Team meeT

Documentation

Brief

Team meeT is an MS Teams clone web application. Users can sign up using email and password or Google account. Upon signing in, groups can be made with other members of the website. Groups allow users to have conversations before and during video call meetings.

Tech-stack

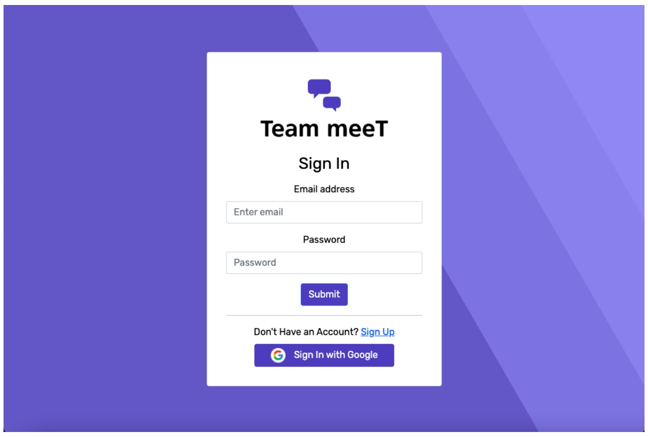
Frontend:

1. **ReactJS**
2. Bootstrap
3. Material UI (Icons)

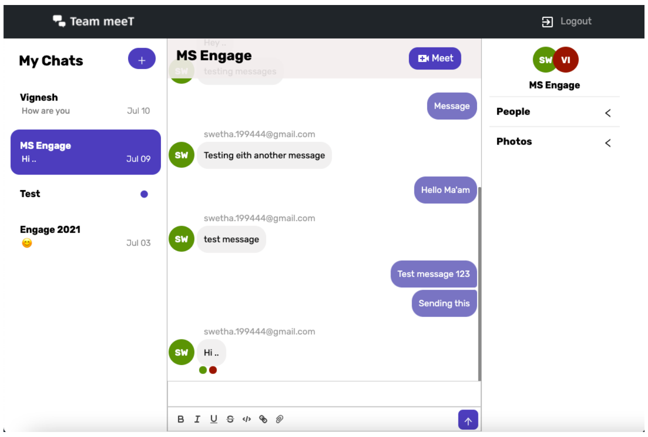
Backend:

1. **Jitsi Meet** - Jitsi External API for video calling
2. **Firebase** - User Authentication & Firestore database
3. **ChatEngine** - Enable chat conversations before and during video calls

Features



Email/Google Authentication



testing messages

Message

swetha.199444@gmail.com

Testing eith another message

Hello Ma'am

swetha.199444@gmail.com

test message

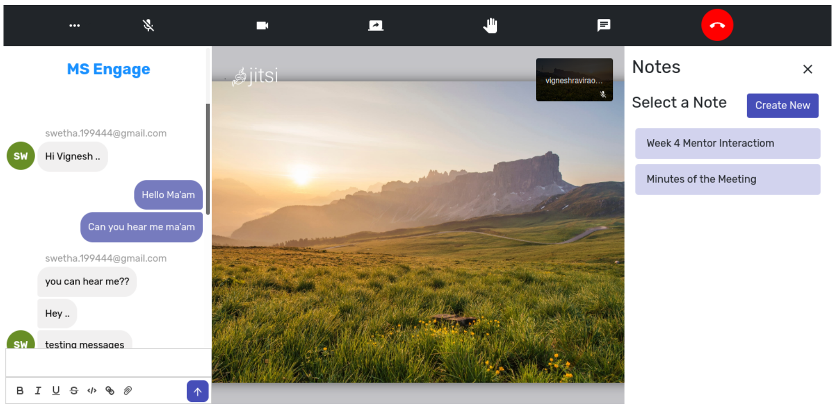
Test message 123

Sending this

swetha.199444@gmail.com

Hi ..

Group Chat



swetha.199444@gmail.com

Hi Vignesh ..

