Mini Project

Report

on

Postman Clone

By

LAVANYA & EKTA RAI
(2001330100149 & 2001330100108)

Under the Supervision of

MR AJEET KUMAR
ASSISTANT PROF

Submitted to the department of Computer Science and Engineering

For the partial fulfillment of the requirements for award of Bachelor of Technology

in

Computer Science and Engineering

Noida Institute of Engineering & Technology Gr. Noida
Dr. A.P.J. Abdul Kalam Technical University, Lucknow, Uttar Pradesh, India
May, 2023-2024

Certificate

This is to certify that the Project report entitled **"POSTMAN CLONE"** is a record of the work done by the following students:

LAVANYA & EKTA RAI
Roll No.2001330100149 & 2001330100108

This work is done under my/our supervision and guidance during the academic year of 2023-24. This report is submitted to the **Noida Institute of Engineering & Technology, Greater Noida** for partial fulfillment for the degree of **B.TECH. (Computer Science and Engineering)** of **Dr A P J Abdul Kalam Technical University, Lucknow, Uttar Pradesh, India.**

I/We wish him all the best for all the endeavors.

Signature of Guide:
MR AJEET KUMAR
ASSISTANT PROF.

ACKNOWLEDGEMENT

I would like to place on record my deep sense of gratitude to **MR. AJEET KUMAR, ASSITANT PROF. Department of Computer Science and Engineering, Noida Institute of Engineering & Technology**, Greater Noida, Gautam Budha Nagar, Uttar Pradesh, India for his generous guidance, help and useful suggestions.

I express my sincere gratitude to **Prof. Kumud Saxena, HOD(CSE)**, Noida Institute of Engineering & Technology, Greater Noida for his stimulating guidance, continuous encouragement and supervision throughout the course of present work.

Date: 05-05-2023

LAVANYA &
EKTA RAI

TABLE OF CONTENTS

ABSTRACT

Postman is an API development and testing tool that simplifies the process of working with APIs. It provides a user-friendly interface that allows developers to create, test, and document APIs quickly and easily.

At a high level, Postman abstracts away the complexities of working with APIs, allowing developers to focus on the core functionality of their applications. It provides a range of features and tools, such as the ability to send HTTP requests, analyze responses, and create automated tests.

In addition, Postman abstracts away many of the details of API documentation, allowing developers to easily create and share documentation for their APIs. With Postman, developers can create and share collections of API requests, test results, and documentation with other team members, and collaborate in real-time on the development of their APIs.

Overall, Postman abstracts away many of the complexities of working with APIs, providing developers with a powerful toolset that simplifies the development and testing of their applications.

CHAPTER 1

INTRODUCTION

This chapter gives an overview about the aim , objectives ,background and operation environment of the system.

1.1 PROJECT AIMS AND OBJECTIVES

The project aims and objectives that will be achieved after completion of this project are discussed in this subchapter. The aims and objectives are as follows:

- To simplify the process of API development and testing for developers.
- To provide a user-friendly interface that makes it easy to create, test, and document APIs.
- To streamline the process of collaborating on API development projects.
- To integrate with a variety of other tools to support the development workflow.
- To improve the overall efficiency and effectiveness of API development.
- To provide a range of tools and features for working with APIs, including the ability to send HTTP requests, analyze responses, and create automated tests.
- To abstract away many of the complexities of working with APIs, allowing developers to focus on the core functionality of their applications.
- To support team collaboration by providing features for sharing API requests, test results, and documentation.
- To integrate with other tools such as Newman, Jenkins, and GitHub to provide a seamless workflow.
- To provide high-quality documentation and support to help developers get the most out of the tool.

1.2 BACKGROUND OF PROJECT

Postman Clone is a project inspired by the popular API development and testing tool, Postman. The aim of the project is to create a tool that offers similar functionality to Postman but with some added features and customization options.

The development of Postman Clone is based on the increasing demand for a lightweight and easy-to-use alternative to Postman. The creators of Postman Clone recognized that while Postman is a

powerful tool, it can be overwhelming for some developers due to its extensive feature set and complex interface.

To address this need, the creators of Postman Clone developed a tool that is designed to be more streamlined and customizable. They have focused on creating a user-friendly interface that allows developers to create and test APIs quickly and easily.

The development of Postman Clone is also driven by the desire to improve the process of API development and testing for developers. The tool offers features such as API documentation, team collaboration, and integration with other tools to help developers streamline their workflow and improve their productivity.

Overall, the background of the Postman Clone project is rooted in the desire to create a tool that simplifies the process of API development and testing while providing developers with the flexibility and customization options they need to get the job done.

