Server side testing - College Event Management

Software Testing (CS731)

Submitted By:
Shreyank Buddhadev (MT2021130)
Biswadeep Das (MT2021032)

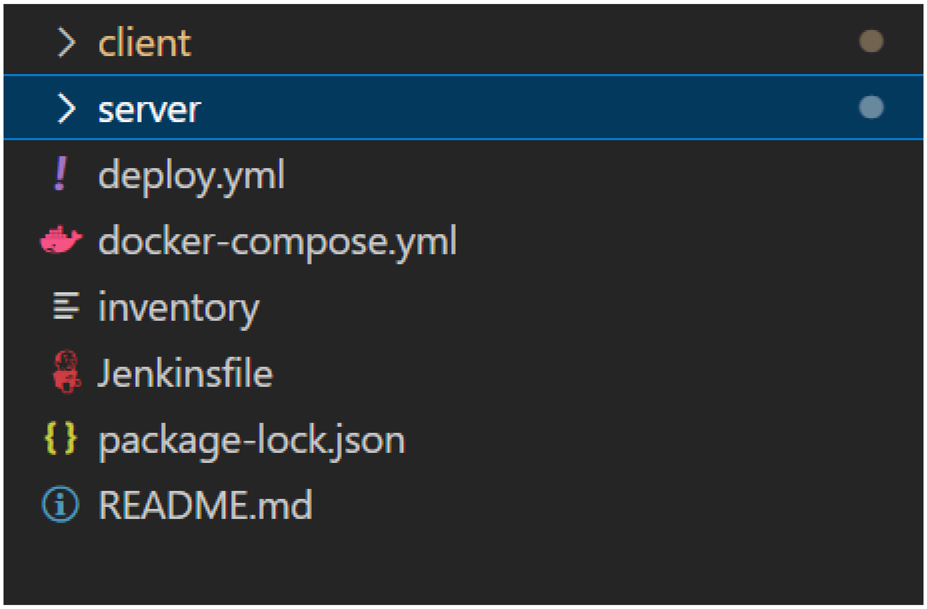
Code repo: https://github.com/ShreyankB/College_Event_Management

Abstract about project:

College event management project is a Webapp that supports hassle free management of college events. It has more than 7-8k lines of code. Codebase is divided into several modules according to dashboard.

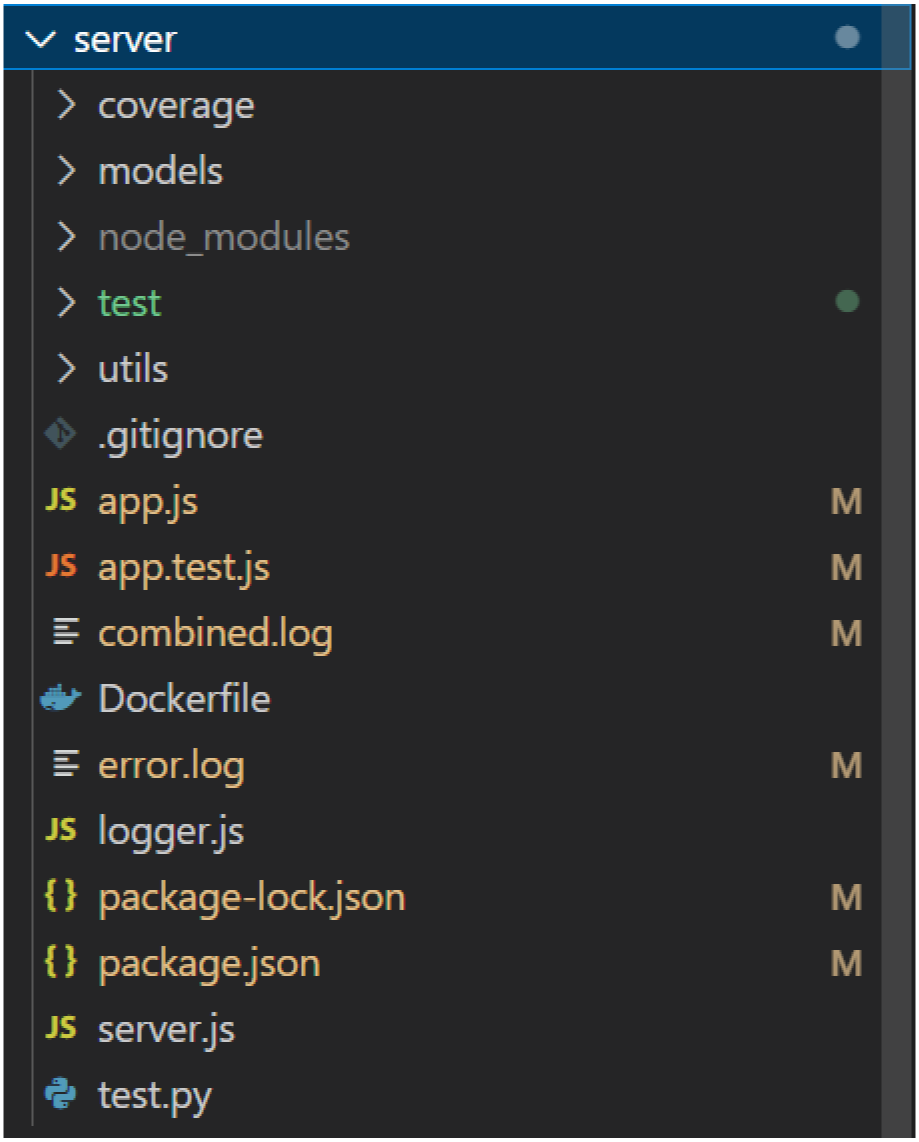
Folder Structure:

Project is divided into client and server module.



```
> client
> server
! deploy.yml
docker-compose.yml
inventory
Jenkinsfile
{} package-lock.json
README.md
```

Server module contains below mentioned files and folders:



```
∨ server                      ●
  > coverage
  > models
  > node_modules
  > test                      ●
  > utils
    .gitignore
  JS app.js                   M
  JS app.test.js              M
    combined.log              M
    Dockerfile
    error.log                 M
  JS logger.js
  {} package-lock.json        M
  {} package.json             M
  JS server.js
    test.py
```

Here, app.test.js is entry point to the project.

