment-gateway-integration-in-django-1657, accessed by 7th July 2022.

[3] OAuth 2.0 for Client-side Web Applications: By Google community, https://developers.google.com/identity/protocols/oauth2/javascript-implicit-flow, accessed by 14th June 2022.

[4] Django Official documentation: Written and maintained By Django Community, https://www.djangoproject.com/, accessed 14th June 2022.

[5] Ecommerce Database Design: Written by Shanika Wickramasinghe, https://fabric.inc/blog/ecommerce-database-design-example/, accessed by 14th June 2022.

[6] Test credentials for testing Paytm: Written on January 23, 2017, by MVC group, https://mvcjaipur.wordpress.com/2017/01/23/test-credentials-for-testing-paytm/, accessed by 7th July 2022.

[7] Paytm for API only: written by Adyen community, https://docs.adyen.com/payment-methods/paytm/api-only, accessed by 7th July 2022.