1.3 OPERATION ENVIRONMENT

PROCESSOR	INTEL CORE PROCESSOR OR BETTER PERFORMANCE
OPERATING SYSTEM	WINDOWS VISTA ,WINDOWS7, UBUNTU
MEMORY	1GB RAM OR MORE
HARD DISK SPACE	MINIMUM 1 GB FOR DATABASE USAGE FOR FUTURE
LANGUAGE	HTML, CSS, JAVASCRIPT
IDE	VISUAL STUDIO CODE, ETC.
WEB APPLICATION	CHROME, FIREFOX, ETC.

CHAPTER 2

SYSTEM ANALYSIS

In this chapter, we will discuss the development process of a Postman tool clone, including the software requirement specification (SRS) and a comparison between the existing Postman tool and the proposed clone. The SRS will include both functional and non-functional requirements to provide a comprehensive description and overview of the system requirements before the development process begins.

2.1 SOFTWARE REQUIREMENT SPECIFICATION

2.1.1 GENERAL DESCRIPTION

PRODUCT DESCRIPTION:

Postman Clone is a computerized tool that helps developers to manage their API development and testing activity in an electronic format. It reduces the risk of errors, saves time, and provides a more efficient and effective solution for API development and testing.

PROBLEM STATEMENT:

The problems occurred before having a computerized tool like Postman Clone includes:

• Manual Testing

Before Postman Clone, developers had to perform manual testing of their APIs which was a time-consuming and error-prone process.

• Lack of Collaboration

Without a tool like Postman Clone, there was a lack of collaboration between team members, which caused delays in the development process and made it difficult to track changes.

• No Centralized Repository

There was no centralized repository for storing and managing API requests and responses, which made it difficult to track the history of changes made to APIs over time.

• Difficulty in Debugging

Developers faced difficulty in debugging APIs without a tool like Postman Clone which could help them to identify errors and bugs in their APIs.

• Limited Integration

Without a tool like Postman Clone, developers faced difficulty in integrating APIs with other tools and services, which made it difficult to create a seamless development environment.

Postman Clone addresses all these issues by providing a centralized repository for API requests and responses, streamlining the testing and debugging process, enabling team collaboration, and integrating with other tools and services. This results in a more efficient and effective solution for API development and testing that saves time, reduces errors, and improves overall productivity.

2.1.2 SYSTEM OBJECTIVES

- Improvement in control and performance
  The system is developed to cope up with the current issues and problems of library .The system can add user, validate user and is also bug free.
- Save cost
  After computerized system is implemented less human force will be required to maintain the library thus reducing the overall cost.
- Save time
  Librarian is able to search record by using few clicks of mouse and few search keywords thus saving his valuable time.
- Option of online Notice board
  Librarian will be able to provide a detailed description of workshops going in the college as well as in nearby colleges

- Lecture Notes
  Teacher have a facility to upload lectures notes in a pdf file having size not more than 10mb.

2.1.3 SYSTEM REQUIREMENTS

2.1.3.1 NON FUNCTIONAL REQUIREMENTS

 Product Requirements

Performance: The tool should have high performance and be able to handle a large number of requests without slowing down or crashing.

Reliability: The tool should be reliable and error-free, and any issues or bugs should be quickly addressed and fixed.

Security: The tool should have strong security measures in place to protect sensitive data and prevent unauthorized access.

Ease of use: The tool should have an intuitive and user-friendly interface, and be easy for users to navigate and perform tasks.

Compatibility: The tool should be compatible with a wide range of operating systems, web browsers, and APIs.

Scalability: The tool should be scalable and able to accommodate growing numbers of users and requests.

Availability: The tool should be available to users at all times, with minimal downtime or maintenance windows.

Documentation: The tool should have comprehensive and up-to-date documentation, including user manuals, guides, and FAQs, to help users understand and use its features effectively.

2.1.3.2 FUNCTIONAL REQUIREMENTS

API Requests: The tool should allow users to create, send and manage RESTful API requests to test API functionality.

Authentication: Postman should allow users to specify and test authentication methods such as OAuth, basic authentication, or API key authentication.

Environments: Postman should allow users to set up different environments (e.g. dev, staging, production) for API testing and switch between them easily.

Tests and Assertions: The tool should allow users to write and run automated tests for API responses and validate them using assertions.

Collections: Postman should allow users to group API requests into collections and organize them for easy management and sharing.

Mock Servers: Postman should allow users to create and run mock servers to simulate API responses and test API integrations.

Collaboration: Postman should allow users to share API requests, collections and test results with team members and collaborate on API development and testing.

