IT314: Software Engineering

Course Project Lab 8

Group: 15 Conference Management System

IDs	Name
202001117	PARVA BHARATKUMAR DHAMI
202001122	MONPARA SUMIT RAJESHBHAI
202001128	MEGHABEN RATHWA
202001132	SONAGARA SARTHAK MOHANBHAI
202001154	METKEL HARSHKUMAR ANANDBHAI
202001162	PADHIYAR KARAN HARESHBHAI
202001172	PATEL VEDANT HARSHADBHAI
202001176	MANGUKIYA JENISH MAHESHBHAI
202001177	PRAJAPATI PARTH SURESHBHAI

Testing for User:

Sr no.	Test cases	Expected outcome	Actual outcome

1	`email: "shreeji@gmail.com",`
	`password: "Shreeji@01",`
	`role: "admin",`
		Valid login	Valid login

2	Wrong role
	`email: "shreeji@gmail.com",`
	`password: "Shreeji@01",`
	`role: "attendee",`
		Invalid login	Invalid login

3	`username: "harsh@test",`
	`email: "harsh@test.com",`
	`password: "Harsh@test1",`
	`role: "attendee",`
		Valid signup	Valid signup

4	Invalid email format
	`username: "harsh@test",`
	`email: "harshtest",`
	`password: "Harsh@test1",`
	`role: "attendee",`
		Invalid signup	Invalid signup

5	`email: "harsh@test.com",` `password: "Harsh@test2",`	Valid update	Valid update

6	Wrong email
	`email: "harsh@testttttt.com",`
	`password: "Harsh@test2",`
		Invalid update	Invalid update

7	`email: "harsh@test.com",`	Valid delete	Valid delete

Content of {User.test.js}

```
process.env.NODE_ENV = "test";
const chai = require("chai");
const chaiHttp = require("chai-http");
const app = require("../server");
const expect = chai.expect;
chai.use(chaiHttp);

// test case 1: valid admin login
-----------------------------------------

describe("User", () => {
  // login module
  describe("POST /auth/user/login", () => {
    it("it should login a user", (done) => {
      chai
        .request(app)
        .post("/auth/user/login")
        .send({
          email: "shreeji@gmail.com",
          password: "Shreeji@01",
          role: "admin",
        })
        .end((err, res) => {
          expect(res).to.have.status(200);
          expect(res.body).to.be.a("object");
          expect(res.body).to.have.property("email");
          expect(res.body).to.have.property("role");
          expect(res.body).to.have.property("token");
          done();
        });
    });
  });
});
```

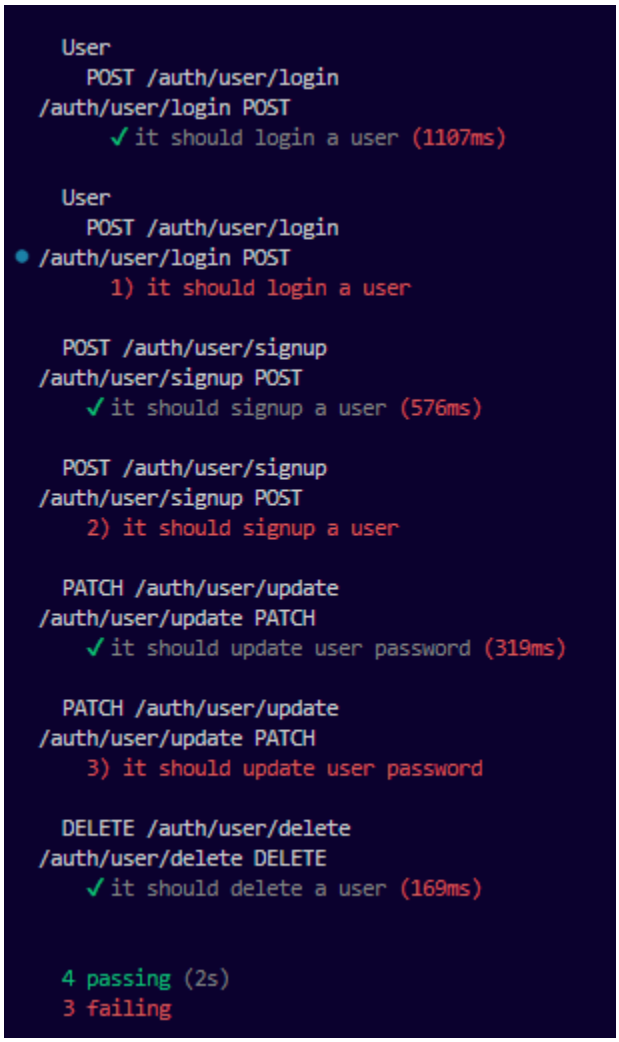
```
// test case 2: invalid role login
-----------------------------------------

describe("User", () => {
  // login module
  describe("POST /auth/user/login", () => {
    it("it should login a user", (done) => {
      chai
        .request(app)
        .post("/auth/user/login")
        .send({
          email: "shreeji@gmail.com",
          password: "Shreeji@01",
          role: "attendee",         // invalid role, role changed to
attendee
        })
        .end((err, res) => {
          expect(res).to.have.status(200);
          expect(res.body).to.be.a("object");
          expect(res.body).to.have.property("email");
          expect(res.body).to.have.property("role");
          expect(res.body).to.have.property("token");
          done();
        });
    });
  });
});

// test case 3: valid signup ----------------------------------------

  describe("POST /auth/user/signup", () => {
    it("it should signup a user", (done) => {
      chai
        .request(app)
        .post("/auth/user/signup")
        .send({
          username: "harsh@test",
```

```
          email: "harsh@test.com",
          password: "Harsh@test1",
          role: "attendee",
        })
        .end((err, res) => {
          expect(res).to.have.status(201);
          expect(res.body).to.be.a("object");
          expect(res.body).to.have.property("email");
          expect(res.body).to.have.property("role");
          expect(res.body).to.have.property("token");
          done();
        });
    });
  });

// test case 4: invalid signup ----------------------------------------

  describe("POST /auth/user/signup", () => {
    it("it should signup a user", (done) => {
      chai
        .request(app)
        .post("/auth/user/signup")
        .send({
          username: "harsh@test",
          email: "harshtest",          // invalid email address
          password: "Harsh@test1",
          role: "attendee",
        })
        .end((err, res) => {
          expect(res).to.have.status(201);
          expect(res.body).to.be.a("object");
          expect(res.body).to.have.property("email");
          expect(res.body).to.have.property("role");
          expect(res.body).to.have.property("token");
          done();
        });
    });
```

```
  });

// test case 5: valid update ----------------------------------------

  describe("PATCH /auth/user/update", () => {
    it("it should update user password", (done) => {
      chai
        .request(app)
        .patch("/auth/user/update")
        .send({
          email: "harsh@test.com",
          password: "Harsh@test2",
        })
        .end((err, res) => {
          expect(res).to.have.status(200);
          expect(res.body).to.be.a("object");
          expect(res.body).to.have.property("user");
          done();
        });
    });
  });

// test case 6: invalid update ----------------------------------------

describe("PATCH /auth/user/update", () => {
  it("it should update user password", (done) => {
    chai
      .request(app)
      .patch("/auth/user/update")
      .send({
        email: "harsh@testtttt.com",      // wrong email address
        password: "Harsh@test2",
      })
      .end((err, res) => {
        expect(res).to.have.status(200);
        expect(res.body).to.be.a("object");
        expect(res.body).to.have.property("user");
```

```
        done();
      });
  });
});

// test case 7: valid delete ----------------------------------------

  describe("DELETE /auth/user/delete", () => {
    it("it should delete a user", (done) => {
      chai
        .request(app)
        .delete("/auth/user/delete")
        .send({
          email: "harsh@test.com",
        })
        .end((err, res) => {
          expect(res).to.have.status(200);
          expect(res.body).to.be.a("object");
          expect(res.body).to.have.property("user");
          done();
        });
    });
  });
```

