Master Team Project WS2021

Milestone 4: HelpMeLearn

Master Team Project Winter 2021

Global Distributed Software Development

CEO & CTO:

Prof. Dr. Rainer Todtenhöfer

Team 04

Hasib Iqbal (hasib.iqbal@informatik.hs-fulda.de)	:	Team Lead | Frontend Developer
Mohammad Rakibul Hasan	:	Frontend Lead
Rohat Sagar Urif Sonu	:	Frontend Developer
Mohammad Salman Haydar	:	Backend Lead
Talha Jahangiri Khan	:	Github Lead | Backend Developer
Chowdhury Amlan Barua	:	Cloud & Backend Developer
Nisha Devi	:	Backend Developer

Revision	Date
Version 1.0	11 March 2022

11 March 2022

Table of Contents

Product Summary

Motivation & Importance

Humanity has become accustomed to clicks as the most expedient method of obtaining what one desires. In this age of clicks, we want to make life easy for students and teachers alike. However, we have chosen to focus on one of everyone's most basic requirements, which is learning. HelpMeLearn is a web-based tutoring service that connects students and teachers.

Functions & Services

Our product aims to establish a communication bridge between tutor and students. From the perspective of students, they can search for tutors for their desired subjects using our platform and also can communicate with them. Our platform provides real time chat facilities which can help the students to get connected with the tutors instantly. They can also share their experience about tutors through our review system. From the perspective of the tutor, they can upload their CV, qualification and show their skills through our tutor profile page to get the interests of the students. They can mention their hourly salary expectations, reply to messages from the students using our platform. Also, every change made by the tutor in their timeline is reviewed and approved by the admin before going live to maintain the standard of our platform. Our platform also provides a guest page where a guest user can search for their desired subjects and tutors without registering. The only catch here is that they cannot communicate with the tutor. Our searching page also provides facilities about filtering different categories like by subject, gender, etc.

Reasons to use our HelpMeLearn Application

Using our HelpMeLearn application for learning will make students and teachers connect in the most efficient way without having to pay a great sum of money for subscription and we are ensuring a safe platform for both users. If you are a student or teacher, it creates a safe platform for a teacher and a student to connect as the opportunity to offer courses by a teacher is endless and students can easily compare the offered courses.

Major Committed Functions:

1. Search courses

- Allowing/Enabling the students to find certain approved courses that satisfy / fulfill a criterion.

2. Filter searched courses

- Specific attributes a student can use to refine the search results or course. e.g., by gender, price range etc.

3. Add /View Reviews

- Allowing/Enabling the students to view reviews of teachers that can help students find suitable teachers

4. Registration

- Allowing the students and teachers to make a new account profile for themselves.

5. Login

- Allowing the students and teachers to log in to their account or dashboard.

6. Post Courses

- Allowing the approved teachers to add new courses from their profiles.

7. Messaging

- Allowing the students and teachers to communicate with each other through messages and also enables them to see the previous message history as well from their dashboard.

8. Approve/Reject courses / teacher information

- Allowing the admins to approve / reject different pending courses or personal information of teachers from their dashboard.

Usability Test Plan

This document describes a test plan for conducting a usability test during the development of HelpMeLearn. The goals of usability testing include establishing a baseline of user performance, establishing, and validating user performance measures, and identifying potential design concerns to be addressed to improve the efficiency, productivity, and end user satisfaction.

Test objectives

The objective of this testing is to get the user feedback and user experience about searching for a specific tutor using searched keywords, subject filter or choosing specific subject level.

Test background and setup

- **System setup**

  HelpMeLearn website have been published on AWS to be tested by participants using the following technologies

  • Operating System: Ubuntu 18.04 Server
  • Database: MySQL v: 8.0
  • Web Server: NGINX 1.18
  • Server-Side Language: Node.js

- **Starting point**

  Participants will test HelpMeLearn website using their laptops or smartphones, and will focus on the search functionality in the home page

- **The intended users**

  The test will be conducted with a targeted category of users, which are the students and tutors of Fulda university. Therefore, we will ask for users of the university to test the website

- **URL of the system to be tested**

  HelpMeLearn (the link may change due to redeploying)

- **What is to be measured**

  We will be focusing on user satisfaction

Usability Testing Phase:

<u>**Test Participant 1:**</u>

Phase 1: Screening and Pretest:

Q1. How old are you?
24

Q2. What is the highest level of education you've completed?
Bachelor's Degree

Q3. What is your current occupation?
Student(Pursuing Masters)

Q4. On a scale of 1 to 5 how would you rate your level of confidence in using your laptop/pc for looking for a tutor online?
5

Q5. When was the last time you looked for a tutoring service online?
In 2020

Phase: 2 Usability Task description

Task 1: Imagine that you are facing difficulty in studying statistics and want to get help from an expert.

Task 2: Imagine that your exams are near and facing difficulty in solving exercises of economics

Task 3: Imagine you are looking for a tutor in fulda and worried about feedback from other students.

Phase: 3 Post Test Questions

Q1. How was your overall experience when searching for a tutor?
Good

Q2. How simple and clean was the interface?
It is very simple.

Q3. Can you tell me what you think about filtering the tutor by subject name and level?
I tried multiple times and it worked without any issues.

Q4. How was your experience looking for tutor qualifications and feedback?
Simple and clear.

Sl. No	**Task Objective**	**Scale 0/5 (0-OK, 5-Good)**
Task 1	Search for a tutor online by subject name	4/5
Task 2	Search for a tutor online by subject level	4/5
Task 3	Looking for tutor feedback	3/5

Test Participant 2:

Phase 1: Screening and Pretest:

Q1. How old are you?
19

Q2. What is the highest level of education you've completed?
College

Q3. What is your current occupation?
Student(Pursuing Bachelors)

Q4. On a scale of 1 to 5 how would you rate your level of confidence in using your laptop/pc for looking for a tutor online?
5

Q5. When was the last time you looked for a tutoring service online?
In 2020

Phase: 2 Usability Task description

Task 1: Imagine that you are facing difficulty in studying statistics and want to get help from an expert.

