FULL STACK PROJECT REPORT

**On**

**“Rhythm And Beats: Music Streaming Website”**

**Submitted by**

**Devansh K Shukla**

**(191500255)**

**Krishnaveer Chahar**

**(191500410)**

**Hemant Mudgal**

**(191500339)**

**Yug Mittal**

**(191500945)**

Department of Computer Engineering & Applications

**Institute of Engineering & Technology**



**GLA University**

**Mathura- 281406, INDIA**

**2020-2021**

**Department of computer Engineering and Applications**

**GLA University, Mathura**

**17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,**

**Mathura – 281406**



**Declaration**

We hereby declare that the work which is being presented in the Full Stack Project “**Rhythm And Beats: Music Streaming Website”,** in partial fulfillment of the requirements for Full Stack Project viva voce, is an authentic record of our own work carried by the team members under the supervision of our mentor Mr. Pankaj Kapoor.

Group Members: Devansh K Shukla (191500255)

Krishnaveer Chahar (191500410)

Hemant Mudgal (191500339)

Yug Mittal (191500945)

Course: B.Tech (Computer Science and Engineering)

Year: 3rd

Semester: 5th

## Supervised By:

Mr. Pankaj Kapoor, Assistant Professor,

GLA University, Department of Computer Engineering & Application

**Department of computer Engineering and Applications**

**GLA University, Mathura**

**17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,**

**Mathura – 281406**



**Certificate**

This is to certify that the above statements made by the candidates are correct to the best of my/our knowledge and belief.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Supervisor

Mr. Pankaj Kapoor

Technical Trainer

Dept of CEA, GLA University

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Project Coordinator Program Coordinator

(Mr. Mayank Srivastava) (Mr. Shashi Shekar)

**About the Project**

The objective of this project is to design a Music store web application with user interface which will enable them to browse, search, and get song recommendations of their choice. The word “design” in the context of a Web Application can mean many things. Its most popular usage probably refers to the visual and user interface (UI) design of a website. This aspect is crucial because, the visitor is often more impressed with how a website looks and how easy it is to use than about which technologies and techniques are used behind the scenes, or what operating system the web server is running. If the site is hard to use and easy to forget, it just doesn’t matter what technologies was used to create it. Unfortunately, this truth makes many inexperienced programmers underestimate the importance of the way the invisible part of the site is implemented—the code, the database, and so on. The visual part of a site gets visitors interested to begin with, but its functionality makes them come back. A web site can sometimes be implemented very quickly based on certain initial requirements, but if not properly architected, it can become difficult, if not impossible, to change.

**Motivation**

We built a music streaming web Application which is responsive and can be used both mobile and desktop. User can stream new songs and listen to them whenever and wherever they want. The user can add songs to their playlist and the streaming songs can be downloaded to their local machine, also we have the search option so that user can browse his own song, album, artist or composer the individual wants to listen to. We have used Spotify API to fetch the songs to our Application. The reason we choose to work upon this project is because music has become part of our day to day life. Music helps to refresh our mind, this is one of the reasons which motivated us to built this application. If we see in the developer point of view by building this application it had helped us to cover all the technologies we have learned so far and mainly it consists all CRUD operations to work on, so we choose to work upon this project.

**Requirements**

**a). Software Requirements:**

* Technology Implemented: Full Stack Web Development
* Languages/Technologies Used: MERN Stack
* IDE Used: Visual Studio Code
* Web Browser: Google Chrome
* GitHub: GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere. GitHub Repository: A GitHub repository can be used to store a development project. It can contain folders and any type of files (HTML, CSS, JavaScript, Documents, Data, Images). A GitHub repository should also include a license file and a README file about the project. A GitHub repository can also be used to store ideas, or any resources that you want to share.
* Visual Studio Code: Visual Studio Code is a free source-code editor made by Microsoft for Windows, Linux and macOS. [7] Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.