Documentation: Postman should allow users to generate API documentation from API requests and collections for easy sharing and reference.

2.1.4 SOFTWARE AND HARDWARE REQUIREMENTS

This section describes the software and hardware requirements of the system

2.1.4.1 SOFTWARE REQUIREMENTS

- Operating System: Windows 7 or later, macOS 10.11 or later, or Linux
- RAM: Minimum 4 GB
- Disk Space: Minimum 400 MB free space for installation
- Browser: Google Chrome or any other modern browser

2.1.4.2 HARDWARE REQUIREMENTS

- Processor: Minimum Intel Pentium 4 or AMD Athlon 64 processor or higher
- Display: Minimum resolution of 1280x800 or higher
- Internet Connectivity: Required for accessing APIs and sending/receiving requests and responses through Postman tool.

2.2 SOFTWARE TOOLS USED

The software tools used for creating a Postman tool clone would depend on the specific technology stack and programming languages used. However, some common software tools and frameworks used for building API testing and development tools like Postman include:

Programming languages: Depending on the requirements, the tool can be built using programming languages like JavaScript, Python, Java, or C#.

Frameworks: Popular frameworks like Node.js, Django, Flask, and Express.js can be used to build the backend of the application.

Database: A database management system like MySQL, PostgreSQL, or MongoDB can be used to store and manage the API requests and responses.

API testing libraries: Libraries like Jest, Mocha, and Chai can be used to write and run tests for the API.

UI Frameworks: For creating the user interface, frameworks like React, Angular, or Vue.js can be used.

Version control: Git and GitHub can be used for version control and collaboration.

Integrated Development Environment (IDE): An IDE like Visual Studio Code, PyCharm, or WebStorm can be used for coding, debugging, and testing.

API Documentation: Swagger or OpenAPI can be used for API documentation.

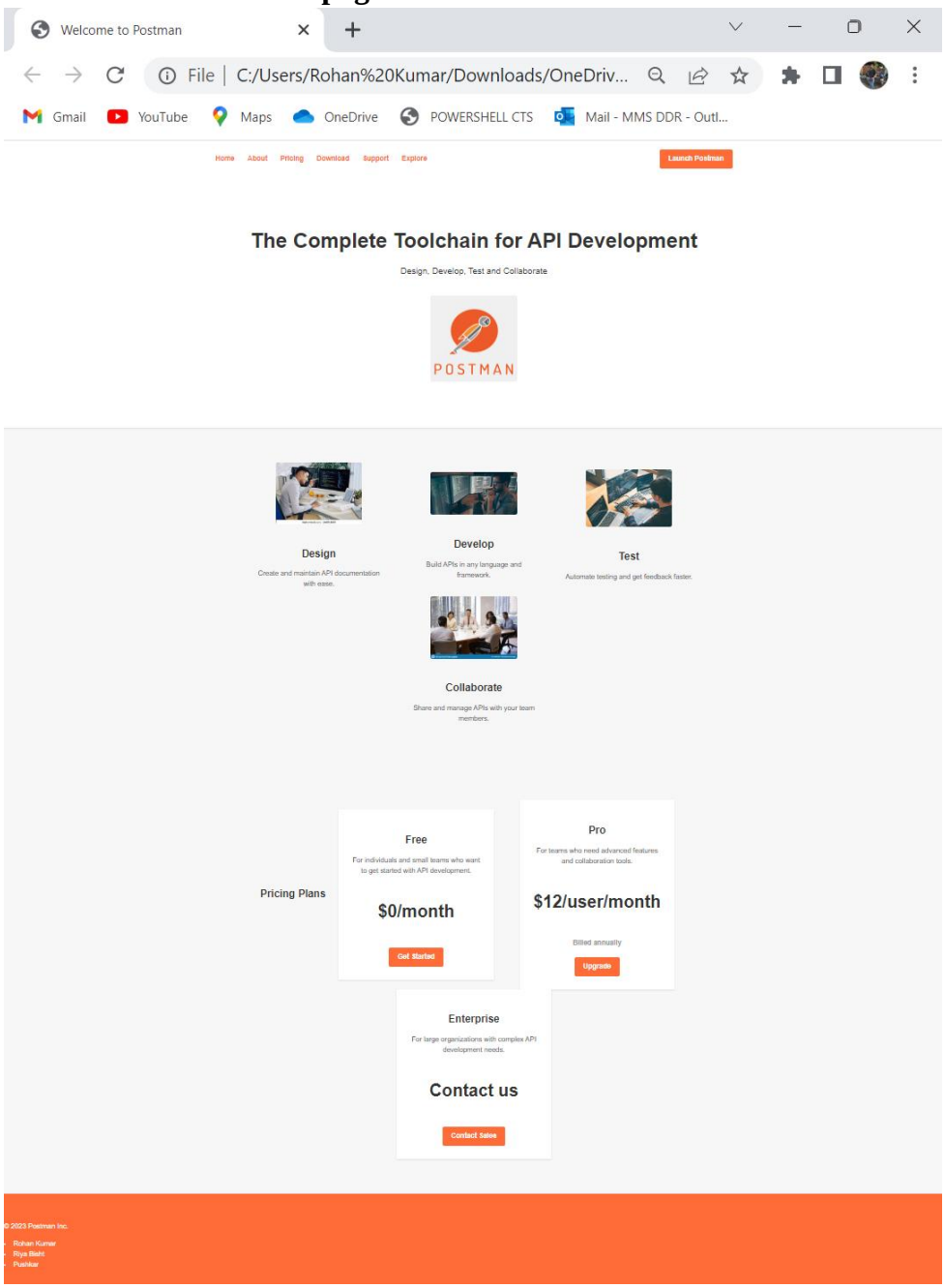
Cloud Hosting: AWS, Azure, or Google Cloud Platform can be used for hosting the application on the cloud.