Hello Ma'am

Can you hear me ma'am

swetha.199444@gmail.com

you can hear me??

Hey ..

testing messages

Week 4 Mentor Interaction

Minutes of the Meeting

Video Calling with-

Group chat

Screenshare

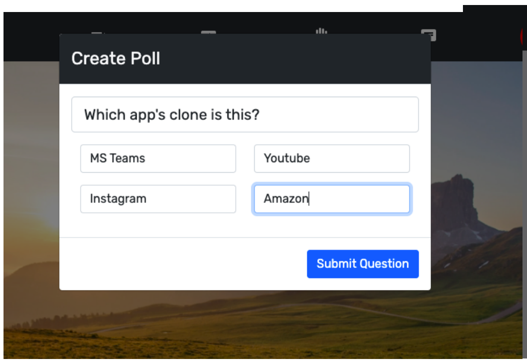
Raisehand

Pin/Kick Participants

Download Attendance

Meeting Notes

Polls - Admins create polls during meeting and download results



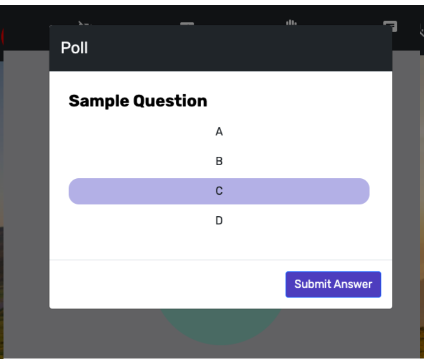
Which app's clone is this?