**b). Hardware Requirements:**

* Processor Required: Intel i5
* Operating System: Windows 10
* RAM: 8GB
* Hardware Devices: Computer System
* Hard Disk: 256GB

**Acknowledgement**

We thank the almighty for giving us the courage and perseverance in completing the project. This project itself is an acknowledgement for all those people who have given us their heartfelt co-operation in making this project a grand success. We extend our sincere thanks to Mr. Pankaj Kapoor, Assistant Professor at “GLA University, Mathura” for providing his valuable guidance at every stage of this project work. We are profoundly grateful towards the unmatched services rendered by him. And last but not least, we would like to express our deep sense of gratitude and earnest thanks giving to our dear parents for their moral support and heartfelt cooperation in doing the main project.

**Rhythm And Beats: Music Streaming Website**

**Abstract**

The objective of this project is to design a Music store web application with user interface which will enable them to browse, search, and get song recommendations of their choice. The word “design” in the context of a Web Application can mean many things. Its most popular usage probably refers to the visual and user interface (UI) design of a website. This aspect is crucial because, the visitor is often more impressed with how a website looks and how easy it is to use than about which technologies and techniques are used behind the scenes, or what operating system the web server is running. If the site is hard to use and easy to forget, it just doesn’t matter what technologies was used to create it. Unfortunately, this truth makes many inexperienced programmers underestimate the importance of the way the invisible part of the site is implemented—the code, the database, and so on. The visual part of a site gets visitors interested to begin with, but its functionality makes them come back. A web site can sometimes be implemented very quickly based on certain initial requirements, but if not properly architected, it can become difficult, if not impossible, to change.

**Contents**

**Acknowledgment…………………………………….........**08

**Abstract…………………………………………………**09

1. **Introduction:**

Introduction to MERN Stack........…......12-13

Pre-requisites…………………………..14

1. **Technologies Used:**

MongoDB……………………………….15

Express……………………......................16

ReactJS………………………………..............17-18

NodeJS...............................................................19

**3. List of Figures...……………………………………**20-26

**4. Software Testing..............................................................**27-31

**5.Conclusion……………………………………………**32

**6.Bibliography………………………………………….**33

**Chapter 1**

**Introduction**

Today Developers around the world are making efforts to enhance user experience of using application as well as to enhance the developer’s workflow of designing applications to deliver projects and rollout change requests under strict timeline. Stacks can be used to build web applications in the shortest span of time. The stacks used in web development are basically the response of software engineers to current demands. They have essentially adopted pre-existing frameworks (including JavaScript) to make their lives easier. While there are many, MEAN and MERN are just two of the popular stacks that have evolved out of JavaScript. Both stacks are made up of open-source components and offer an end-to-end framework for building comprehensive web apps that enable browsers to connect with databases. The common theme between the two is JavaScript and this is also the key benefit of using either stack. One can basically avoid any syntax errors or any confusion by just coding in one programming language, JavaScript. Another advantage of building web projects with MERN is the fact that one can benefit from its enhanced flexibility. In order to understand MERN stack, we need to understand the four components that make up the MERN namely

 – MongoDB, Express.js, React and Node.js

The MERN architecture allows you to easily construct a 3-tier architecture (frontend, backend, database) entirely using JavaScript and JSON.

**React.js Front End**

The top tier of the MERN stack is React.js, the declarative JavaScript framework for creating dynamic client-side applications in HTML. React lets you build up complex interfaces through simple Components, connect them to data on your backend server, and render them as HTML.

React’s strong suit is handling stateful, data-driven interfaces with minimal code and minimal pain, and it has all the bells and whistles you’d expect from a modern web framework: great support for forms, error handling, events, lists, and more.

**Express.js and Node.js Server Tier**

The next level down is the Express.js server-side framework, running inside a Node.js server. Express.js bills itself as a “fast, unopinionated, minimalist web framework for Node.js,” and that is indeed exactly what it is. Express.js has powerful models for URL routing (matching an incoming URL with a server function), and handling HTTP requests and responses.

By making XML HTTP Requests (XHRs) or GETs or POSTs from your React.js front-end, you can connect to Express.js functions that power your application. Those functions in turn use MongoDB’s Node.js drivers, either via callbacks for using Promises, to access and update data in your MongoDB database.

**MongoDB Database Tier**

If your application stores any data (user profiles, content, comments, uploads, events, etc.), then you’re going to want a database that’s just as easy to work with as React, Express, and Node.