Task 2: Imagine that your exams are near and facing difficulty in solving exercises of economics

Task 3: Imagine you are looking for a tutor in fulda and worried about feedback from other students.

Phase: 3 Post Test Questions

Q1. How was your overall experience when searching for a tutor?
Good and simple.

Q2. How simple and clean was the interface?
It is very simple and easy to use.

Q3. Can you tell me what you think about filtering the tutor by subject name and level?
Filters are clear and easy to use.

Q4. How was your experience looking for tutor qualifications and feedback?
Clearly mentioned.

Sl. No	**Task Objective**	**Scale 0/5 (0-OK, 5-Good)**
Task 1	Search for a tutor online by subject name	4/5
Task 2	Search for a tutor online by subject level	3/5
Task 3	Looking for tutor feedback	4/5

<u>**Test Participant 3:**</u>

Phase 1: Screening and Pretest:

Q1. How old are you?
21

Q2. What is the highest level of education you've completed?
University

Q3. What is your current occupation?
Student(Pursuing Bachelors in Natural Science)

Q4. On a scale of 1 to 5 how would you rate your level of confidence in using your laptop/pc for looking for a tutor online?
4

Q5. When was the last time you looked for a tutoring service online?
In 2019

Phase: 2 Usability Task description

Task 1: Imagine that you are facing difficulty in studying statistics and want to get help from an expert.

Task 2: Imagine that your exams are near and facing difficulty in solving exercises of economics

Task 3: Imagine you are looking for a tutor in fulda and worried about feedback from other students.

Phase: 3 Post Test Questions

Q1. How was your overall experience when searching for a tutor?
Great and amazing.

Q2. How simple and clean was the interface?
Interface is very easy to use. And I did not have any issue adjusting to the interface.

Q3. Can you tell me what you think about filtering the tutor by subject name and level?
Filters are clear and easy to use. However, I would appreciate more filtering options.

Q4. How was your experience looking for tutor qualifications and feedback?
Clearly mentioned.

Sl. No	**Task Objective**	**Scale 0/5 (0-OK, 5-Good)**
Task 1	Search for a tutor online by subject name	4/5
Task 2	Search for a tutor online by subject level	4/5
Task 3	Looking for tutor feedback	3/5

QA Test Plan

Test Objective:

The test objectives are to verify the functionality of website HelpMeLearn and to ensure that the software is as per business requirements by identifying any bugs or issues and fixing them before release.

Hardware and Software setup:

Laptop or smartphone using Edge, Google Chrome or Firefox browser.

Feature to be tested:

- Search By Subject Name
- Search By Subject Level
- Search By Tutor Gender

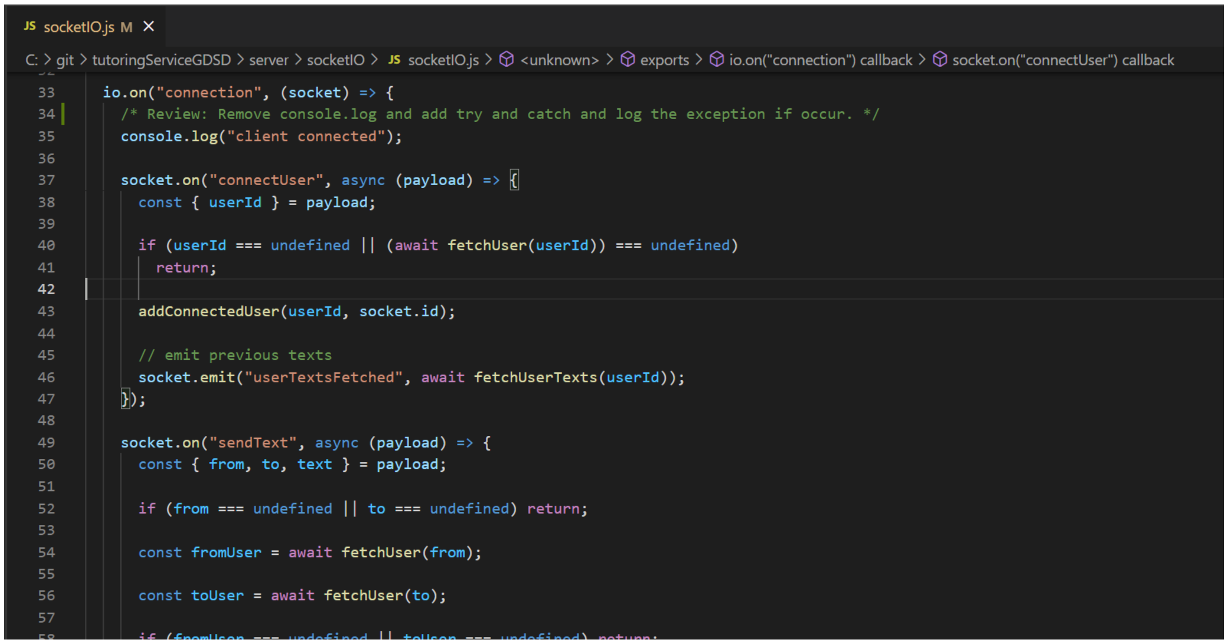
Test Case Id & Browser	**Test Title**	**Test Description**	**Test Input**	**Expected Result**	**Test Result**
TC_1 (Chrome, Firefox)	View all tutors	Students are able to view all tutors in a list view.	Login with valid credentials. Students will be navigated to the Search Page	The Search Page displays all approved tutors.	PASS
TC_2 (Chrome, Firefox)	Filter Tutors by subject name	Students are able to filter tutors by subject name.	Input '**Math101**' on **SubjectName** text field and click on the **Search** button	Students are able to see all approved tutors who teach the **Math101** subject.	PASS
TC_3 (Chrome, Firefox)	Filter Tutors by subject level.	Students are able to filter tutors by subject level.	Select **'Master'** in the subject level drop down field and click on the **Search** button.	Students are able to see all approved tutors who teach at least one subject with a master's level.	PASS
TC_4 (Chrome, Firefox)	Filter Tutors by gender.	Students are able to filter tutors by gender.	Select **'Male'** in the **Gender** drop down field and click on the **Search** button.	Students are able to see all approved tutors whose gender is male.	PASS