Postman API: Developers can also use the Postman API to integrate their application with Postman collections and environments.

CHAPTER -3

SCREENSHOTS

3.1 Screenshot for homepage



3.1.1 SCREENSHOT OF CODE

Home.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Welcome to Postman</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <header>
        <nav>
            <ul>
                <li><a href="home.html">Home</a></li>
                <li><a href="https://www.postman.com/company/about-postman/">About</a></li>
                <li><a href="https://www.postman.com/pricing/">Pricing</a></li>
                <li><a href="https://www.postman.com/downloads/">Download</a></li>
                <li><a href="https://www.postman.com/company/contact-us/">Support</a></li>
                <li><a href="https://www.postman.com/explore?utm_source=postman-
website&utm_medium=referral">Explore</a></li>
            </ul>
            <a href="index.html" class="button">Launch Postman</a>
        </nav>
        <div class="header-content">
            <h1>The Complete Toolchain for API Development</h1>
            <p>Design, Develop, Test and Collaborate</p>
            <img src="postman.png" alt="Postman Logo">
        </div>
    </header>
    <main>
        <section class="features">
            <div class="feature">
                <img src="design.jpg" alt="Design">
                <h3>Design</h3>
                <p>Create and maintain API documentation with ease.</p>
            </div>
            <div class="feature">
                <img src="develop.jpg" alt="Develop">
                <h3>Develop</h3>
                <p>Build APIs in any language and framework.</p>
            </div>
            <div class="feature">
                <img src="test.jpg" alt="Test">
```

```
                <h3>Test</h3>
                <p>Automate testing and get feedback faster.</p>
            </div>
                <div class="feature">
                <img src="collabrate.jpg" alt="Collaborate">
                <h3>Collaborate</h3>
                <p>Share and manage APIs with your team members.</p>
            </div>
        </section>
        <section class="pricing">
            <h2>Pricing Plans</h2>
            <div class="plan">
                <h3>Free</h3>
                <p>For individuals and small teams who want to get started with API
development.</p>
                <div class="price">
                    <h4>$0/month</h4>
                </div>
                <a href="#" class="button">Get Started</a>
            </div>
            <div class="plan">
                <h3>Pro</h3>
                <p>For teams who need advanced features and collaboration tools.</p>
                <div class="price">
                    <h4>$12/user/month</h4>
                    <p>Billed annually</p>
                </div>
                <a href="#" class="button">Upgrade</a>
            </div>
            <div class="plan">
                <h3>Enterprise</h3>
                <p>For large organizations with complex API development needs.</p>
                <div class="price">
                    <h4>Contact us</h4>
                </div>
                <a href="#" class="button">Contact Sales</a>
            </div>
        </section>
    </main>
    <footer>
        <p>&copy; 2023 Postman Inc.</p>
        <li> Rohan Kumar</li>
        <li> Riya Bisht</li>
        <li> Pushkar</li>
    </footer>
```

