NAAN MUDHALVAN PROJECT REPORT

PROJECT TITLE RHYTHMIC TUNES–MUSIC

TEAM LEADER:

NAME : SAMYUKATHA.K(code developer)

EMAIL_ID :samyu6802@gmail.com

TEAM MEMBERS:

NAME	MAIL ID
• S.Bavadaharani(code developer**)**	bavadaharani2209@gmail.com
• C.harshini(documentation)	achuchinna007@gmail.com
• S.Shameem **(**demo video linking)	shameemshameem3333@gmail.com

2.PROJECT OVERVIEW:

- Rhythmic Tunes is a project that explores the power of rhythm and music.It focuses on
- creating simple rhythmic patterns using instruments or digital tools.The project studieshow rhythm affects mood, focus, and creativity.Participants can experience rhythm through listening, clapping playing.

Features :

- Creative Rhythms – Generates simple and engaging rhythmic patterns.
- Interactive Experience – Allows participation through clapping, tapping, or playing. Mood Enhancement – Tunes designed to boost energy, focus, or relaxation.
- Cultural Blend – Showcases rhythms from traditional and modern music Easy Accessibility – Can be enjoyed through digital tools or simple

3.ARCHITECTURE:

- Input Layer – Users provide input by clapping, tapping, singing, or using digita instruments.
- Processing Layer – The system/software records and analyzes beats, rhythm, and patterns.
- Rhythm Generator – Creates rhythmic tunes using pre-set patterns or user-created sequences
- Output Layer – Plays back the rhythmic tunes through speakers, headphones, or instruments

- Feedback Layer – Users can listen, repeat, or modify the rhythm for creativity and learning.

Role:

The user interface that delivers a smooth, responsive, and interactive experience.

Technologies Used:

- React.js: Component-based structure for dynamic UI.
- Bootstrap**:** Layout grid system, responsiveness, and basic styling.
- Material UI**:** Modern, sleek UI components (buttons, cards, modals, etc.).**Backend: Node.js + Express.js**

Role:

Handles business logic, API routing, user authentication, and connection with the database.

Technologies Used:

- **Node.js**: Event-driven, non-blocking backend runtime for handling high concurrency.
- **Express.js**: Lightweight framework to build RESTful APIs and manage server-side logic.

Database: MongoDB

Stores structured and unstructured data in flexible JSON-like documents. [

React.js (Frontend)]

|

```
         | REST API Calls

         ↓

[ Node.js + Express.js (Backend) ]

         |

         | Mongoose Queries

         ↓

[ MongoDB (Database) ]
```

4.SETUP INSTRUCTIONS:

Prerequisites:

- Node.js
- MongoDB
- Git
- React.js
- Express.js – Mongoose – Visual Studio Code

Installation Steps

- Clone the Repository

```
git clone <your-repo-url>
cd <repo-folder-name>
```

- Install Client Dependencies

```
cd client
npm install
```

- Install Server Dependencies

```
Cd ../server
npm install
```

Start the Application

Start Client (Frontend):

Bash

```
npm start
```

StartServer(Backed)

```
cd server

npm start
```

5. FOLDER STRUCTURE:

```
  rhythmic-tunes565/
 |
 ├── src/                      # All source code
|   ├── assets/                # Audio files, images, etc.
|   |   ├── audio/             # Sound files (e.g., .mp3, .wav)
|   |   └── images/            # UI images, icons
|   |
|   ├── components/            # Reusable UI or logic components
|   ├── modules/               # Feature-specific code (e.g., beat generator)
|   |   ├── player/            # Music player logic
|   |   ├── sequencer/         # Rhythm/timing features
|   |   └── recorder/          # Audio recording/upload
|   |
|   ├── utils/                 # Helper functions
|   ├── config/                # App config, constants
|   └── main.py / app.js       # App entry point (based on language)
|
├── public/                    # Static files (index.html, icons, etc.)
```

```
|
├── tests/                    # Unit and integration tests
|   ├── test_player.py          # Example test file
|   └── ...
|
├── README.md                  # Project overview
├── requirements.txt           # Python dependencies
├── package.json              # JS/Node dependencies
├── .gitignore               # Git ignore rules
└── LICENSE                  # Optional license file
```

- **<u>RUN THE APPLICATION:</u>**

fronten

cd client npm

start **backen**

cd server npm

start

Access: visit http://localhost:3000

- **<u>COMPONENT DOCUMENTATION:</u>**

Key Components:

Handles beat toggling and visual timeline.