TC_5 (Chrome, Firefox)	Filter Tutors by subject Name, level, and gender.	Students are able to filter tutors by subject name, level and gender.	Input **‘Math101’** in the Subject Name Field, select **‘Master’** in the subject level drop down field, select **‘Male’** in the gender drop down field and click on the **Search** button.	Students are able to see all approved tutors who teach SubjectName ‘Math101’ with subject level ‘Master’ and gender is male.	PASS

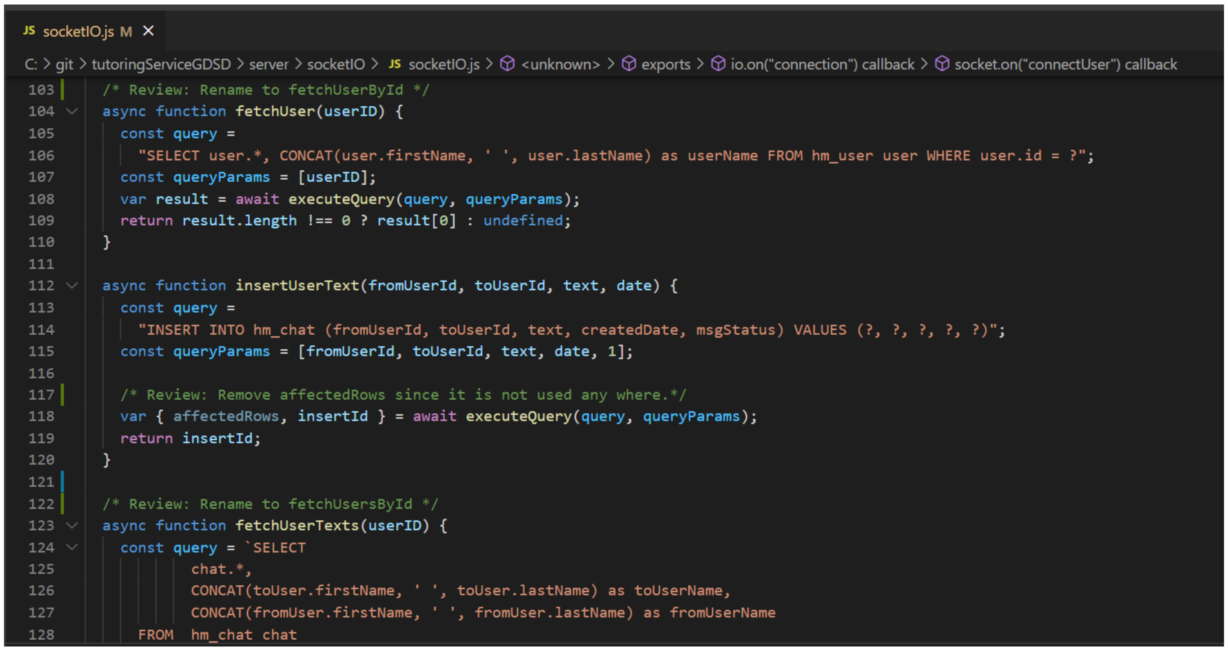
Code Review

Developer	Rohat Sagar Urif Sonu
Reviewer	Nisha Devi

File: socketIO.js

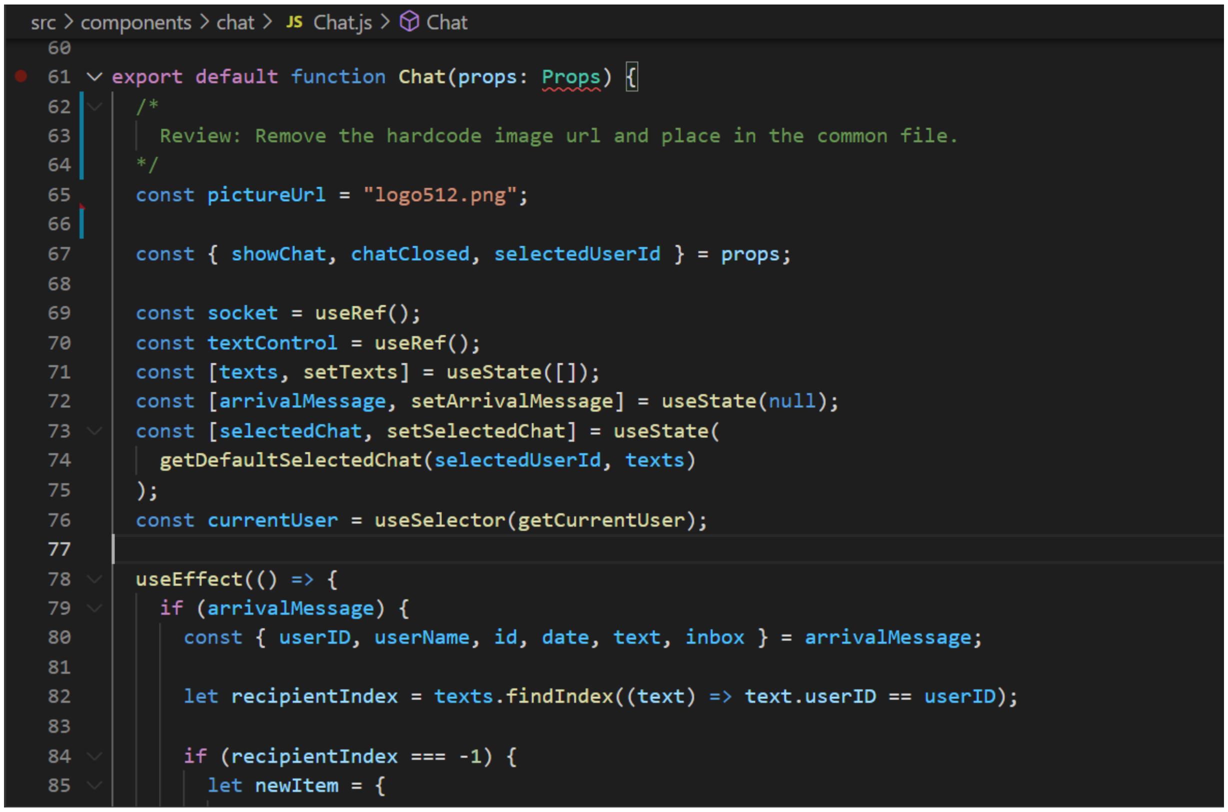

socketIO.js M
C: > git > tutoringServiceGDSD > server > socketIO > socketIO.js > <unknown> > exports > io.on("connection") callback > socket.on("connectUser") callback

```
  io.on("connection", (socket) => {
    /* Review: Remove console.log and add try and catch and log the exception if occur. */
    console.log("client connected");

    socket.on("connectUser", async (payload) => {
      const { userId } = payload;

      if (userId === undefined || (await fetchUser(userId)) === undefined)
        return;

      addConnectedUser(userId, socket.id);

      // emit previous texts
      socket.emit("userTextsFetched", await fetchUserTexts(userId));
    });

    socket.on("sendText", async (payload) => {
      const { from, to, text } = payload;

      if (from === undefined || to === undefined) return;

      const fromUser = await fetchUser(from);

      const toUser = await fetchUser(to);

      if (fromUser === undefined || toUser === undefined) return;
```



```
JS socketIO.js M ×
C: > git > tutoringServiceGDSD > server > socketIO > JS socketIO.js > <unknown> > exports > io.on("connection") callback > socket.on("connectUser") callback
/* Review: Rename to fetchUserById */
async function fetchUser(userID) {
  const query =
    "SELECT user.*, CONCAT(user.firstName, ' ', user.lastName) as userName FROM hm_user user WHERE user.id = ?";
  const queryParams = [userID];
  var result = await executeQuery(query, queryParams);
  return result.length !== 0 ? result[0] : undefined;
}

async function insertUserText(fromUserId, toUserId, text, date) {
  const query =
    "INSERT INTO hm_chat (fromUserId, toUserId, text, createdDate, msgStatus) VALUES (?, ?, ?, ?, ?)";
  const queryParams = [fromUserId, toUserId, text, date, 1];

  /* Review: Remove affectedRows since it is not used any where.*/
  var { affectedRows, insertId } = await executeQuery(query, queryParams);
  return insertId;
}

/* Review: Rename to fetchUsersById */
async function fetchUserTexts(userID) {
  const query = `SELECT
        chat.*,
        CONCAT(toUser.firstName, ' ', toUser.lastName) as toUserName,
        CONCAT(fromUser.firstName, ' ', fromUser.lastName) as fromUserName
    FROM  hm_chat chat
```