```
</body>
</html>
```

Style.css

```
/* CSS for Postman Landing Page */

/* Global Styles */
body {
    font-family: 'Helvetica Neue', sans-serif;
    font-size: 16px;
    line-height: 1.5;
    color: #333;
    background-color: #f7f7f7;
    margin: 0;
    padding: 0;
}
img{
    border-radius: 5px;
}

a {
    color: #ff6c37;
    text-decoration: none;
}

a:hover {
    text-decoration: underline;
}

button, .button {
    display: inline-block;
    padding: 10px 20px;
    background-color: #ff6c37;
    color: #fff;
    border: none;
    border-radius: 4px;
    font-size: 16px;
    font-weight: bold;
    cursor: pointer;
    transition: background-color 0.3s ease;
}
```

```css
button:hover, .button:hover {
    background-color: #005fac;
}

/* Header */
header {
    background-color: #fff;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    position: relative;
    z-index: 1;
}

nav {
    max-width: 1200px;
    margin: 0 auto;
    padding: 20px 0;
    display: flex;
    justify-content: space-between;
    align-items: center;
}

nav ul {
    display: flex;
    list-style: none;
    margin: 0;
    padding: 0;
}

nav li {
    margin-left: 30px;
}

nav li:first-child {
    margin-left: 0;
}

nav a {
    font-weight: bold;
}

.header-content {
    max-width: 1200px;
    margin: 0 auto;
    padding: 80px 0;
    text-align: center;
```

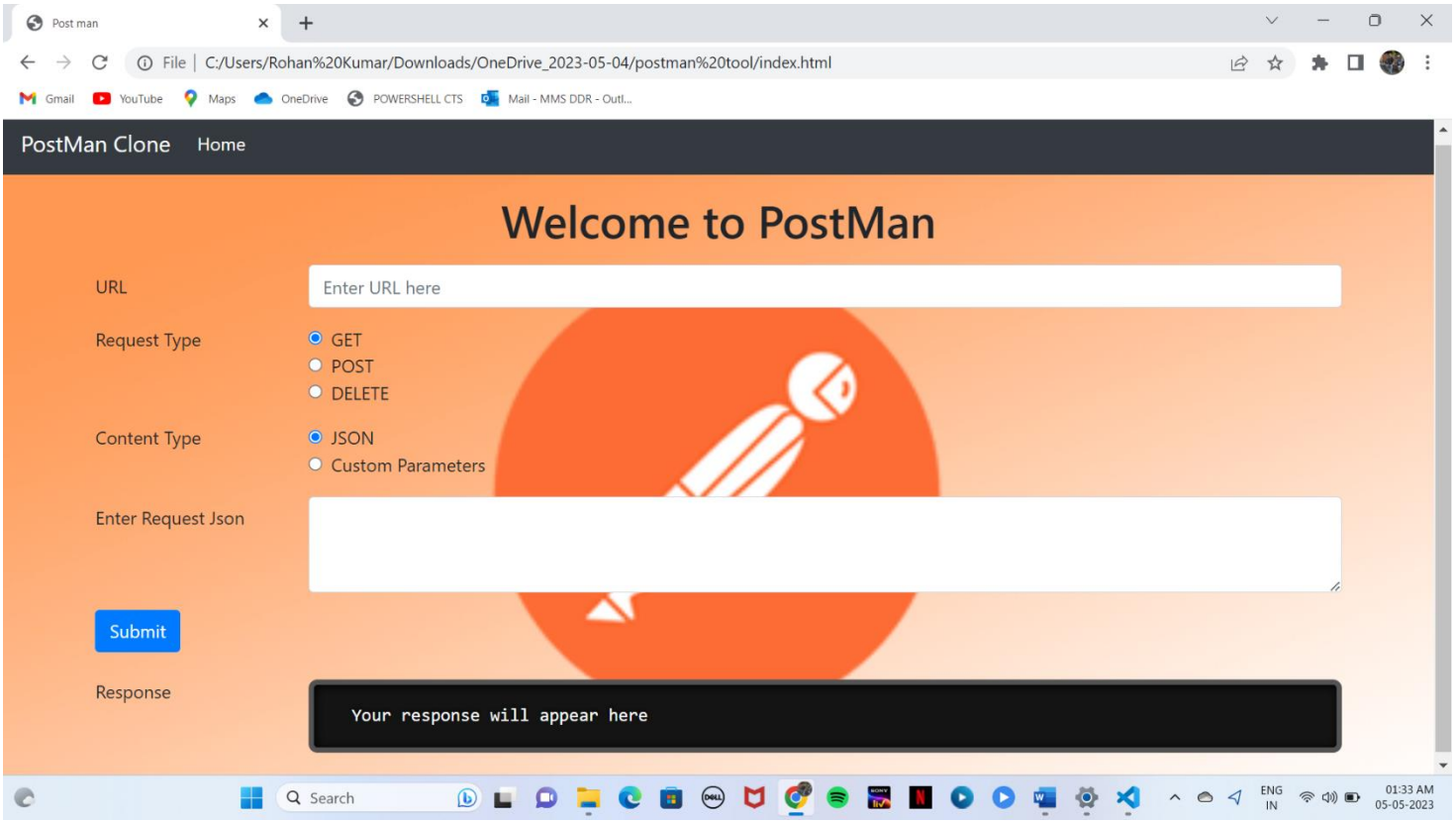
```
}