That’s where MongoDB comes in: JSON documents created in your React.js front end can be sent to the Express.js server, where they can be processed and (assuming they’re valid) stored directly in MongoDB for later retrieval. Again, if you’re building in the cloud, you’ll want to look at Atlas.

**Pre-requisite**

Hands-on knowledge of JavaScript, HTML and CSS is essential before working on the concepts for making of webpages. Make sure that you have the browser or chrome installed and running before opening website.

**Chapter 2**

**Technologies Used**

**MongoDB**

MongoDB is a cross-platform document-oriented NoSQL database used for high volume data storage that provides high performance, high availability and easy scalability. MongoDB stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time. The document model maps to the objects in the application code, making data easy to work with. The data model available within MongoDB allows users to represent hierarchical relationships to store arrays, and other more complex structures more easily. MongoDB works on concept of collections and documents. Each database contains collections which in turn contains documents. Each document can have varying number of fields. The size and content of each document can also be different from each other.

**Key Components of MongoDB Architecture**

**1.\_id** **–**This is a 24-digit unique identifier field required in every MongoDB document in the collection. The \_id field is like the document's primary key. If the user creates a new document without an \_id field, MongoDB will automatically create the field.

**2.** **Collection** - Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Typically, all documents in a collection are of similar or related purpose.

**3.** **Document** - A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

**4.** **Database** - Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.

**5.** **Field -** A name-value pair in a document. A document has zero or more fields. Fields are analogous to columns in relational databases.

**Express**

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. It is an open source framework developed and maintained by the Node.js foundation. Express provides us the tools that are required to build our app, be it single-page, multi-page or hybrid web applications. It is flexible as there are numerous modules available on npm(NodePackage Manager), which can be directly plugged into Express. Unlike its competitors like Rails and Django, which have an opinionated way of building applications, Express has no "best way" to do something. It is very flexible and pluggable. Pug (earlier known as Jade) is a terse language for writing HTML templates. It produces HTML, supports dynamic code and code reusability (DRY). It is one of the most popular template languages used with Express. Express can be thought of as a layer built on the top of the Node.js that helps manage a server and routes. It allows users to setup middleware to respond to HTTP Requests and defines a routing table which is used to perform different actions based on HTTP method and URL. Express allows to dynamically render HTML Pages based on passing arguments to templates. Express is asynchronous and single threaded and performs I/O operations quickly.

**Why use Express?**

• Ultra-fast I/O.

• Asynchronous and single threaded.

• MVC like structure.

• Robust API makes routing easy

**ReactJS**

ReactJS is a declarative, efficient, and flexible JavaScript library for building reusable UI components. It is an open-source, component-based front-end library which is responsible only for the view layer of the application. It was initially developed and maintained by Facebook and later used in its products like WhatsApp & Instagram. A ReactJS application is made up of multiple components, each component responsible for outputting a small, reusable piece of HTML code. The components are the heart of all React applications. These Components can be nested with other components to allow complex applications to be built of simple building blocks. ReactJS uses virtual DOM based mechanism to fill data in HTML DOM. The virtual DOM works fast as it only changes individual DOM elements instead of reloading complete DOM every time. Instead of using regular JavaScript, React codes are written in something called JSX (JavaScript Syntax Extension). JSX is basically a syntax extension of regular JavaScript and is used to create React elements. These elements are then rendered to the React DOM. JSX is faster than normal JavaScript as it performs optimizations while translating to regular JavaScript.

**Why use React?**

• Uses virtual DOM which is a JavaScript object. This will improve apps performance, since JavaScript virtual DOM is faster than the regular DOM.

• Can be used on client and server side as well as with other frameworks.

• Component and data patterns improve readability, which helps to maintain larger apps.

**NodeJS**

Node.js is a very powerful JavaScript-based platform built on Google Chrome's JavaScript V8Engine. It is used to develop I/O intensive web applications like video streaming sites, single-page applications, and other web applications. Node.js is open source, completely free, and used by thousands of developers around the world. Node.js is a server-side platform built on Google Chrome's JavaScript Engine (V8 Engine). Node.js was developed by Ryan Dahl in 2009. Node.js applications are written in JavaScript and can be run within the Node.js runtime on OSX, Microsoft Windows, and Linux. Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

**Features of Node.js**

**1.** **Extremely fast:** Node.js is built on Google Chrome's V8 JavaScript Engine, so its library is very fast in code execution.