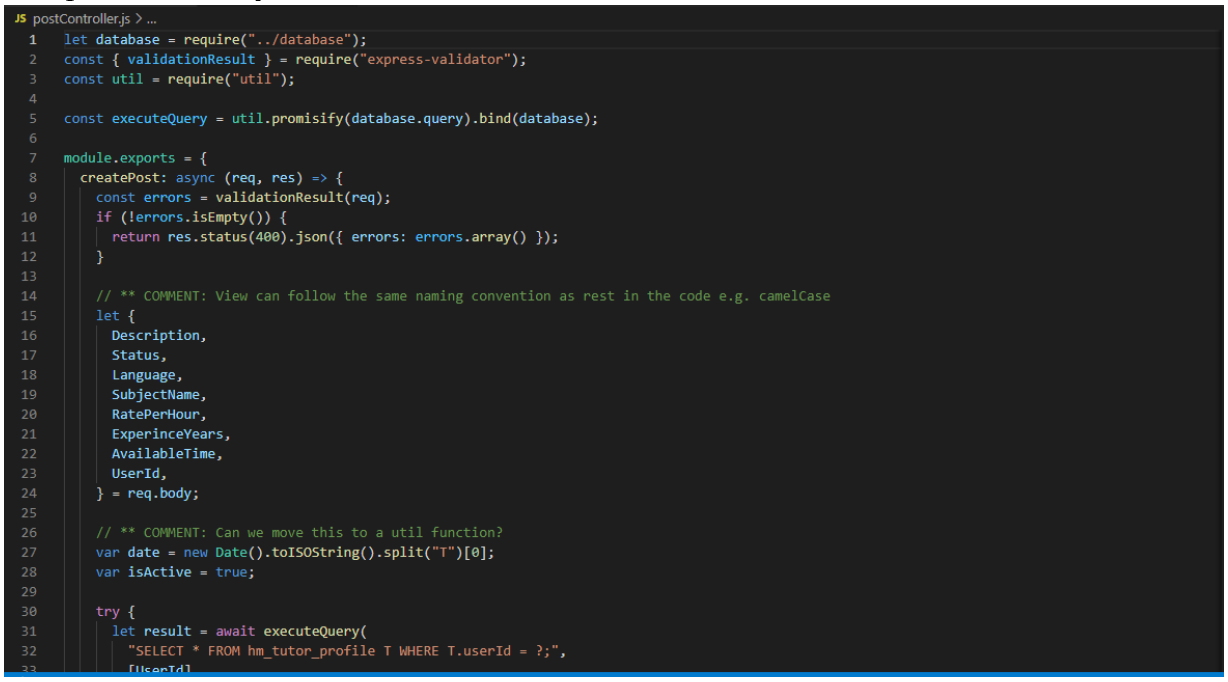
File: Chat.js



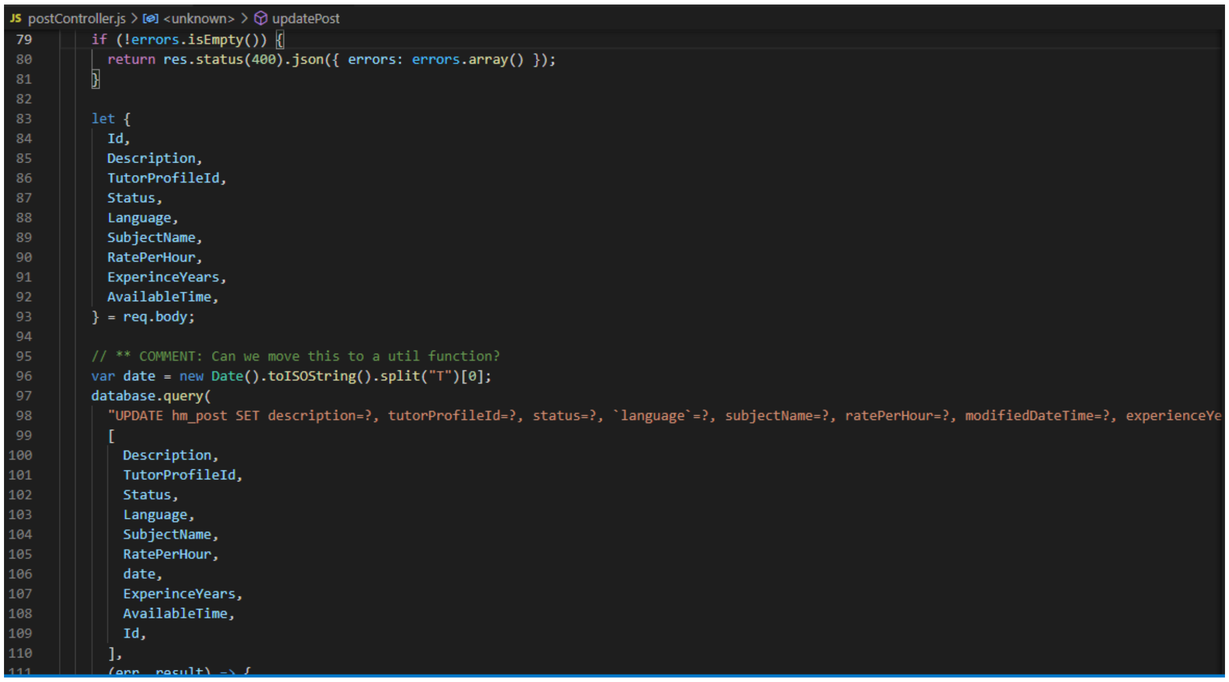
```
src > components > chat > JS Chat.js > Chat

export default function Chat(props: Props) {
  /*
    Review: Remove the hardcode image url and place in the common file.
  */
  const pictureUrl = "logo512.png";

  const { showChat, chatClosed, selectedUserId } = props;

  const socket = useRef();
  const textControl = useRef();
  const [texts, setTexts] = useState([]);
  const [arrivalMessage, setArrivalMessage] = useState(null);
  const [selectedChat, setSelectedChat] = useState(
    getDefaultSelectedChat(selectedUserId, texts)
  );
  const currentUser = useSelector(getCurrentUser);

  useEffect(() => {
    if (arrivalMessage) {
      const { userID, userName, id, date, text, inbox } = arrivalMessage;

      let recipientIndex = texts.findIndex((text) => text.userID == userID);

      if (recipientIndex === -1) {
        let newItem = {
```

Developer	Nisha Devi
Reviewer	Rohat Sagar Urif Sonu