.header-content h1 {
    font-size: 48px;
    font-weight: bold;
    margin-bottom: 20px;
}

.header-content p {
    font-size: 20px;
    margin-bottom: 40px;
}

.header-content img {
    max-width: 400px;
    margin-bottom: 20px;
}

/* Features */
.features {
    max-width: 1200px;
    margin: 0 auto;
    padding: 80px 0;
    display: flex;
    justify-content: center;
    align-items: center;
    flex-wrap: wrap;
}

.feature {
    width: 300px;
    margin: 0 30px;
    text-align: center;
}

.feature img {
    max-width: 200px;
    margin-bottom: 20px;
}

.feature h3 {
    font-size: 24px;
    font-weight: bold;
    margin-bottom: 10px;
}
```

```
.feature p {
    font-size: 16px;
}

/* Pricing */
.pricing {
    max-width: 1200px;
    margin: 0 auto;
    padding: 80px 0;
    display: flex;
    justify-content: center;
    align-items: center;
    flex-wrap: wrap;
}

.plan {
    width: 300px;
    margin: 0 30px;
    background-color: #fff;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    padding: 30px;
    text-align: center;
}

.plan h3 {
    font-size: 24px;
    font-weight: bold;
    margin-bottom: 10px;
}

.plan p {
    font-size: 16px;
    margin-bottom: 20px;
}

.price {
    font-size: 40px;
    font-weight: bold;
    margin-bottom: 20px;
}

.price p {
    font-size: 16px;
    color: #999;
```

```
}

/* Footer */
footer {
    background-color: #ff6c37;
    color: #fff;
    padding: 50px 0;
    }

    .footer-content {
    max-width: 1200px;
    margin: 0 auto;
    display: flex;
    flex-wrap: wrap;
    justify-content: space-between;
    align-items: center;
    }

    .footer-col {
    flex-basis: 30%;
    }

    .footer-col h3 {
    font-size: 24px;
    font-weight: bold;
    margin-bottom: 20px;
    }

    .footer-col ul {
    list-style: none;
    margin: 0;
    padding: 0;
    }

    .footer-col li {
    margin-bottom: 10px;
    }

    .footer-col a {
    color: #fff;
    }

    .footer-col a:hover {
    text-decoration: underline;
    }
```

```
/* Media Queries */
@media (max-width: 768px) {
.header-content h1 {
font-size: 36px;
}
.header-content p {
    font-size: 18px;
}

.features {
    flex-direction: column;
}

.feature {
    margin: 20px 0;
}

.plan {
    margin: 20px 0;
}

.footer-content {
    flex-direction: column;
}

.footer-col {
    flex-basis: 100%;
    margin-bottom: 30px;
}}
```

3.2 Screenshot of postman clone page



3.2.1 SCREENSHOT OF CODE

Index.html

```
<!doctype html>
<html lang="en">

<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <link rel="stylesheet" href="prism.css">
    <!-- Bootstrap CSS -->
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
        integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">

    <style>
```

```
        #responsePre {
            max-height: 500px;
        }

        /* body {

        } */

    </style>
    <title>Post man</title>
</head>

<body style="background-image: url('background.png'); background-size: cover;
background-position: center;
background-attachment: fixed;">
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <a class="navbar-brand" href="#" style="text-align: center;">PostMan Clone</a>
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarText"
            aria-controls="navbarText" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarText">
            <ul class="navbar-nav mr-auto">
                <li class="nav-item active">
                    <a class="nav-link" href="home.html">Home <span class="sr-
only">(current)</span></a>
                </li>

            </ul>

        </div>
    </nav>

    <div class="container">
        <h1 class="my-3" style="text-align: center;">Welcome to PostMan</h1>
    </div>

    <div class="container">

        <!-- URL box -->
        <div class="form-group row">
            <label for="url" class="col-sm-2 col-form-label">URL</label>
            <div class="col-sm-10">
                <input type="text" class="form-control" id="url" placeholder="Enter URL here">
```