MS Teams

Youtube

Instagram

Amazon



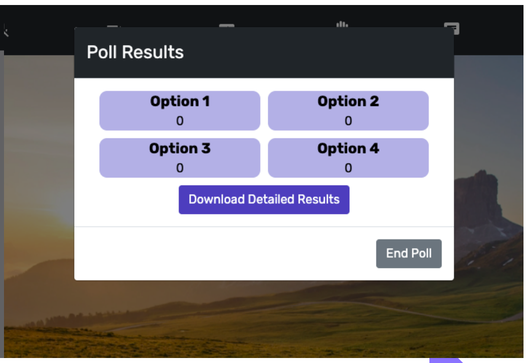
Sample Question

A

B

C

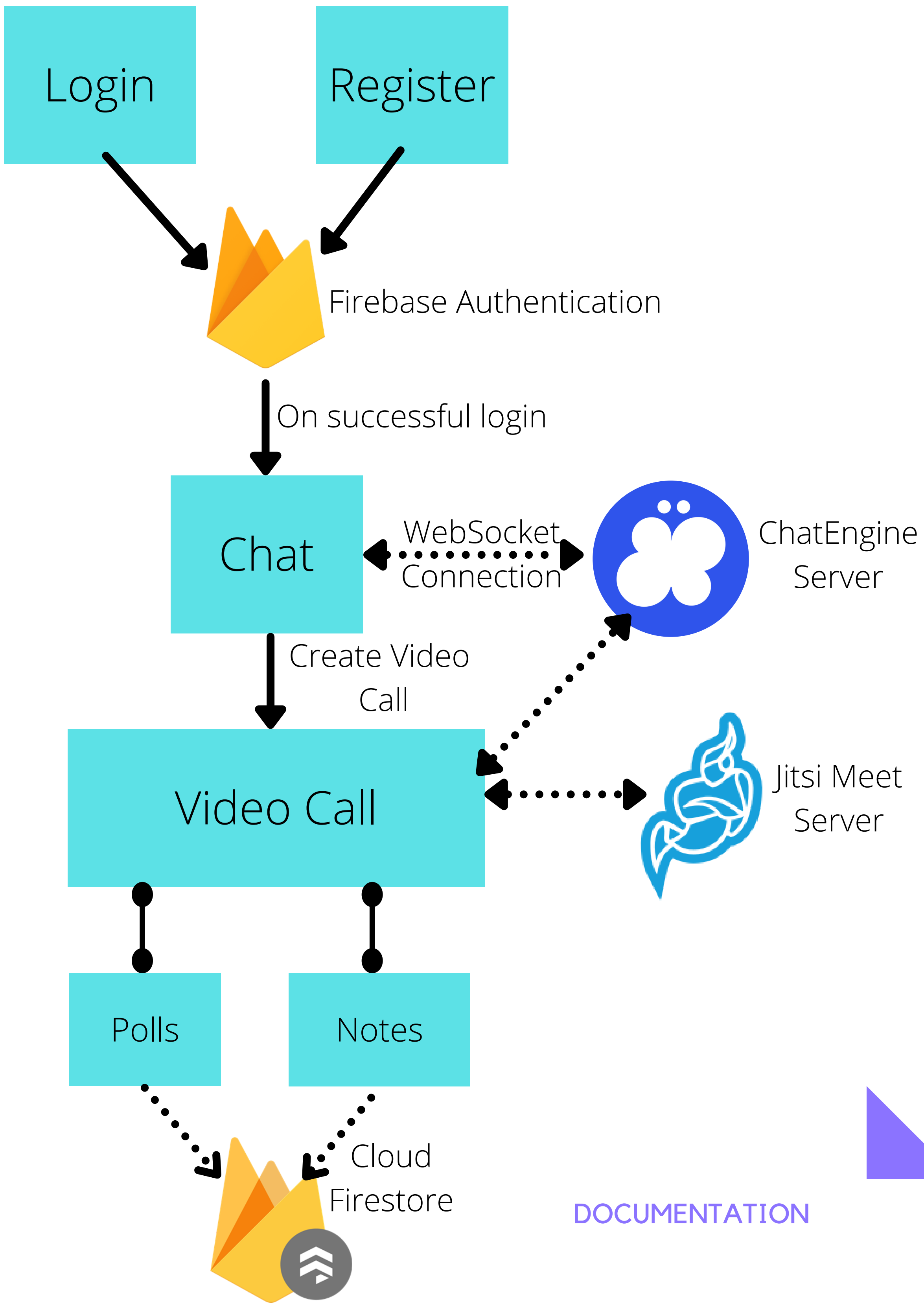
D



Option 1 0	Option 2 0
Option 3 0	Option 4 0

Implementation Details

Component Diagram

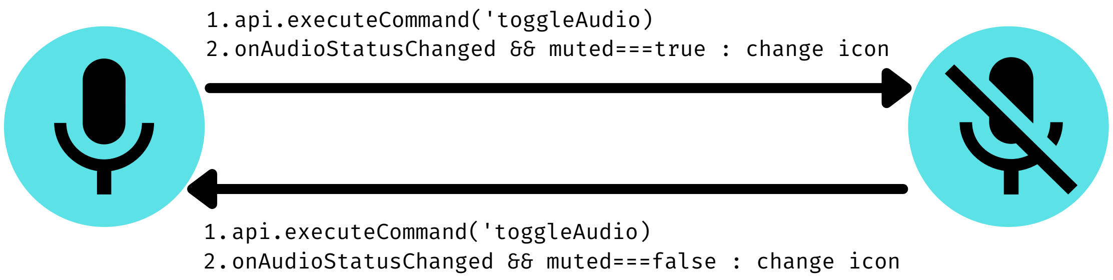


Firebase Integration

- A project was created for the app on Firebase console and linked using API Keys.
- An **AuthContext** component at the **root** of the React component tree passes the authenticated user's details to all components in the subtree.
- Firestore database was used to fetch and store poll question details - questions and answers.

Jitsi Integration

- An **IFrame** provided by **Jitsi Meet** was embedded into the Video call screen.
- The video call screen was customized by overwriting the **config and inferface** options
- Event listeners were hooked onto the Jitsi meeting to account for changes in participant list, video/audio toggles etc.
- The custom toolbar buttons use two states to display the current video, audio, raisehand and screenshare status.
- The toolbar button icons change state when triggered by Jitsi rather than onClick to synchronise changes in the custom toolbar with the IFrame.