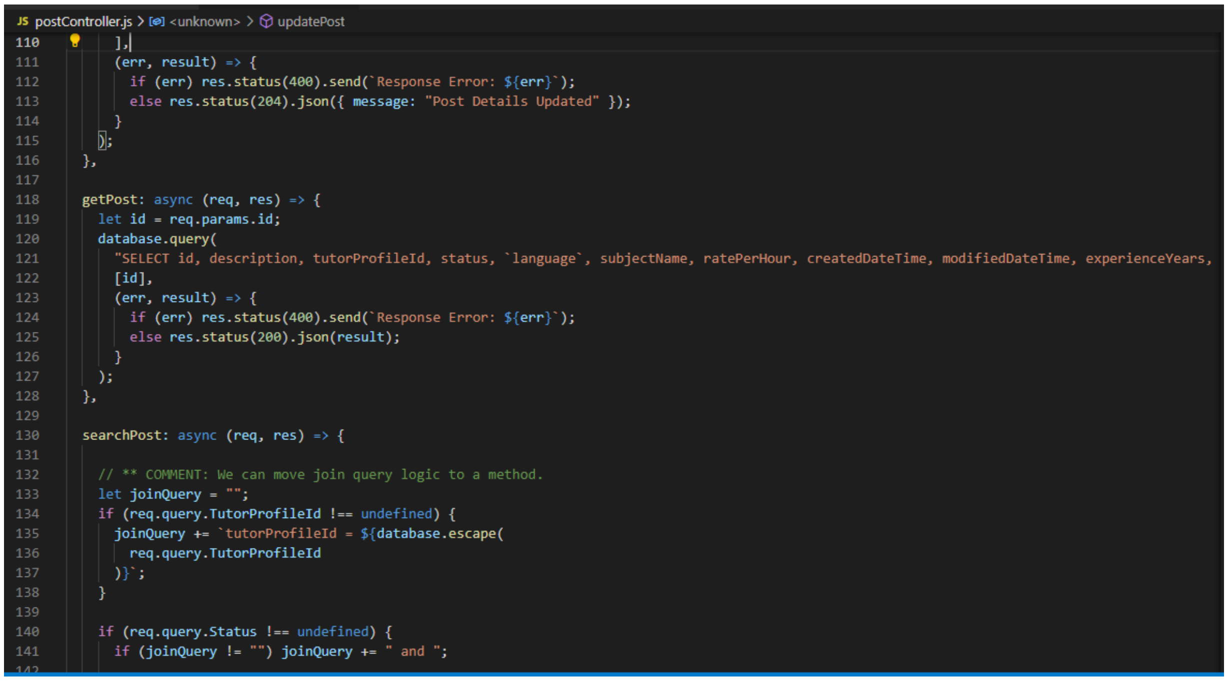
File: postController.js



```
let database = require("../database");
const { validationResult } = require("express-validator");
const util = require("util");

const executeQuery = util.promisify(database.query).bind(database);

module.exports = {
  createPost: async (req, res) => {
    const errors = validationResult(req);
    if (!errors.isEmpty()) {
      return res.status(400).json({ errors: errors.array() });
    }

    // ** COMMENT: View can follow the same naming convention as rest in the code e.g. camelCase
    let {
      Description,
      Status,
      Language,
      SubjectName,
      RatePerHour,
      ExperinceYears,
      AvailableTime,
      UserId,
    } = req.body;

    // ** COMMENT: Can we move this to a util function?
    var date = new Date().toISOString().split("T")[0];
    var isActive = true;

    try {
      let result = await executeQuery(
        "SELECT * FROM hm_tutor_profile T WHERE T.userId = ?;",
        [UserId]
```