**2.** **I/O is Asynchronous and Event Driven:** All APIs of Node.js library are asynchronous i.e. non-blocking. So, a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call. It is also a reason that it is very fast.

**3.** **Single threaded:** Node.js follows a single threaded model with event looping.

**4.** **Highly Scalable:** Node.js is highly scalable because event mechanism helps the server to respond in a non-blocking way.

**5.** **No buffering:** Node.js cuts down the overall processing time while uploading audio and video files. Node.js applications never buffer any data. These applications simply output the data in chunks.

**6.** **Open source:** Node.js has an open-source community which has produced many excellent modules to add additional capabilities to Node.js application.

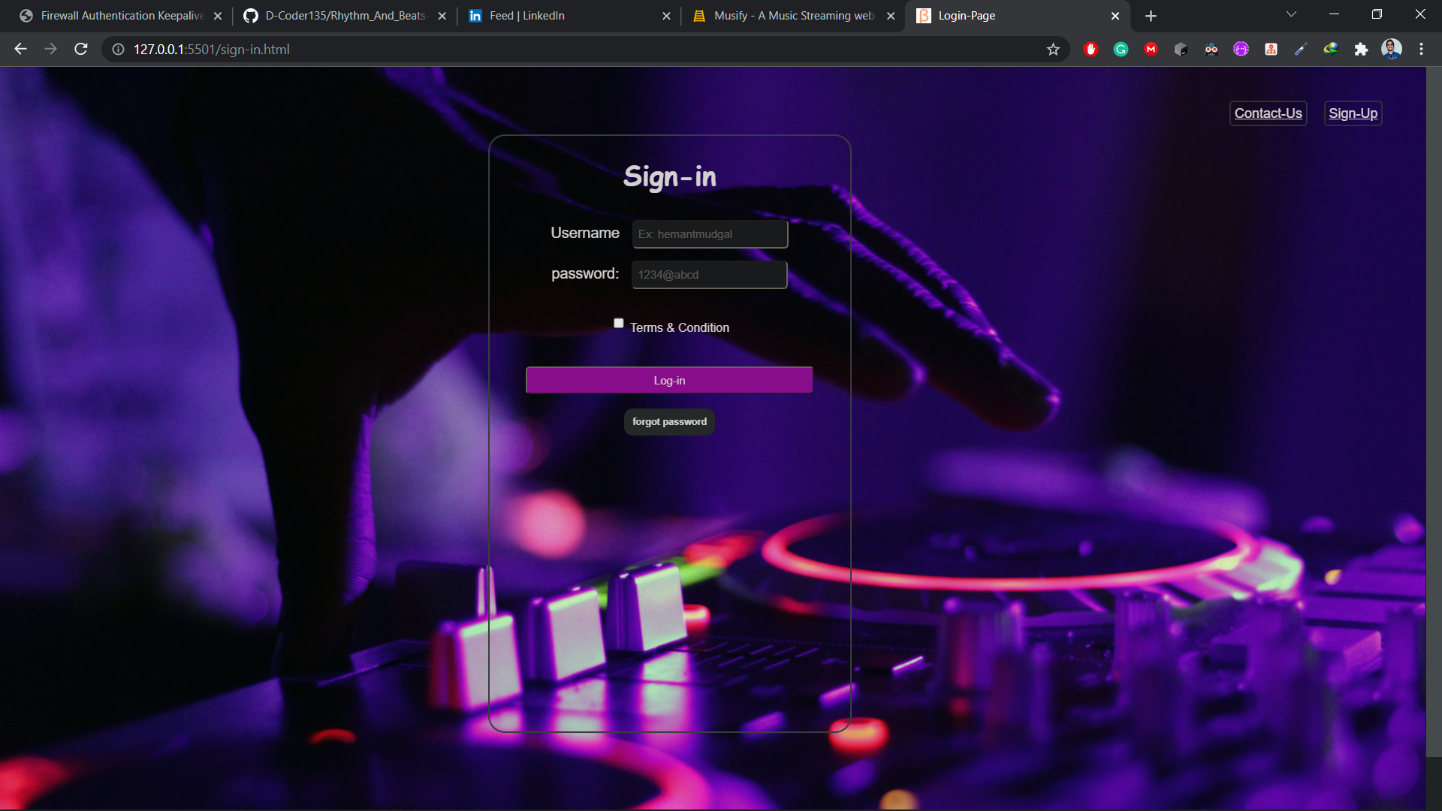
**Chapter 3**

**List of Figures**

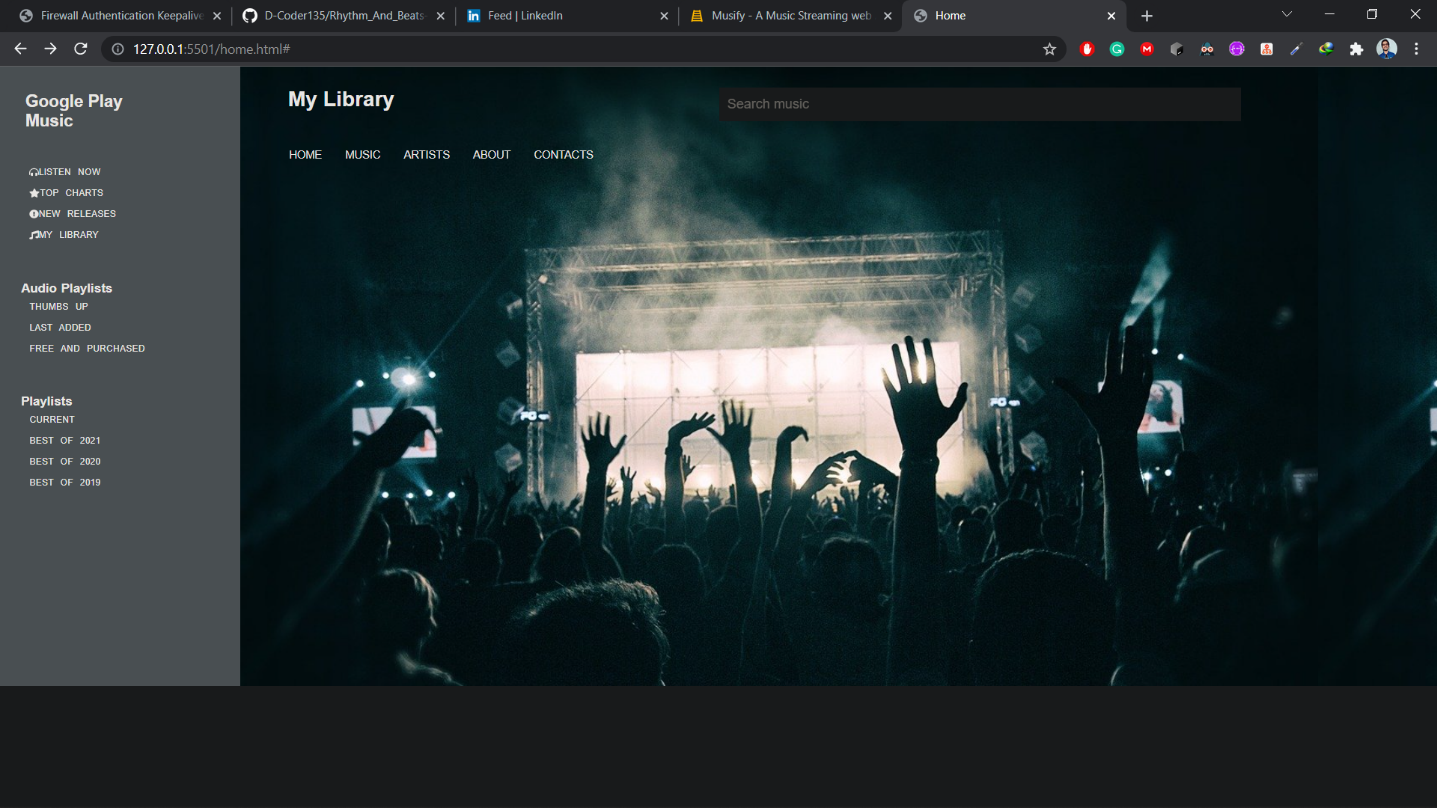
**1. SIGN-UP PAGE**

****

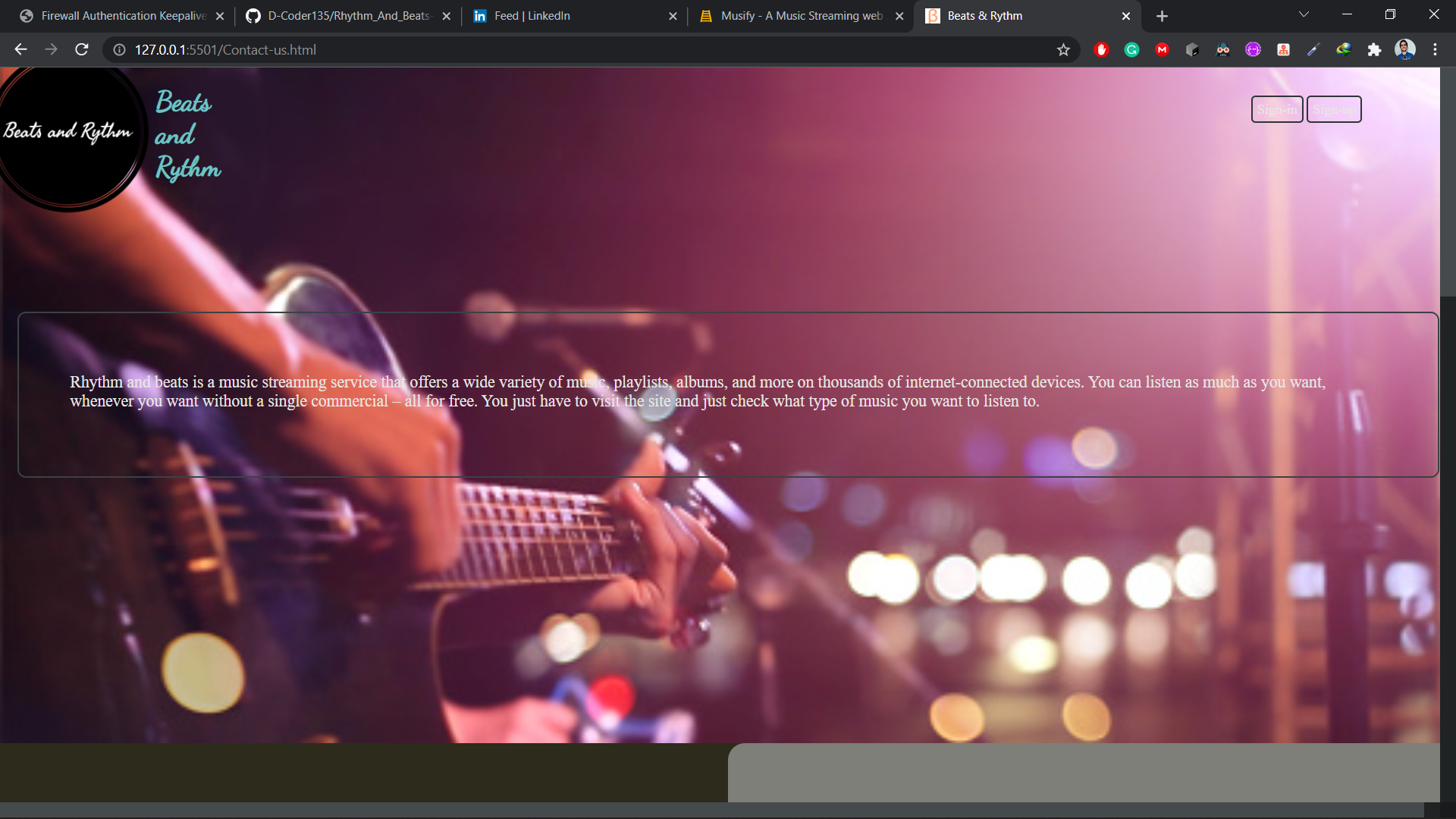
**2. LOGIN PAGE**

****

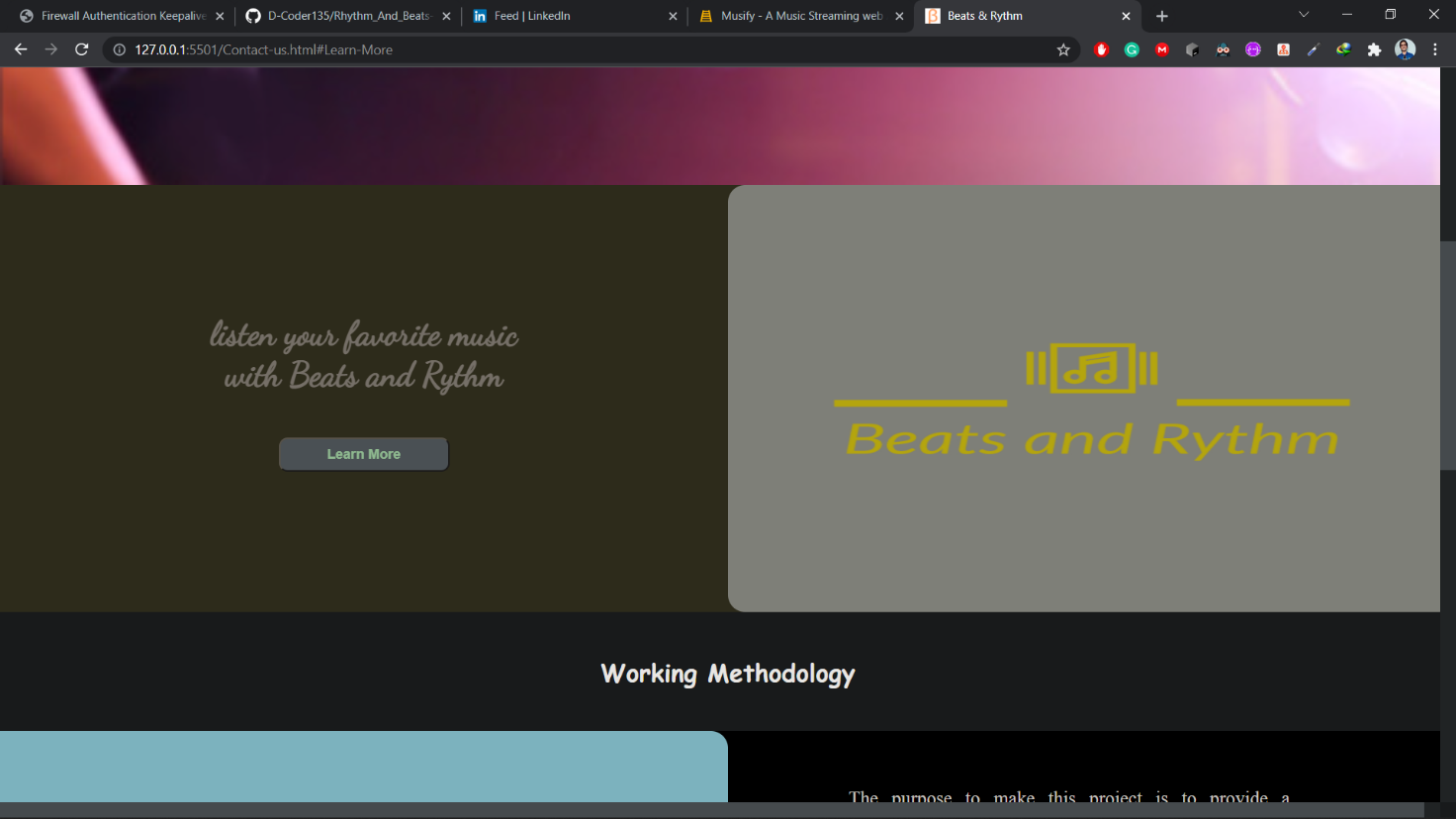
**3. HOMEPAGE**

****

**4. CONTACT US PAGE**

****

**5.**



**Chapter 4**

**Software Testing**

Once source code has been generated, software must be tested to uncover as many errors as possible before delivery. It is very important to work the system successfully and