Result of test cases and execution time:



We got all the test cases evaluated correctly. Our frontend & backend will verify the email & password details accurately and will notify the user accordingly.

Testing for Organization:

Sr no.	Test cases	Expected outcome	Actual outcome
1	`email: "org1@abc.com",` `password: "Org@1234",`	Valid login	Invalid login

2	`orgname: "ABC",`
	`email: "org1@abc.com",`
	`password: "Org@1234",`
		Valid signup	Valid signup

3	`.get("/auth/org")`	Get all org	Invalid output
4	`email: "org1@abc.com",`	Valid delete	Valid delete

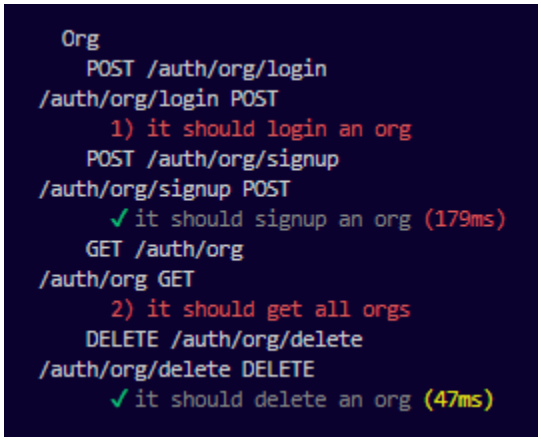
Content of {Org.test.js}

```
process.env.NODE_ENV = "test";
const chai = require("chai");
const chaiHttp = require("chai-http");
const app = require("../server");
const { describe } = require("mocha");
const expect = chai.expect;
chai.use(chaiHttp);

describe("Org", () => {
  // login org
  describe("POST /auth/org/login", () => {
    it("it should login an org", (done) => {
      chai
        .request(app)
        .post("/auth/org/login")
        .send({
          email: "org1@abc.com",
```

```
        password: "Org@1234",
      })
      .end((err, res) => {
        expect(res).to.have.status(200);
        expect(res.body).to.be.a("object");
        expect(res.body).to.have.property("email");
        done();
      });
  });
});

// signup org
describe("POST /auth/org/signup", () => {
  it("it should signup an org", (done) => {
    chai
      .request(app)
      .post("/auth/org/signup")
      .send({
        orgname: "ABC",
        email: "org1@abc.com",
        password: "Org@1234",
      })
      .end((err, res) => {
        expect(res).to.have.status(201);
        expect(res.body).to.be.a("object");
        expect(res.body).to.have.property("email");
        done();
      });
  });
});

// get all orgs
describe("GET /auth/org", () => {
  it("it should get all orgs", (done) => {
    chai
      .request(app)
      .get("/auth/org")
```

```
        .end((err, res) => {
          expect(res).to.have.status(200);
          expect(res.body).to.be.a("array");
          done();
        });
    });
  });

  // delete org
  describe("DELETE /auth/org/delete", () => {
    it("it should delete an org", (done) => {
      chai
        .request(app)
        .delete("/auth/org/delete")
        .send({
          email: "org1@abc.com",
        })
        .end((err, res) => {
          expect(res).to.have.status(200);
          expect(res.body).to.be.a("object");
          expect(res.body).to.have.property("org");
          done();
        });
    });
  });
});
```

Result of test cases and execution time:



We got to know about 2 invalid errors due to test cases. There are some implementation issues for them, which will be solved at the earliest by the team.