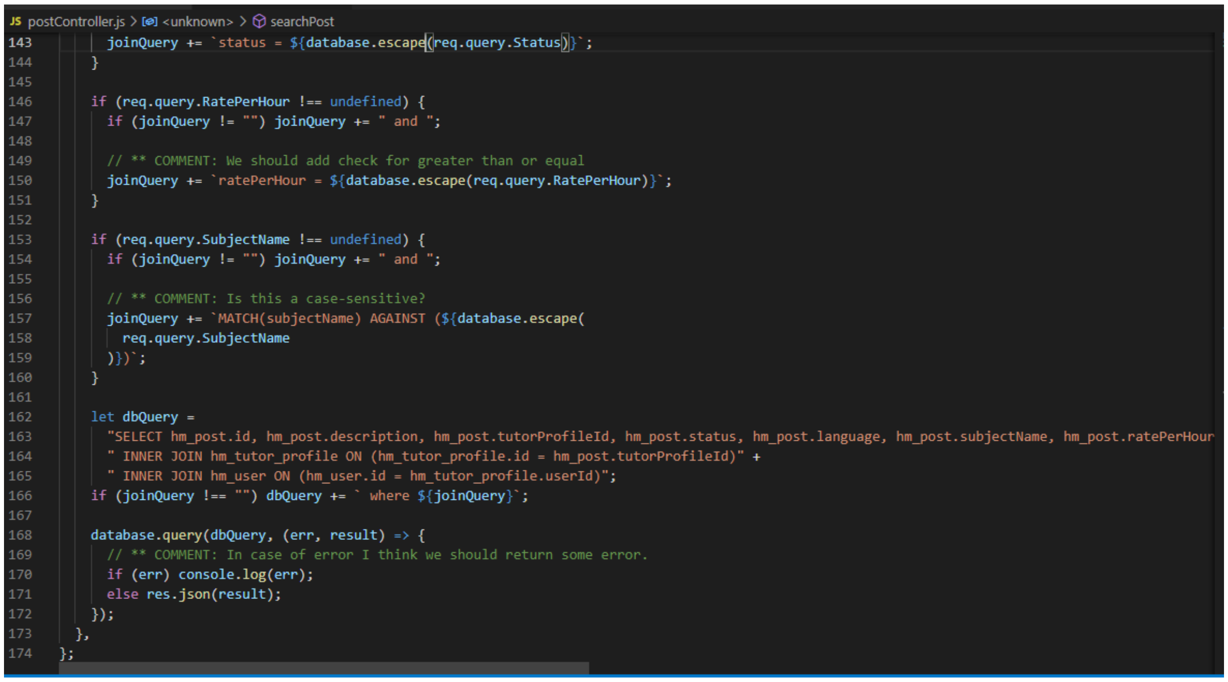


```
    if (!errors.isEmpty()) {
      return res.status(400).json({ errors: errors.array() });
    }

    let {
      Id,
      Description,
      TutorProfileId,
      Status,
      Language,
      SubjectName,
      RatePerHour,
      ExperinceYears,
      AvailableTime,
    } = req.body;

    // ** COMMENT: Can we move this to a util function?
    var date = new Date().toISOString().split("T")[0];
    database.query(
      "UPDATE hm_post SET description=?, tutorProfileId=?, status=?, `language`=?, subjectName=?, ratePerHour=?, modifiedDateTime=?, experienceYe
      [
        Description,
        TutorProfileId,
        Status,
        Language,
        SubjectName,
        RatePerHour,
        date,
        ExperinceYears,
        AvailableTime,
        Id,
      ],
      (err, result) => {
```



postController.js > <unknown> > updatePost

```javascript
      ],
      (err, result) => {
        if (err) res.status(400).send(`Response Error: ${err}`);
        else res.status(204).json({ message: "Post Details Updated" });
      }
    );
  },

  getPost: async (req, res) => {
    let id = req.params.id;
    database.query(
      "SELECT id, description, tutorProfileId, status, `language`, subjectName, ratePerHour, createdDateTime, modifiedDateTime, experienceYears,
      [id],
      (err, result) => {
        if (err) res.status(400).send(`Response Error: ${err}`);
        else res.status(200).json(result);
      }
    );
  },

  searchPost: async (req, res) => {

    // ** COMMENT: We can move join query logic to a method.
    let joinQuery = "";
    if (req.query.TutorProfileId !== undefined) {
      joinQuery += `tutorProfileId = ${database.escape(
        req.query.TutorProfileId
      )}`;
    }

    if (req.query.Status !== undefined) {
      if (joinQuery != "") joinQuery += " and ";
```



postController.js > <unknown> > searchPost

```javascript
      joinQuery += `status = ${database.escape(req.query.Status)}`;
    }

    if (req.query.RatePerHour !== undefined) {
      if (joinQuery != "") joinQuery += " and ";

      // ** COMMENT: We should add check for greater than or equal
      joinQuery += `ratePerHour = ${database.escape(req.query.RatePerHour)}`;
    }

    if (req.query.SubjectName !== undefined) {
      if (joinQuery != "") joinQuery += " and ";

      // ** COMMENT: Is this a case-sensitive?
      joinQuery += `MATCH(subjectName) AGAINST (${database.escape(
        req.query.SubjectName
      )})`;
    }

    let dbQuery =
      "SELECT hm_post.id, hm_post.description, hm_post.tutorProfileId, hm_post.status, hm_post.language, hm_post.subjectName, hm_post.ratePerHour
      " INNER JOIN hm_tutor_profile ON (hm_tutor_profile.id = hm_post.tutorProfileId)" +
      " INNER JOIN hm_user ON (hm_user.id = hm_tutor_profile.userId)";
    if (joinQuery !== "") dbQuery += ` where ${joinQuery}`;

    database.query(dbQuery, (err, result) => {
      // ** COMMENT: In case of error I think we should return some error.
      if (err) console.log(err);
      else res.json(result);
    });
  },
};
```