Tools used: Jest and Supertest

Why Jest?

Jest is a JavaScript testing framework designed to ensure correctness of any JavaScript codebase. It allows you to write tests with an approachable, familiar and feature-rich API that gives you results quickly.

Jest is well-documented, requires little configuration and can be extended to match your requirements.

Why Supertest?

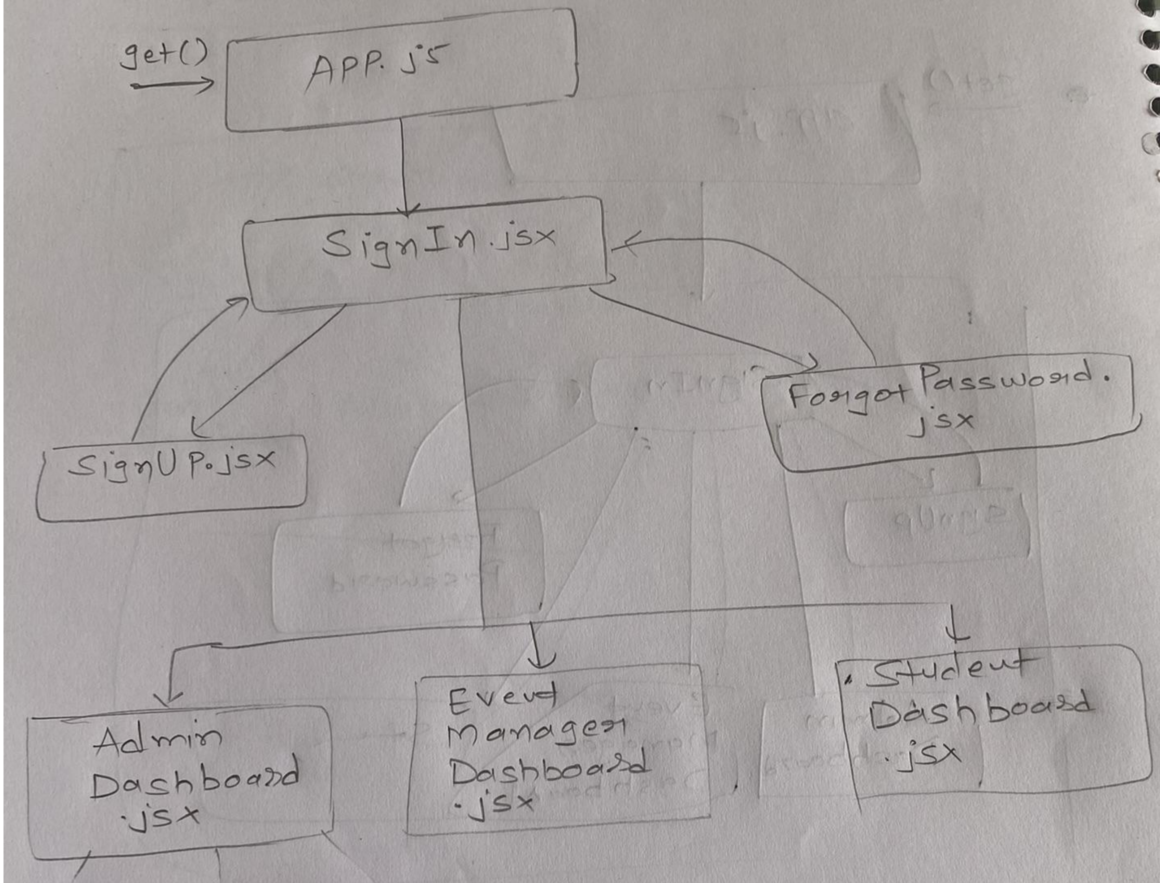
Supertest is a Node.js library that allows developers and testers to test the APIs. It enables developers and testers to write automated tests for routes and endpoints. Supertest can be used as a standalone library or with testing frameworks like Mocha and Jest. It provides a high-level abstraction to test HTTP requests. That is, if the developer's Node.js application is running an HTTP server, they don't need a running server.

You can make requests using Supertest directly. Besides, Supertest can integrate with other testing frameworks easily without any additional tools. With just a few lines of commands by sending a method like GET, PUT, and POST, Supertest can test any HTTP requests easily.