1.api.executeCommand('toggleAudio)
2.onAudioStatusChanged && muted===true : change icon
1.api.executeCommand('toggleAudio)
2.onAudioStatusChanged && muted===false : change icon

ChatEngine Integration

- A project was created on ChatEngine.io and was linked to the frontend application using a Project_ID and Project_Secret.
- When a Firebase authenticated user logs in, a **GET Request** is sent to `/users/me` endpoint to check if the user exists on ChatEngine server. In case the user doesn't exist, a POST Request is sent to the `/users` endpoint to create a new user.
- These secret keys were stored as **environment variables** in the deployment server to avoid public access to chats.
- To allow users to continue their group chat during a video call, the VideoCallChat.js component was **connected to the ChatEngine server using a web socket that broadcasts updates specific to the current group chat** (specified by a unique chat_id).
- Since the default Chat Headers do not come with any buttons, a custom ChatHeader component was created to allow users to create. VideoCall and transfer control to the VideoCall component.

Notes during Video Call

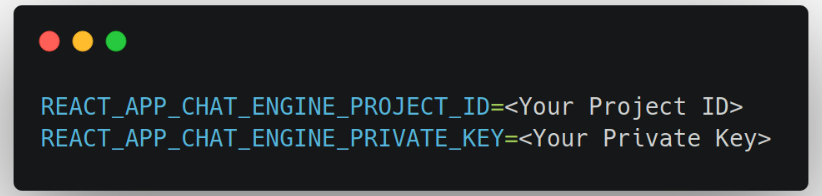
- Notes were stored in Cloud Firestore database with the following paths:
  - `<ChatID>/Notes/<UID>`
- UID refers to the unique identification string provided by Firebase for each user

Polls during Video Call

- Polls were stored in Cloud Firestore database with the following paths:
  - `Questions: <ChatID>/Poll/Questions/details`
  - `Answers: <ChatID>/Poll/Answers`
- The `details` document contains the following fields:
  - `Question, Option1, Option2, Option3, Option4, Valid`
  - `Valid` is a boolean field such that `Valid == true` **implies that the poll is currently ongoing**
- In the `Answers` subcollection, documents were created with participant email as document ID and the following fields:
  - `option, question`
- **For non-admins:** The VideoCall component listens for changes to the question details document. Whenever the Valid field turns true, a Modal pops ups with the question and options. When the user submits, a document is created in the Answers subcollection.
- **For admins**: Whenever a new poll is the details document is updated with Valid = true. Then, the Answers subcollection is listened to for updates about participant answers. Lastly, the question field in the participants' document is compared to ensure only answers to the current question are considered.

Steps to run locally

1. Run the following commands
   - `$ git clone https://github.com/vrrao01/team_meet.git`
   - `$ cd team_meet`
2. Create a project on ChatEngine.io *
   a. Click on Sign Up and create an account
   b. After you have been signed in, click on "New Project"
   c. Make note of your ProjectID and PrivateKey
3. In the team_meet folder, create file named ".env" and save the credentials created in step 2c as follows:



```
REACT_APP_CHAT_ENGINE_PROJECT_ID=<Your Project ID>
REACT_APP_CHAT_ENGINE_PRIVATE_KEY=<Your Private Key>
```

4. Now, run the command:
   - `npm start`

Agile Workflow

Sprints

1

Week 1 : Research on available tools that can be used to build the app. Create a design of the components and features to add in the app.

2

Week 2 : Build small prototypes using WebRTC, Open Vidu and Jitsi External API that meet the minimum criteria. Then, decided to go ahead with Jitsi due to its ease of use and ability to handle multiparty conferences without load on browser.