Developer	Hasib Iqbal
Reviewer	Chowdhury Amlan Barua
File Name	ManageTutorsProfile.js tutor.js

File Name: ManageTutorsProfile.js

```
import React from "react";
import { useSelector } from "react-redux";
import { ListGroup } from "react-bootstrap";
import TutorProfileItem from "./TutorProfileItem";
import { getTutorsProfileList } from
"../../../../core/selectors/manageTutorsProfile";
import Page from "../../../../components/page/Page";
import FilterBar from "./filterBar/FilterBar";

// Destructuring the props might be a good idea. You can do this with the
reference below:
// https://medium.com/@lcriswell/destructuring-props-in-react-b1c295005ce0

function ManageTutorsProfile(props) {
 var data = useSelector(getTutorsProfileList);

 if (data === undefined) {
   return <div></div>;
 }

 return (
   <Page>
     <FilterBar />
     <br />
     <ListGroup>
       {data?.map((item, i) => {
         return <TutorProfileItem key={i} item={item} />;
       })}
     </ListGroup>
     <br />
   </Page>
 );
}

export default ManageTutorsProfile;
```

File Name: tutor.js

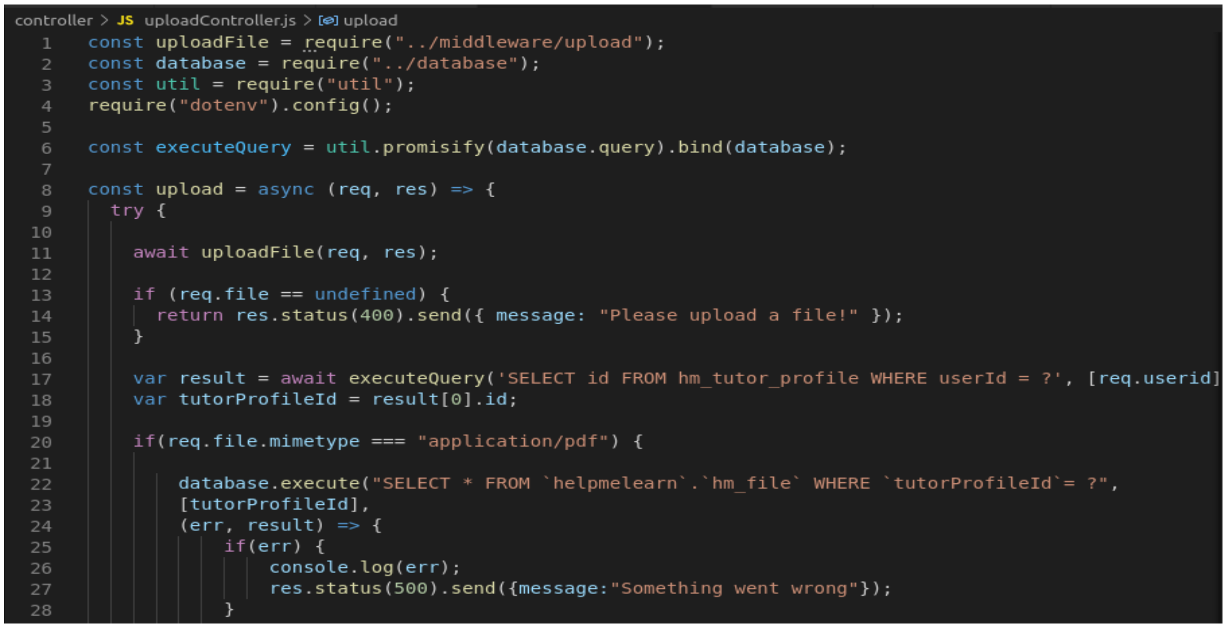
```
export function* getTutorList(action: Object): Saga<void> {
 const { filters } = action.payload;

 var url = allTutorListApi;

 if (filters.fName) {
   url += `&FirstName=${filters.fName}`;
 }
 if (filters.lName) {
   url += `&LastName=${filters.lName}`;
 }
 if (filters.email) {
   url += `&Email=${filters.email}`;
 }


 const apiOptions: ApiOptions = {
   url: url,
   method: "GET",
   useJwtSecret: false,
 };

 const apiResponse: ApiResponse = yield call(executeApiCall, apiOptions);

 const { isSuccessful, response = {} } = apiResponse;

 if (isSuccessful) {
   var data = response;
   yield put(getTutorListSuccess({ data }));
 } else {
   var msg = "Failed to load data from API"; //A more descriptive error
message might be constructed
```

```
    yield put(getTutorListFailed({ msg }));
  }
}
```