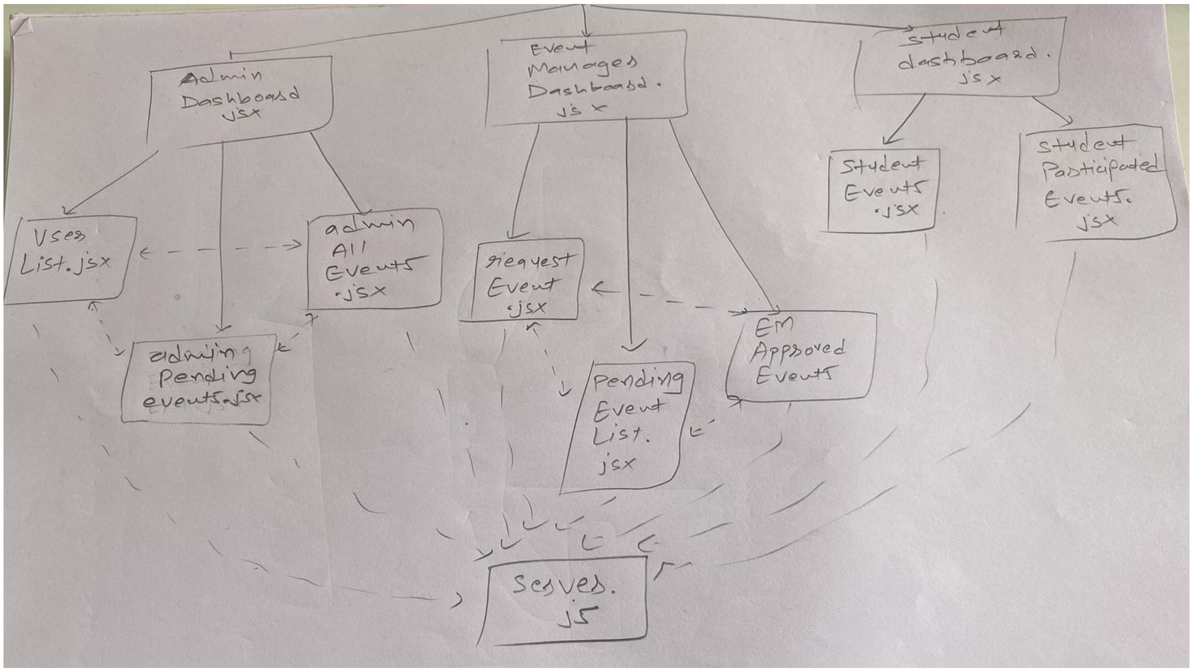
Testing Strategy: Server side testing

ATG Model diagram:

ATG Diagram:



[conti.]

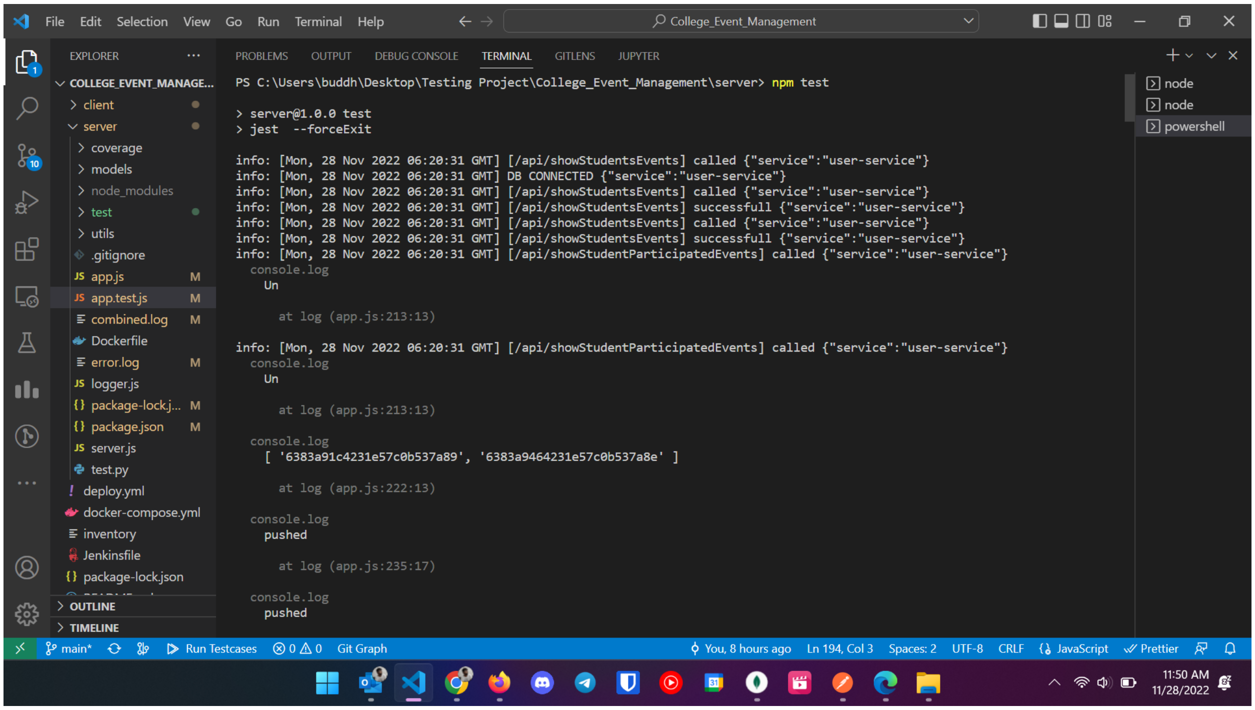


How to run the test suits:

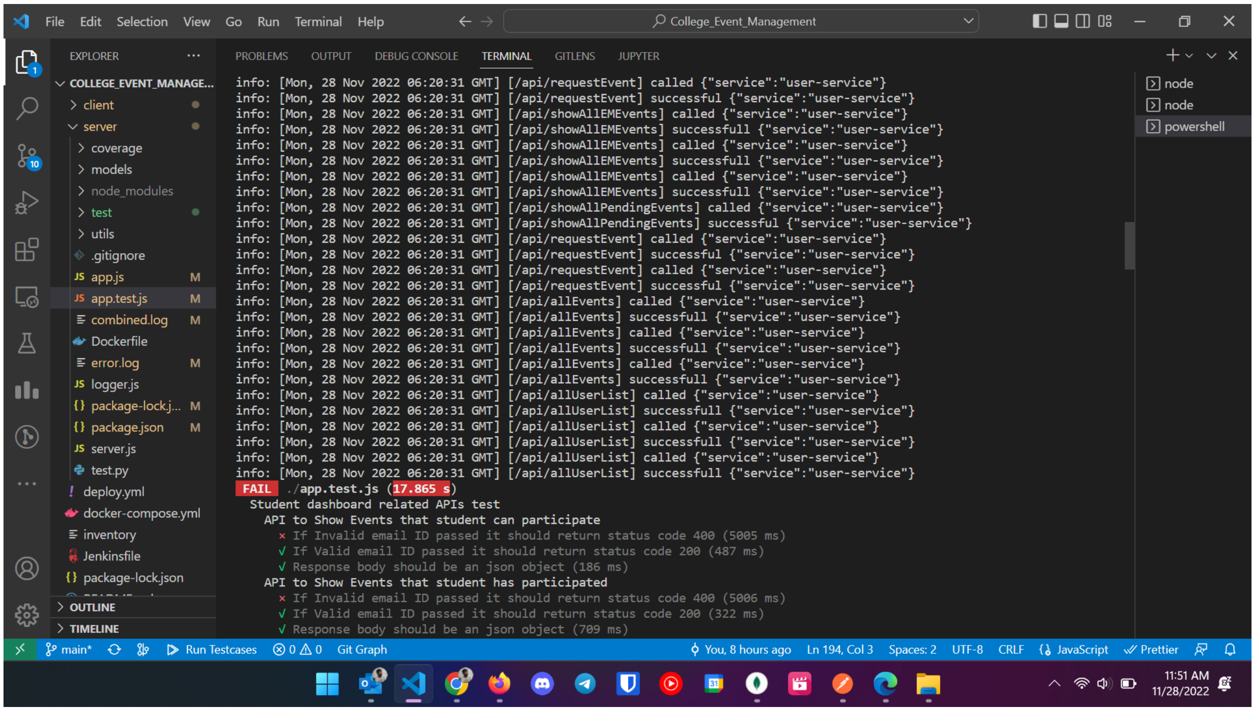
Unzip the repo. Then follow below mentioned commands.

```
cd server
npm i
npm test
```

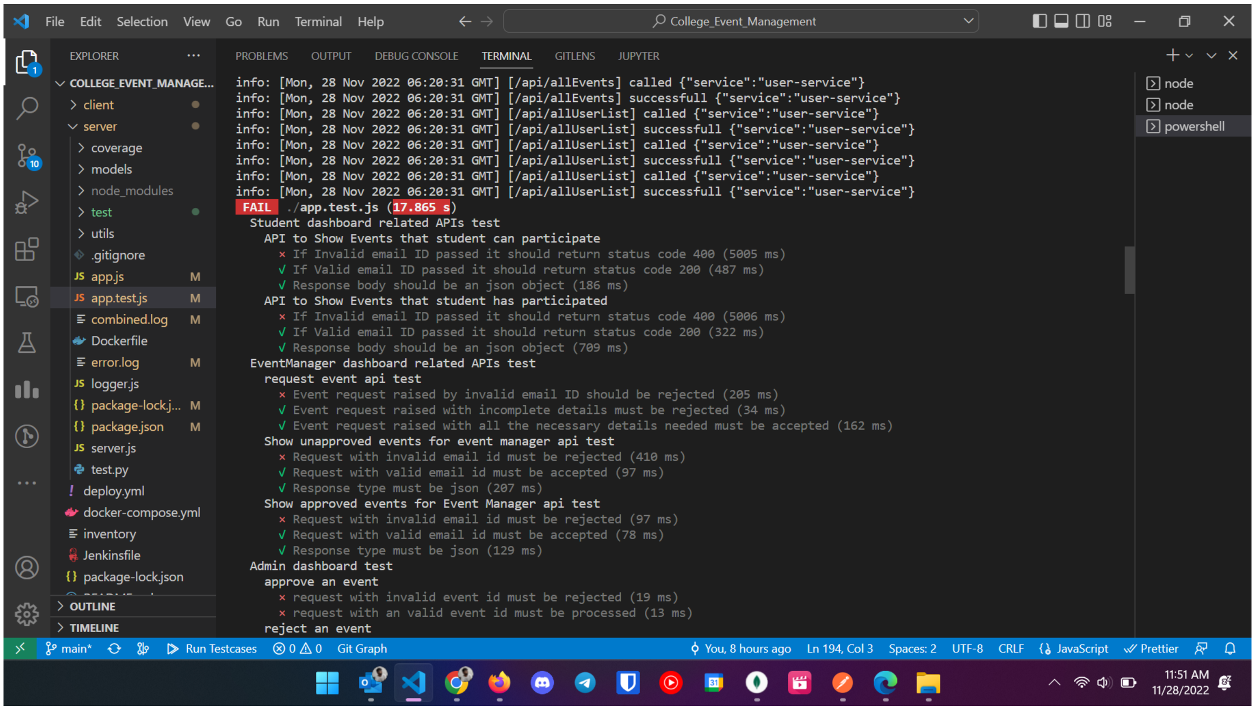
Screenshots of testing output:



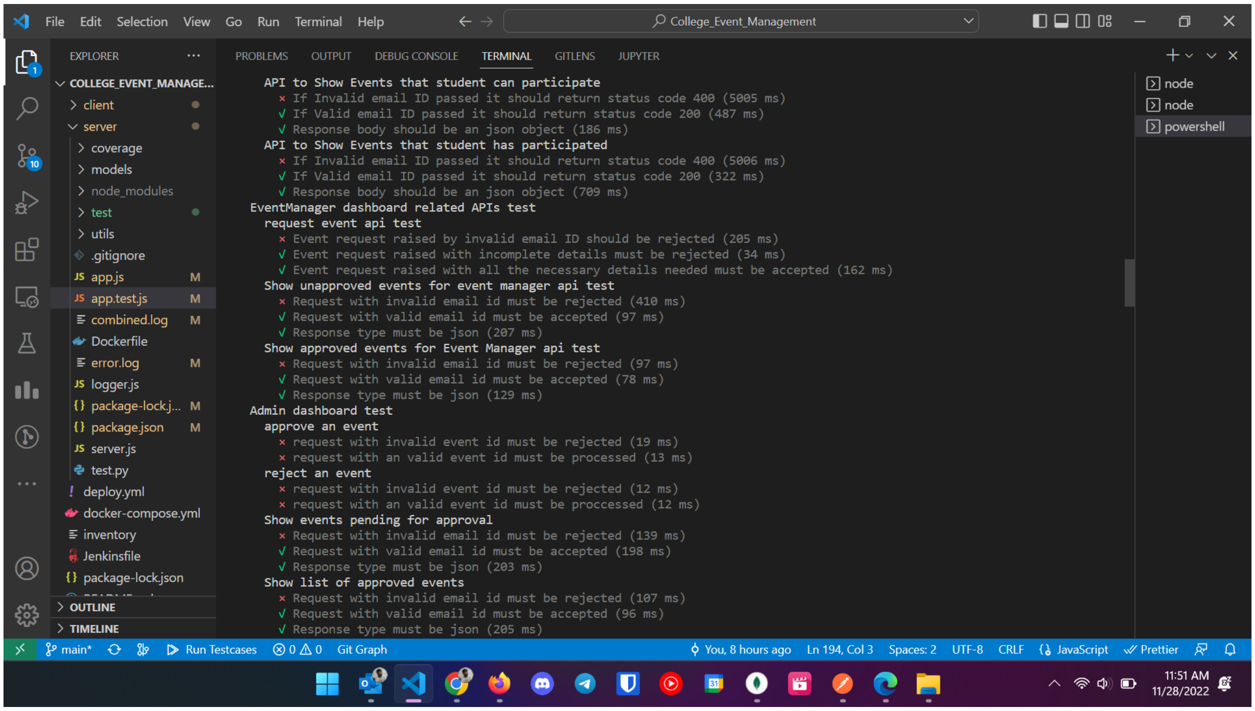
```
PS C:\Users\buddh\Desktop\Testing Project\College_Event_Management\server> npm test

> server@1.0.0 test
> jest  --forceExit

info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/showStudentsEvents] called {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] DB CONNECTED {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/showStudentsEvents] called {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/showStudentsEvents] successfull {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/showStudentsEvents] called {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/showStudentsEvents] successfull {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/showStudentParticipatedEvents] called {"service":"user-service"}
  console.log
    Un

      at log (app.js:213:13)

info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/showStudentParticipatedEvents] called {"service":"user-service"}
  console.log
    Un

      at log (app.js:213:13)

  console.log
    [ '6383a91c4231e57c0b537a89', '6383a9464231e57c0b537a8e' ]

      at log (app.js:222:13)

  console.log
    pushed

      at log (app.js:235:17)

  console.log
    pushed
```