3

Week 3 : Extend the prototype to include the other features planned. GitHub was used to keep track of progress and help with the agile workflow. (discussed below).
The features were added in the following order: Firebase authentication, Chat Engine group chat, Jitsi custom toolbar, Google Authentication, Polls

4

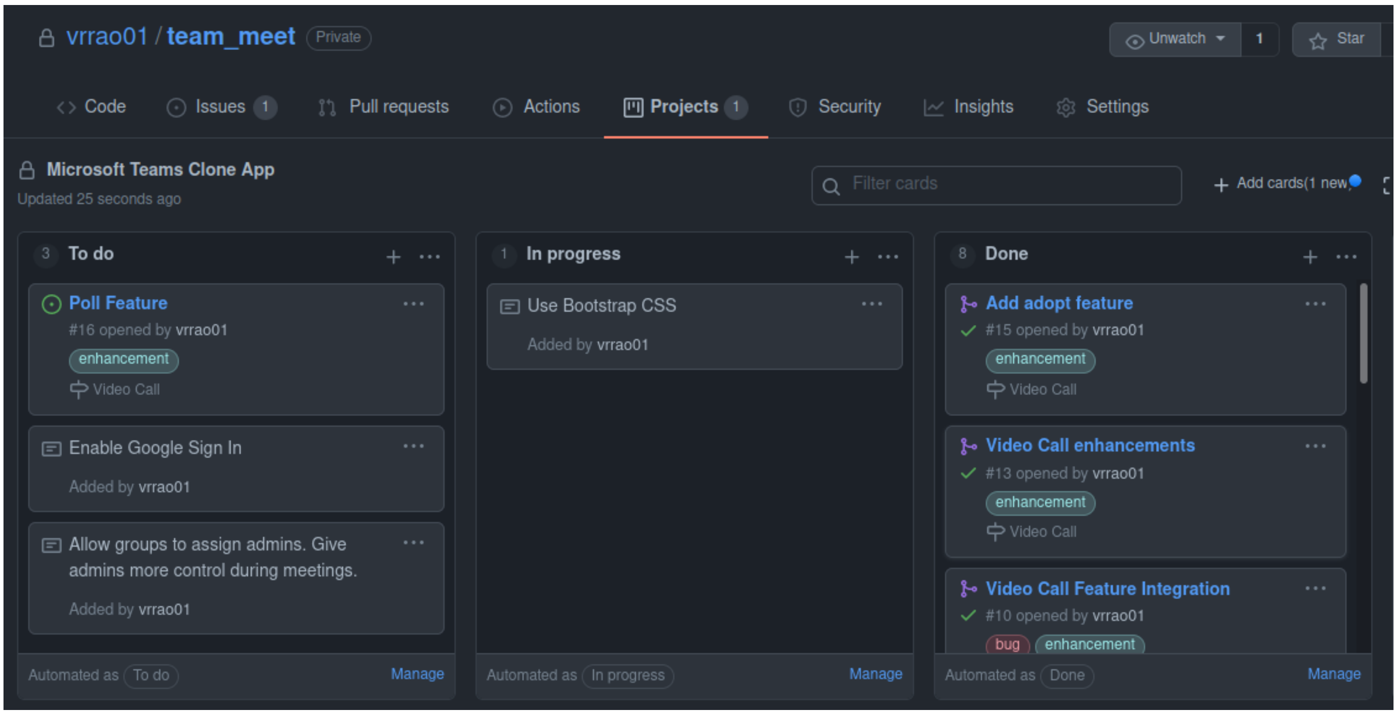
Week 4: Work on integrating **Adopt** feature. Since, Chat Engine was already used for group chats, major work involved allowing users to access existing group chat during video call rather than use the chat option provided by Jitsi. Also work on documentation.