```
            </div>
        </div>

        <!-- Request type box -->
        <fieldset class="form-group">
            <div class="row">
                <legend class="col-form-label col-sm-2 pt-0">Request Type</legend>
                <div class="col-sm-10">
                    <div class="form-check">
                        <input class="form-check-input" type="radio" name="requestType"
id="get" value="GET" checked>
                        <label class="form-check-label" for="get">
                            GET
                        </label>
                    </div>
                    <div class="form-check">
                        <input class="form-check-input" type="radio" name="requestType"
id="post" value="POST">
                        <label class="form-check-label" for="post">
                            POST
                        </label>
                    </div>
                    <div class="form-check">
                        <input class="form-check-input" type="radio" name="requestType"
id="delete" value="DELETE">
                        <label class="form-check-label" for="delete">
                            DELETE
                        </label>
                    </div>
            </div>
        </fieldset>

        <!-- Content type box -->
        <fieldset class="form-group">
            <div class="row">
                <legend class="col-form-label col-sm-2 pt-0">Content Type</legend>
                <div class="col-sm-10">
                    <div class="form-check">
                        <input class="form-check-input" type="radio" name="contentType"
id="jsonRadio" value="json"
                            checked>
                        <label class="form-check-label" for="json">
                            JSON
                        </label>
                    </div>
```

```
                    <div class="form-check">
                        <input class="form-check-input" type="radio" name="contentType"
id="paramsRadio" value="params">
                        <label class="form-check-label" for="params">
                            Custom Parameters
                        </label>
                    </div>

                </div>
            </div>
        </fieldset>

        <!-- Parameters box - This will hide on clicking json option in content type  -->
        <div id="parametersBox">
            <div class="form-row">
                <label for="url" class="col-sm-2 col-form-label">Parameter 1</label>
                <div class="col-md-4">
                    <input type="text" class="form-control" id="parameterKey1"
placeholder="Enter Parameter 1 Key">
                </div>
                <div class="col-md-4">
                    <input type="text" class="form-control" id="parameterValue1"
placeholder="Enter Parameter 1 Value">
                </div>
                <button id="addParam" class="btn btn-primary">+</button>
            </div>
            <div id="params"></div>
        </div>

        <!-- Json Request box - This will hide on clicking parameters option in content type --
>
        <div class="my-3" id="requestJsonBox">
            <div class="form-group row">
                <label for="requestJsonText" class="col-sm-2 col-form-label">Enter Request
Json</label>
                <div class="col-sm-10">
                    <textarea class="form-control" id="requestJsonText" rows="3"></textarea>
                </div>
            </div>
        </div>

        <!-- Submit button which will trigger fetch api -->
        <div class="form-group row my-2">
            <div class="col-sm-10">
                <button id="submit" class="btn btn-primary">Submit</button>
```

```
            </div>
        </div>


        <div class="my-3" id="responseJsonBox">
            <div class="form-group row">
                <label for="responseJsonText" class="col-sm-2 col-form-label">Response</label>
                <div class="col-sm-10">
                    <!-- <textarea class="form-control" id="responseJsonText" rows="3">Your
response will appear here</textarea> -->

                    <pre id='responsePre'
                        class="language-javascript"> <code id='responsePrism' class="language-
javascript"> Your response will appear here </code> </pre>
                </div>
            </div>
        </div>

    </div>



    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
        integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
        crossorigin="anonymous"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
        integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
        crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
        integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
        crossorigin="anonymous"></script>

    <script src="index.js"></script>
    <script src="prism.js"></script>
</body>

</html>
```

Index.js

```
// Utility functions:
// 1. Utility function to get DOM element from string
```

```
function getElementFromString(string) {
    let div = document.createElement('div');
    div.innerHTML = string;
    return div.firstElementChild;
}

// Initialize no of parameters
let addedParamCount = 0;

// Hide the parameters box initially
let parametersBox = document.getElementById('parametersBox');
parametersBox.style.display = 'none';

// If the user clicks on params box, hide the json box
let paramsRadio = document.getElementById('paramsRadio');
paramsRadio.addEventListener('click', () => {
    document.getElementById('requestJsonBox').style.display = 'none';
    document.getElementById('parametersBox').style.display = 'block';
})

// If the user clicks on json box, hide the params box
let jsonRadio = document.getElementById('jsonRadio');
jsonRadio.addEventListener('click', () => {
    document.getElementById('requestJsonBox').style.display = 'block';
    document.getElementById('parametersBox').style.display = 'none';
})