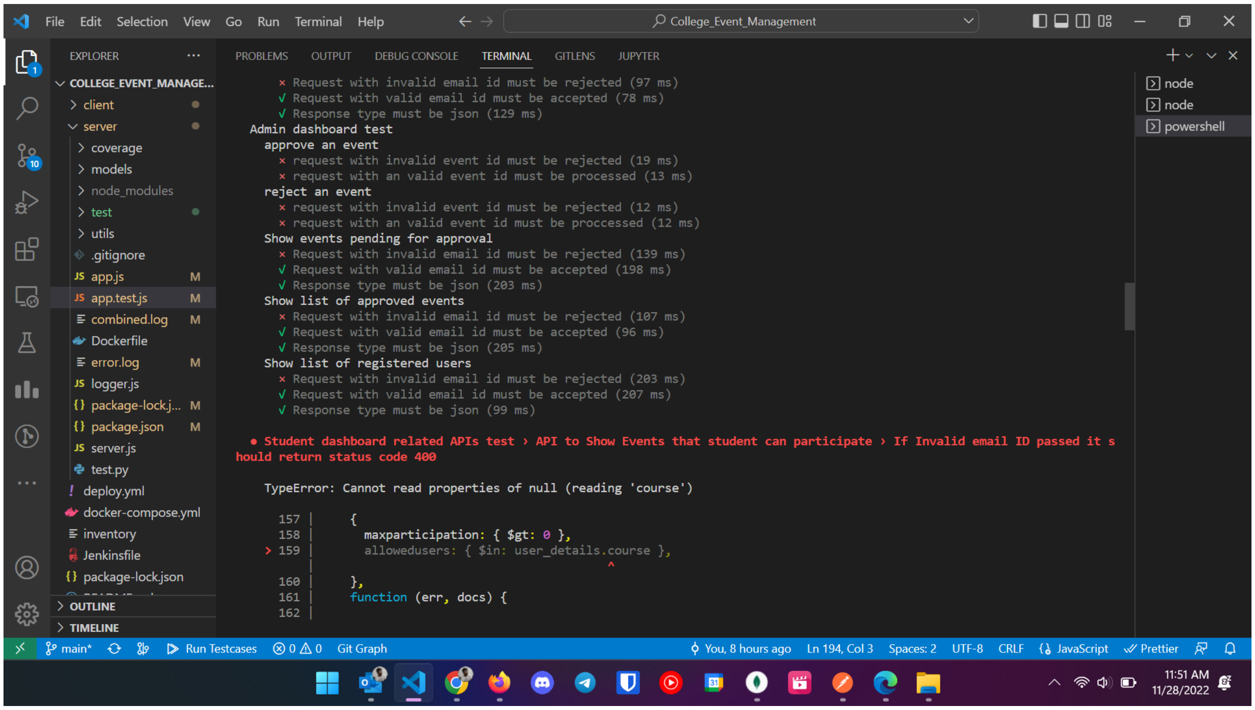
```
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/requestEvent] called {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/requestEvent] successfull {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/showAllEMEvents] called {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/showAllEMEvents] successfull {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/showAllEMEvents] called {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/showAllEMEvents] successfull {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/showAllEMEvents] called {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/showAllEMEvents] successfull {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/showAllPendingEvents] called {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/showAllPendingEvents] successfull {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/requestEvent] called {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/requestEvent] successfull {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/requestEvent] called {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/requestEvent] successfull {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/allEvents] called {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/allEvents] successfull {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/allEvents] called {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/allEvents] successfull {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/allEvents] called {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/allEvents] successfull {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/allUserList] called {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/allUserList] successfull {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/allUserList] called {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/allUserList] successfull {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/allUserList] called {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/allUserList] successfull {"service":"user-service"}
 FAIL  ./app.test.js (17.865 s)
  Student dashboard related APIs test
    API to Show Events that student can participate
      × If Invalid email ID passed it should return status code 400 (5005 ms)
      √ If Valid email ID passed it should return status code 200 (487 ms)
      √ Response body should be an json object (186 ms)
    API to Show Events that student has participated
      × If Invalid email ID passed it should return status code 400 (5006 ms)
      √ If Valid email ID passed it should return status code 200 (322 ms)
      √ Response body should be an json object (709 ms)
```



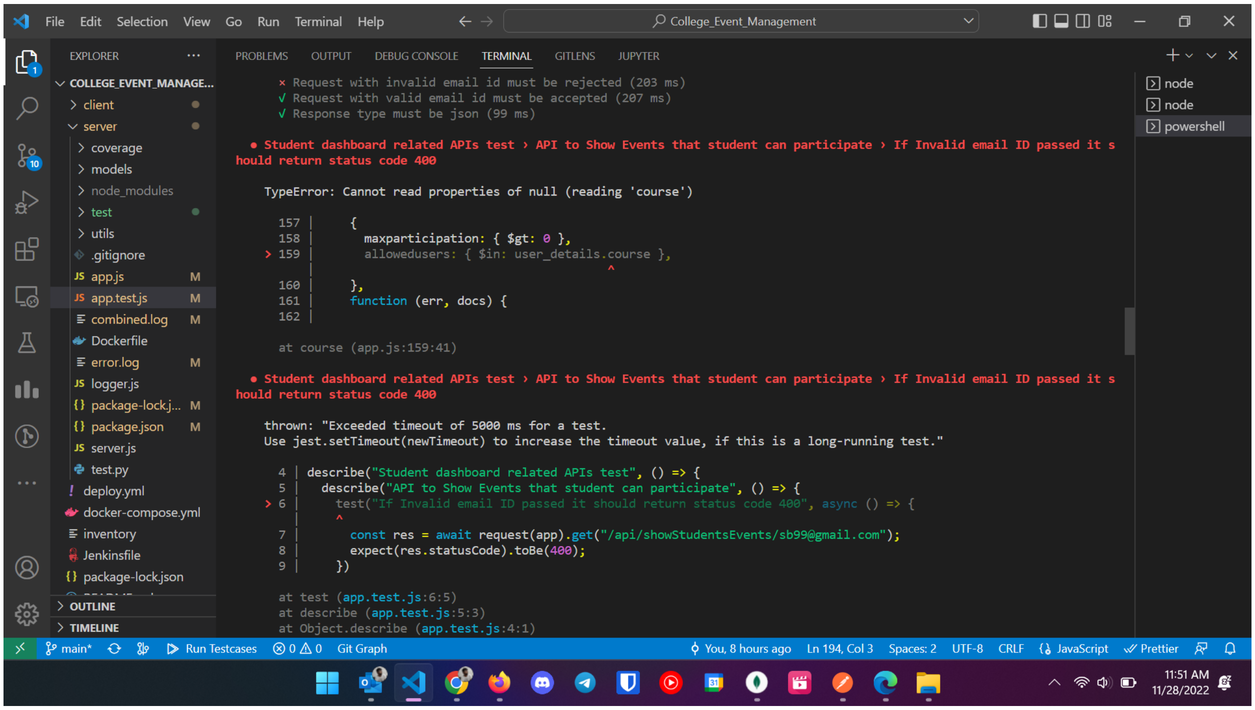
```
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/allEvents] called {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/allEvents] successfull {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/allUserList] called {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/allUserList] successfull {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/allUserList] called {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/allUserList] successfull {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/allUserList] called {"service":"user-service"}
info: [Mon, 28 Nov 2022 06:20:31 GMT] [/api/allUserList] successfull {"service":"user-service"}
 FAIL  ./app.test.js (17.865 s)
  Student dashboard related APIs test
    API to Show Events that student can participate
      × If Invalid email ID passed it should return status code 400 (5005 ms)
      √ If Valid email ID passed it should return status code 200 (487 ms)
      √ Response body should be an json object (186 ms)
    API to Show Events that student has participated
      × If Invalid email ID passed it should return status code 400 (5006 ms)
      √ If Valid email ID passed it should return status code 200 (322 ms)
      √ Response body should be an json object (709 ms)
  EventManager dashboard related APIs test
    request event api test
      × Event request raised by invalid email ID should be rejected (205 ms)
      √ Event request raised with incomplete details must be rejected (34 ms)
      √ Event request raised with all the necessary details needed must be accepted (162 ms)
    Show unapproved events for event manager api test
      × Request with invalid email id must be rejected (410 ms)
      √ Request with valid email id must be accepted (97 ms)
      √ Response type must be json (207 ms)
    Show approved events for Event Manager api test
      × Request with invalid email id must be rejected (97 ms)
      √ Request with valid email id must be accepted (78 ms)
      √ Response type must be json (129 ms)
  Admin dashboard test
    approve an event
      × request with invalid event id must be rejected (19 ms)
      × request with an valid event id must be processed (13 ms)
    reject an event
```