achieve high quality of software. Testing include designing a series of test cases that have a high likelihood of finding errors by applying software-testing techniques. System testing makes logical assumptions that if all the parts of the system are correct, the goal will be successfully achieved. The system should be checked logically. Validations and cross checks should be there. Avoid duplications of record that cause redundancy of data. In other Words, Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. It is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

The preliminary goal of implementation is to write source code and internal documentation so that conformance of the code to its specifications can be easily verified, and so that debugging, testing and modifications are eased. This goal can be achieved by making the source code as clear and straightforward as possible. Simplicity, clarity and elegance are the hallmark of good programs, obscurity, cleverness, and complexity are indications of inadequate design and misdirected thinking. Source code clarity is enhanced by structured coding techniques, by good coding style, by, appropriate supporting documents, by good internal comments, and by feature provided in modern programming languages. The implementation team should be provided with a well-defined set of software requirement, an architectural design specification, and a detailed design description. Each team member must understand the objectives of implementation.

4.1 TERMINOLOGY

Error The term error is used in two ways. It refers to the difference between the actual output of software and the correct output, in this interpretation, error is essential a measure of the difference between actual and ideal. Error is also to used to refer to human action that result in software containing a defect or fault.

Fault is a condition that causes to fail in performing its required function. A fault is a basic reason for software malfunction and is synonymous with the commonly used term Bug.

Failure is the inability of a system or component to perform a required function according to its specifications. A software failure occurs if the behavior of the software is the different from the specified behavior. Failure may be caused due to functional or performance reasons.

4.2 TYPES OF TESTING

**a. Unit Testing** The term unit testing comprises the sets of tests performed by an individual programmer prior to integration of the unit into a larger system. A program unit is usually small enough that the programmer who developed it can test it in great detail, and certainly in greater detail than will be possible when the