// If the user clicks on + button, add more parameters
let addParam = document.getElementById('addParam');
addParam.addEventListener('click', () => {
    let params = document.getElementById('params');
    let string = `<div class="form-row my-2">
                    <label for="url" class="col-sm-2 col-form-label">Parameter
${addedParamCount + 2}</label>
                    <div class="col-md-4">
                        <input type="text" class="form-control"
id="parameterKey${addedParamCount + 2}" placeholder="Enter Parameter ${addedParamCount + 2}
Key">
                    </div>
                    <div class="col-md-4">
                        <input type="text" class="form-control"
id="parameterValue${addedParamCount + 2}" placeholder="Enter Parameter ${addedParamCount + 2}
Value">
                    </div>
                    <button class="btn btn-primary deleteParam"> - </button>
```

```
                    </div>`;
    // Convert the element string to DOM node
    let paramElement = getElementFromString(string);
    params.appendChild(paramElement);
    // Add an event listener to remove the parameter on clicking - button
    let deleteParam = document.getElementsByClassName('deleteParam');
    for (item of deleteParam) {
        item.addEventListener('click', (e) => {
            // TODO: add a confirmation box to confirm parameter deletion
            e.target.parentElement.remove();
        })
    }
    addedParamCount++;
})

// If the user clicks on submit button
let submit = document.getElementById('submit');
submit.addEventListener('click', () => {
    // Show please wait in the response box to request patience from the user
    // document.getElementById('responseJsonText').value = "Please wait.. Fetching
response...";
    document.getElementById('responsePrism').innerHTML = "Please wait.. Fetching response...";

    // Fetch all the values user has entered
    let url = document.getElementById("url").value;
    let requestType = document.querySelector("input[name='requestType']:checked").value;
    let contentType = document.querySelector("input[name='contentType']:checked").value;

    // If user has used params option instead of json, collect all the parameters in an object
    if (contentType == 'params') {
        data = {};
        for (let i = 0; i < addedParamCount + 1; i++) {
            if (document.getElementById('parameterKey' + (i + 1)) != undefined) {
                let key = document.getElementById('parameterKey' + (i + 1)).value;
                let value = document.getElementById('parameterValue' + (i + 1)).value;
                data[key] = value;
            }
        }
        data = JSON.stringify(data);
    }
    else {
        data = document.getElementById('requestJsonText').value;
    }

    // Log all the values in the console for debugging
```

```
    console.log('URL is ', url);
    console.log('requestType is ', requestType);
    console.log('contentType is ', contentType);
    console.log('data is ', data);

    // if the request type is get, invoke fetch api to create a post request

if (requestType === 'DELETE') {
    fetch(url, {
        method: 'DELETE',
        headers: {
            'Content-Type': 'application/' + contentType
        }
    })
    .then(response => response.text())
    .then((data) => {
        document.getElementById('responsePrism').innerHTML = data;
        Prism.highlightAll();
    });
}
    else if (requestType=='GET'){
        fetch(url, {
            method: 'GET',
        })
        .then(response=> response.text())
        .then((text) =>{
            // document.getElementById('responseJsonText').value = text;
            document.getElementById('responsePrism').innerHTML = text;
            Prism.highlightAll();
        });
    }

    else {
        fetch(url, {
            method: 'POST',
            body: data,
            headers: {
                "Content-type": "application/json; charset=UTF-8"
              }
        })

        .then(response=> response.text())
        .then((text) =>{
            // document.getElementById('responseJsonText').value = text;
            document.getElementById('responsePrism').innerHTML = text;
```

```
            Prism.highlightAll();
        });

    }



});
```

CHAPTER 5

CONCLUSION & FUTURE SCOPE

Conclusion:

In conclusion, Postman tool is a widely used API development platform that simplifies the process of testing and managing APIs. It provides a user-friendly interface for designing, testing, documenting, and monitoring APIs. Postman tool has become an essential tool for developers, QA engineers, and API consumers in today's fast-paced development environment.

Future Scope:

The future of Postman tool looks promising, with its continuous updates and improvements to meet the evolving needs of the developer community. Some of the areas that Postman tool can focus on in the future are:

Integration with more tools and services: Postman tool can integrate with more tools and services to provide a seamless experience for developers.

Improving collaboration features: Postman tool can enhance its collaboration features to make it easier for teams to work together on API development.

Enhancing security features: With the increasing focus on API security, Postman tool can provide more robust security features to help developers secure their APIs.

More automation and scripting capabilities: Postman tool can provide more automation and scripting capabilities to help developers automate their API testing and monitoring processes.

Overall, the future of Postman tool looks bright, and it will continue to play a significant role in API development and testing.

CHAPTER 6

BIBLIOGRAPHY

- www.google.com
- W3school.com
- www.wikipedia.com
- www.tutorialspoint.com