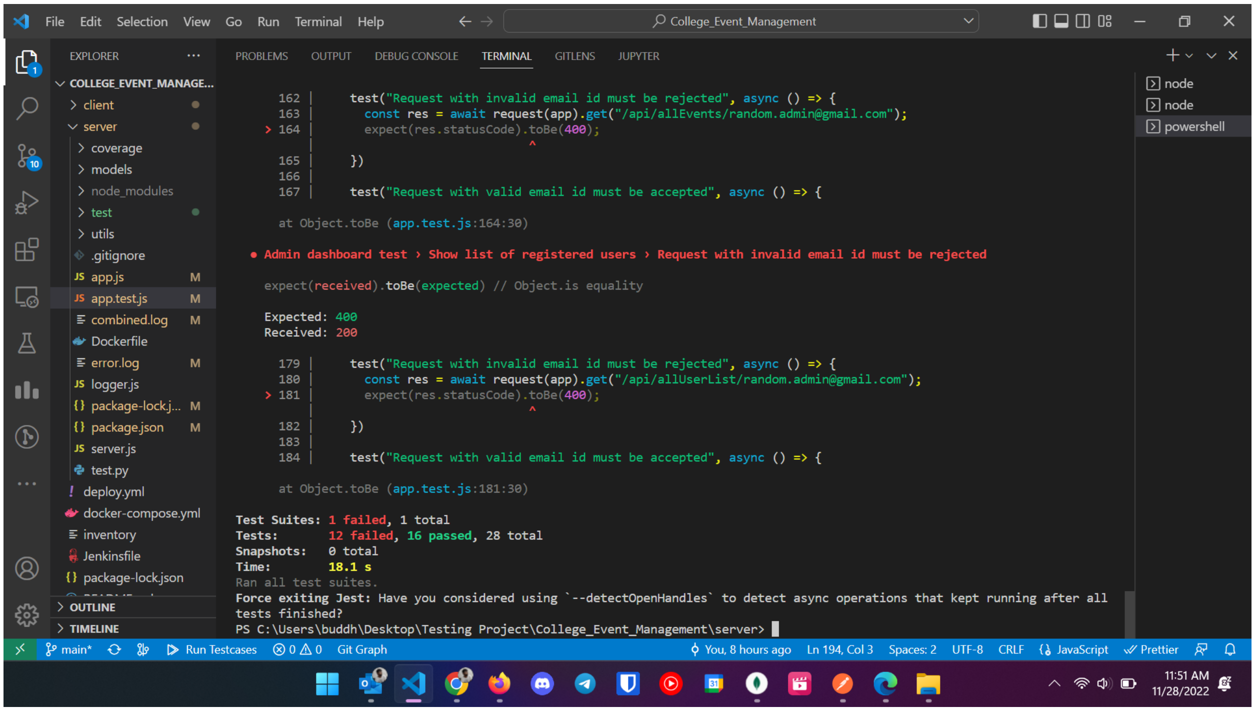
```
    API to Show Events that student can participate
      × If Invalid email ID passed it should return status code 400 (5005 ms)
      √ If Valid email ID passed it should return status code 200 (487 ms)
      √ Response body should be an json object (186 ms)
    API to Show Events that student has participated
      × If Invalid email ID passed it should return status code 400 (5006 ms)
      √ If Valid email ID passed it should return status code 200 (322 ms)
      √ Response body should be an json object (709 ms)
  EventManager dashboard related APIs test
    request event api test
      × Event request raised by invalid email ID should be rejected (205 ms)
      √ Event request raised with incomplete details must be rejected (34 ms)
      √ Event request raised with all the necessary details needed must be accepted (162 ms)
    Show unapproved events for event manager api test
      × Request with invalid email id must be rejected (410 ms)
      √ Request with valid email id must be accepted (97 ms)
      √ Response type must be json (207 ms)
    Show approved events for Event Manager api test
      × Request with invalid email id must be rejected (97 ms)
      √ Request with valid email id must be accepted (78 ms)
      √ Response type must be json (129 ms)
  Admin dashboard test
    approve an event
      × request with invalid event id must be rejected (19 ms)
      × request with an valid event id must be processed (13 ms)
    reject an event
      × request with invalid event id must be rejected (12 ms)
      × request with an valid event id must be proccessed (12 ms)
    Show events pending for approval
      × Request with invalid email id must be rejected (139 ms)
      √ Request with valid email id must be accepted (198 ms)
      √ Response type must be json (203 ms)
    Show list of approved events
      × Request with invalid email id must be rejected (107 ms)
      √ Request with valid email id must be accepted (96 ms)
      √ Response type must be json (205 ms)
```



```
      × Request with invalid email id must be rejected (97 ms)
      √ Request with valid email id must be accepted (78 ms)
      √ Response type must be json (129 ms)
  Admin dashboard test
    approve an event
      × request with invalid event id must be rejected (19 ms)
      × request with an valid event id must be processed (13 ms)
    reject an event
      × request with invalid event id must be rejected (12 ms)
      × request with an valid event id must be proccessed (12 ms)
    Show events pending for approval
      × Request with invalid email id must be rejected (139 ms)
      √ Request with valid email id must be accepted (198 ms)
      √ Response type must be json (203 ms)
    Show list of approved events
      × Request with invalid email id must be rejected (107 ms)
      √ Request with valid email id must be accepted (96 ms)
      √ Response type must be json (205 ms)
    Show list of registered users
      × Request with invalid email id must be rejected (203 ms)
      √ Request with valid email id must be accepted (207 ms)
      √ Response type must be json (99 ms)

  ● Student dashboard related APIs test › API to Show Events that student can participate › If Invalid email ID passed it s
hould return status code 400

    TypeError: Cannot read properties of null (reading 'course')

      157 |         {