Agile Workflow

GitHub Project Board

A GitHub project was created for the repository to keep track of features to be implemented and prioritise tasks. A **Kanban style project board** was setup with columns for **Todo, In Progress and Done.**
Cards were automatically added into the appropriate columns as follows:

1. **New Issues** are automatically added into the Todo Column
2. **Merged pull requests** from the development branch are added into the Done column.



vrrao01 / team_meet Private

Unwatch 1 Star

Code · Issues 1 · Pull requests · Actions · Projects 1 · Security · Insights · Settings

Microsoft Teams Clone App
Updated 25 seconds ago

Filter cards · Add cards(1 new)

3 To do

- Poll Feature
  #16 opened by vrrao01
  enhancement
  Video Call
- Enable Google Sign In
  Added by vrrao01
- Allow groups to assign admins. Give admins more control during meetings.
  Added by vrrao01

Automated as To do · Manage

1 In progress

- Use Bootstrap CSS
  Added by vrrao01

Automated as In progress · Manage

8 Done

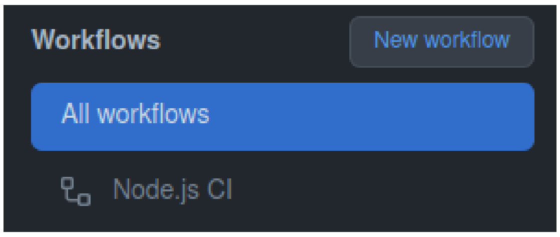
- Add adopt feature
  #15 opened by vrrao01
  enhancement
  Video Call
- Video Call enhancements
  #13 opened by vrrao01
  enhancement
  Video Call
- Video Call Feature Integration
  #10 opened by vrrao01
  bug enhancement

Automated as Done · Manage

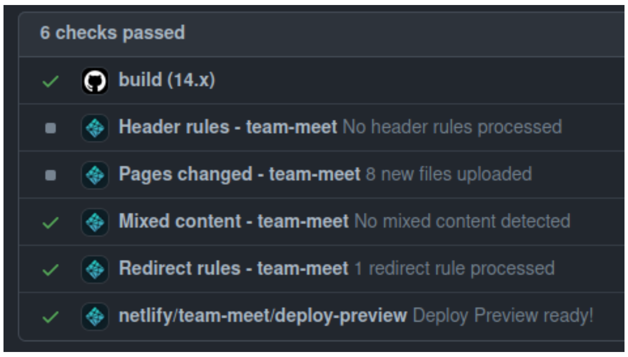
CI/CD

GitHub Actions

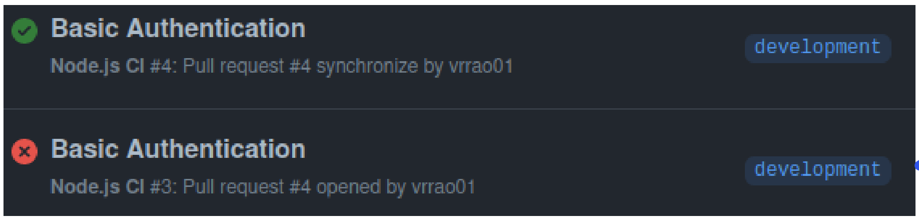
A **NodeJS CI workflow** was added on GitHub actions for the project repository. This ensured that new changes that were pushed **do not break the build step** required for deployment.
By connecting the repository with Netlify, a series of additional checks were done fore each pull request to ensure


Workflows
New workflow
All workflows
Node.js CI

GitHub NodeJS workflow


6 checks passed
✓ build (14.x)
■ Header rules - team-meet No header rules processed
■ Pages changed - team-meet 8 new files uploaded
✓ Mixed content - team-meet No mixed content detected
✓ Redirect rules - team-meet 1 redirect rule processed
✓ netlify/team-meet/deploy-preview Deploy Preview ready!

Checks done before merging pull request


✓ Basic Authentication
Node.js CI #4: Pull request #4 synchronize by vrrao01
development

✗ Basic Authentication
Node.js CI #3: Pull request #4 opened by vrrao01
development

An example of an update that failed the GitHub NodeJS build test

Continuous Deployment was done by adding the repository's *main* branch as the **production branch** on Netlify. All new changes were first pushed to the ***development*** **branch** and merged into the main branch via a pull request if all the build and deployment checks by GitHub and Netlify were successful