unit is integrated into an evolving software product. In the unit testing the programs are tested separately, independent of each other. Since the check is done at the program level, it is also called program teasing.

**b. Module Testing** A module and encapsulates related component. So can be tested without other system module.

**c. Subsystem Testing** Subsystem testing may be independently design and implemented common problems are sub-system interface mistake in this checking we concenton it. There are four categories of tests that a programmer will typically perform on a program unit.

i Functional test

ii Performance test

iii Stress test

iv Structure test

**Functional Test** Functional test cases involve exercising the code with Nominal input values for which expected results are known; as well as boundary values (minimum values, maximum values and values on and just outside the functional boundaries) and special values.

**Performance Test** Performance testing determines the amount of execution time spent in various parts of the unit, program throughput, response time, and device utilization by the program unit. A certain amount of avoid expending too much effort on fine-tuning of a program unit that contributes little to the overall performance of the entire system. Performance testing is most productive at the subsystem and system levels.

**Stress Test** Stress test are those designed to intentionally break the unit. A great deal can be learned about the strengths and limitations of a program by examining the manner in which a program unit breaks.

**Structure Test** Structure tests are concerned with exercising the internal logic of a program and traversing particular execution paths. Some authors refer collectively to functional performance and stress testing as “black box” testing. While structure testing is referred to as “white box” or “glass box” testing. The major activities in structural testing are deciding which path to exercise, deriving test date to exercise those paths, determining the test coverage criterion to be used, executing the test, and measuring the test coverage achieved when the test cases are exercised.

**Chapter 5**

**Conclusion**

We have completed our project within time limit with the coordination of our team members under the supervision of our mentor Mr. Pankaj Kapoor.

Our project repository is available at

https://github.com/D-Coder135/Rhythm\_And\_Beats-Music-Streaming-Website-

**Chapter 6**

**Bibliography**

[**www.google.com**](http://www.google.com)

**www.geeksforgeeks.org**

[**www.youtube.com**](http://www.youtube.com)

**www.w3schools.com**

**www.beta-labs.in**