      158 |           maxparticipation: { $gt: 0 },
    > 159 |           allowedusers: { $in: user_details.course },
          |                                             ^
      160 |         },
      161 |         function (err, docs) {
      162 |
```



```
      × Request with invalid email id must be rejected (203 ms)
      √ Request with valid email id must be accepted (207 ms)
      √ Response type must be json (99 ms)

  ● Student dashboard related APIs test › API to Show Events that student can participate › If Invalid email ID passed it s
hould return status code 400

    TypeError: Cannot read properties of null (reading 'course')

      157 |     {
      158 |       maxparticipation: { $gt: 0 },
    > 159 |       allowedusers: { $in: user_details.course },
          |                                         ^
      160 |     },
      161 |     function (err, docs) {
      162 |

      at course (app.js:159:41)

  ● Student dashboard related APIs test › API to Show Events that student can participate › If Invalid email ID passed it s
hould return status code 400

    thrown: "Exceeded timeout of 5000 ms for a test.
    Use jest.setTimeout(newTimeout) to increase the timeout value, if this is a long-running test."

      4 | describe("Student dashboard related APIs test", () => {
      5 |   describe("API to Show Events that student can participate", () => {
    > 6 |     test("If Invalid email ID passed it should return status code 400", async () => {
        |     ^
      7 |       const res = await request(app).get("/api/showStudentsEvents/sb99@gmail.com");
      8 |       expect(res.statusCode).toBe(400);
      9 |     })

      at test (app.test.js:6:5)
      at describe (app.test.js:5:3)
      at Object.describe (app.test.js:4:1)
```



```
      162 |     test("Request with invalid email id must be rejected", async () => {
      163 |       const res = await request(app).get("/api/allEvents/random.admin@gmail.com");
    > 164 |       expect(res.statusCode).toBe(400);
          |                              ^
      165 |     })
      166 |
      167 |     test("Request with valid email id must be accepted", async () => {

      at Object.toBe (app.test.js:164:30)

  ● Admin dashboard test › Show list of registered users › Request with invalid email id must be rejected

    expect(received).toBe(expected) // Object.is equality

    Expected: 400
    Received: 200

      179 |     test("Request with invalid email id must be rejected", async () => {
      180 |       const res = await request(app).get("/api/allUserList/random.admin@gmail.com");
    > 181 |       expect(res.statusCode).toBe(400);
          |                              ^
      182 |     })
      183 |
      184 |     test("Request with valid email id must be accepted", async () => {

      at Object.toBe (app.test.js:181:30)

Test Suites: 1 failed, 1 total
Tests:       12 failed, 16 passed, 28 total
Snapshots:   0 total
Time:        18.1 s
Ran all test suites.
Force exiting Jest: Have you considered using `--detectOpenHandles` to detect async operations that kept running after all
tests finished?
PS C:\Users\buddh\Desktop\Testing Project\College_Event_Management\server>
```

Individual contribution:

Shreyank Buddhadev(MT2021130)

- Tested Admin and Event manager Dashboard
- ATG Diagram

Biswadeep Das(MT2021032)

- Tested Student dashboard and other miscellaneous test cases.
- ATG Diagram