Handles beat toggling and visual timeline.

Reusable Components:

Generic clickable button with customizable styles and icons.

Used in PlaybackControls, PatternManager, etc.

- **<u>STATE MANAGEMENT:</u>**

Global State:

Managed Context for music, favorites, and user login status.

Local State:

Form input states managed inside Add MusicForm.

- **USER INTERFACE:**

Include screenshots or GIFs of:

- Home page showing music
- music detail page
- Adding a music

- **STYLING:**

CSS Frameworks/Libraries:

Tailwind CSS for styling; Styled Components for scoped styles.

Theming:

Dark and light mode toggle implemented via context.

- **TESTING:**

Unit testing:

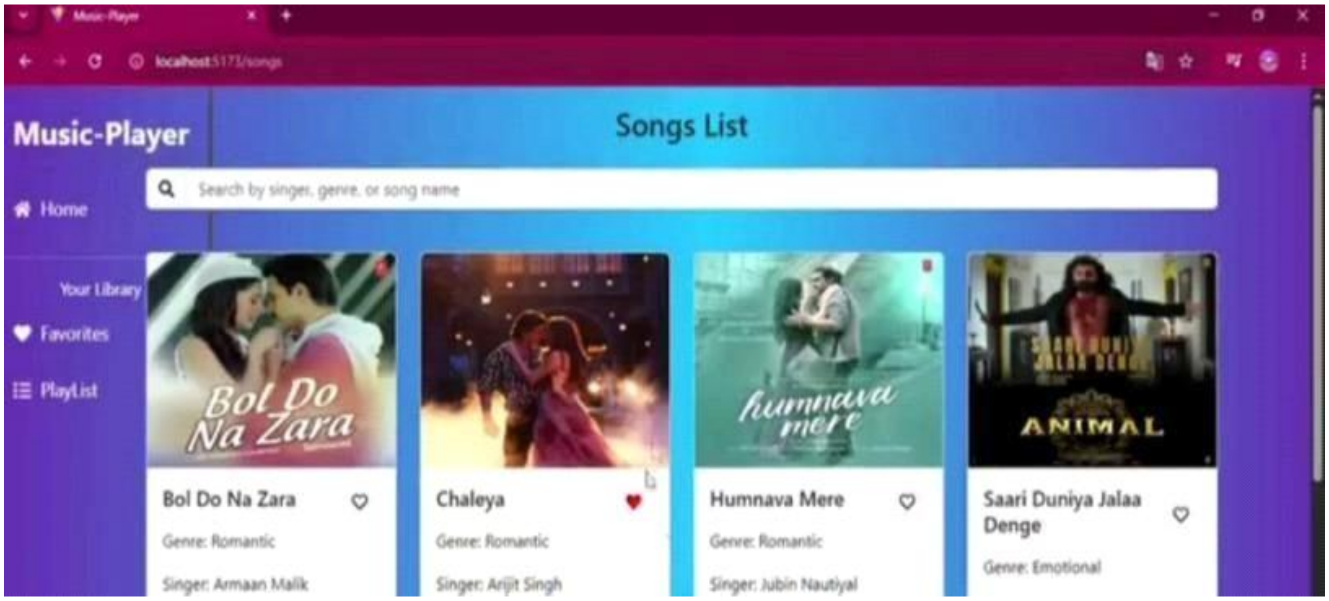
Testing individual components or functions in isolation to ensure they work correctly.

Integration testing:

Testing how different components or modules work together as a whole

- **<u>SCREENSHOTS OR DEMO:</u>**

Add actual screenshots or a demo link:



<u>FUTURE ENHAMNCEMENT</u>:

- AI-assisted rhythm analysis – tools to detect and correct rhythmic irregularities in composition

- Interactivelearningapps– softwarethattrainsstudentstointernalizecomplexmetersandpolyrhy

- Dynamicnotationsystems– smarternotationthatbetterrepresentsswing,groove,andhumanfe

- Cross-culturalrhythmintegration– blendingrhythmictraditions(Indiantala,Africanpolyrhythm Western meter)fornewpossibilities.