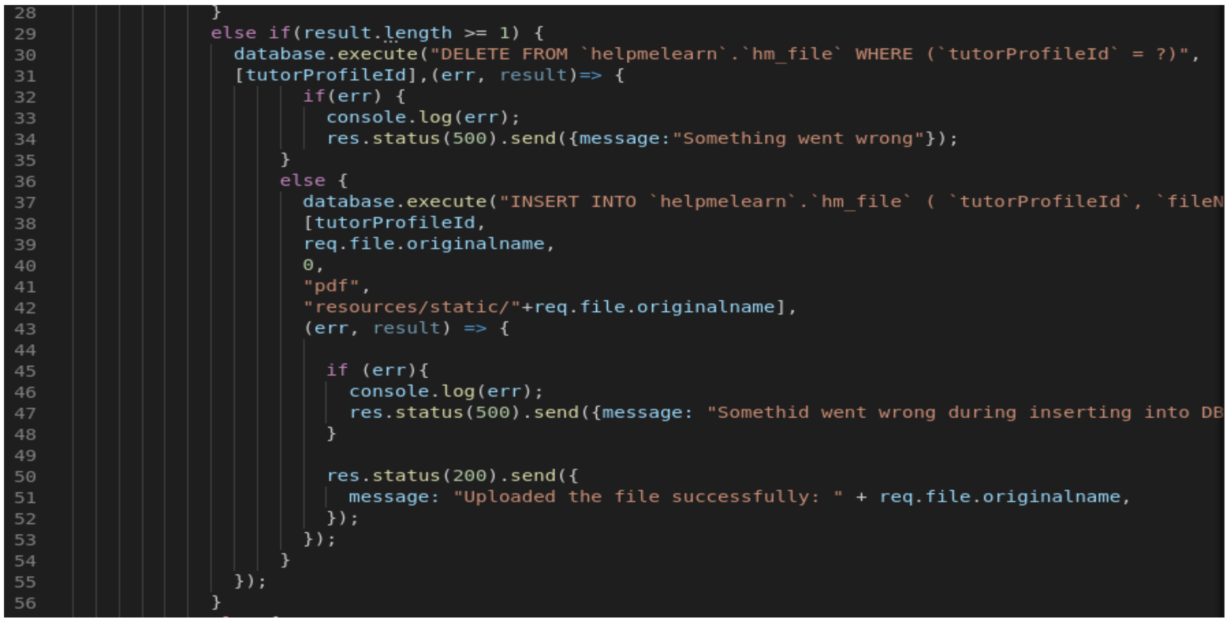
Developer	Mohammad Rakibul Hasan
Reviewer	Hasib Iqbal
File Name	AddQualification.js

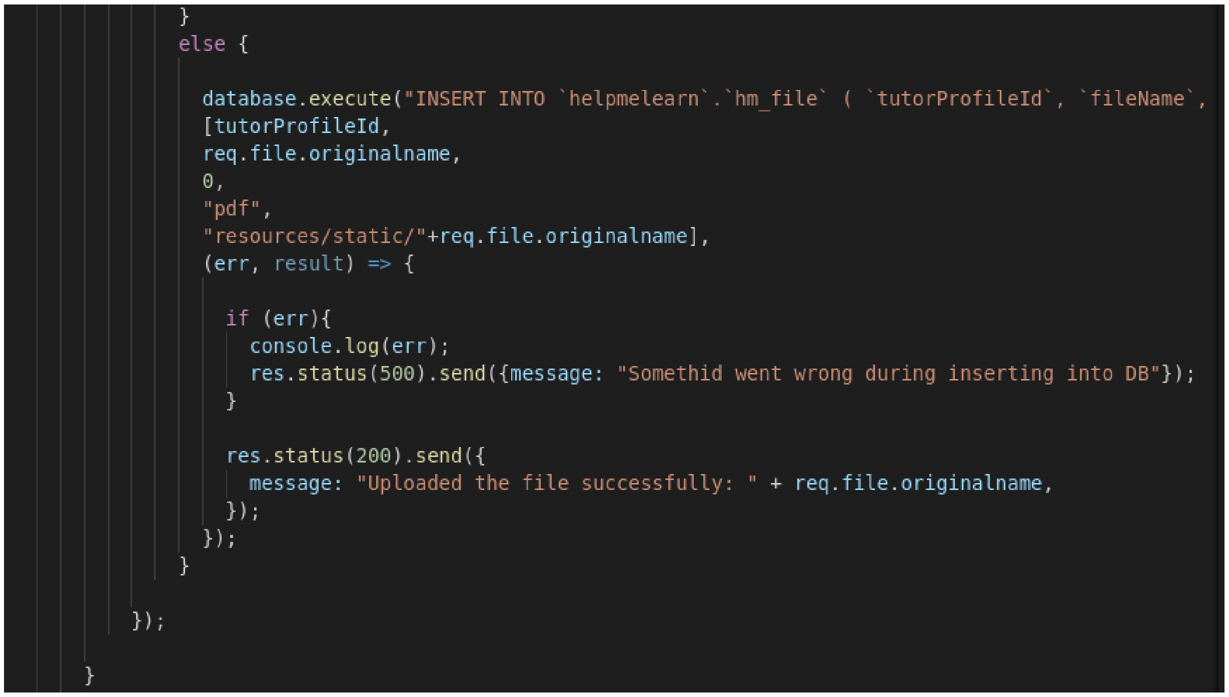
```
function AddQualification(props) {
  const dispatch = useDispatch();

  const subjectRef = useRef(null);
  const qualificationRef = useRef(null);
  const gradeRef = useRef(null);
  const descriptionRef = useRef(null);

  const user = useSelector(getCurrentUser);
  console.log("userid" + user );

  // Review Comment:
  // 1.Follow Javascript naming convention for variable names (Ref>
https://www.w3schools.com/js/js_conventions.asp)
  // 2.Add comments for better code readability
  // 3. Remove unnecessary codes


  //function to save the qualification
  const submitQualification = () => {
    const qualification = {
      SubjectName: subjectRef.current.value,
      Grade: gradeRef.current.value,
      Description: descriptionRef.current.value,
      UserId: user.id
    };
    console.log(qualification);
    dispatch(saveQualification(qualification));
    //test
    // dispatch(fetchQualificationById(1));
  };

  return (
    <div>
    <Page></Page>
    <div className="qualification-page">
      <div className="qualification-content">
        <h1>Add Qualification</h1>
        <Form>
          <br />
          <Form.Control type="text" ref={subjectRef} placeholder="Subject"
/>
          <br />
          {/* <Form.Control type="text" ref={qualificationRef}
placeholder="Qualification" />
          <br /> */}
          <Form.Control type="text" ref={gradeRef} placeholder="Grade" />
          <br />
          <Form.Control
            ref={descriptionRef}
            as="textarea"
            rows={3}
            placeholder="Description"
          />
          <Button className="btn btn-success" variant="primary"
onClick={submitQualification} type="submit">
            Save
          </Button>
        </Form>
      </div>
    </div>
    </div>
  );
}
```

Developer	Mohammad Salman Haydar
Reviewer	Talha Jahangir Khan
File Name	UploadController.js

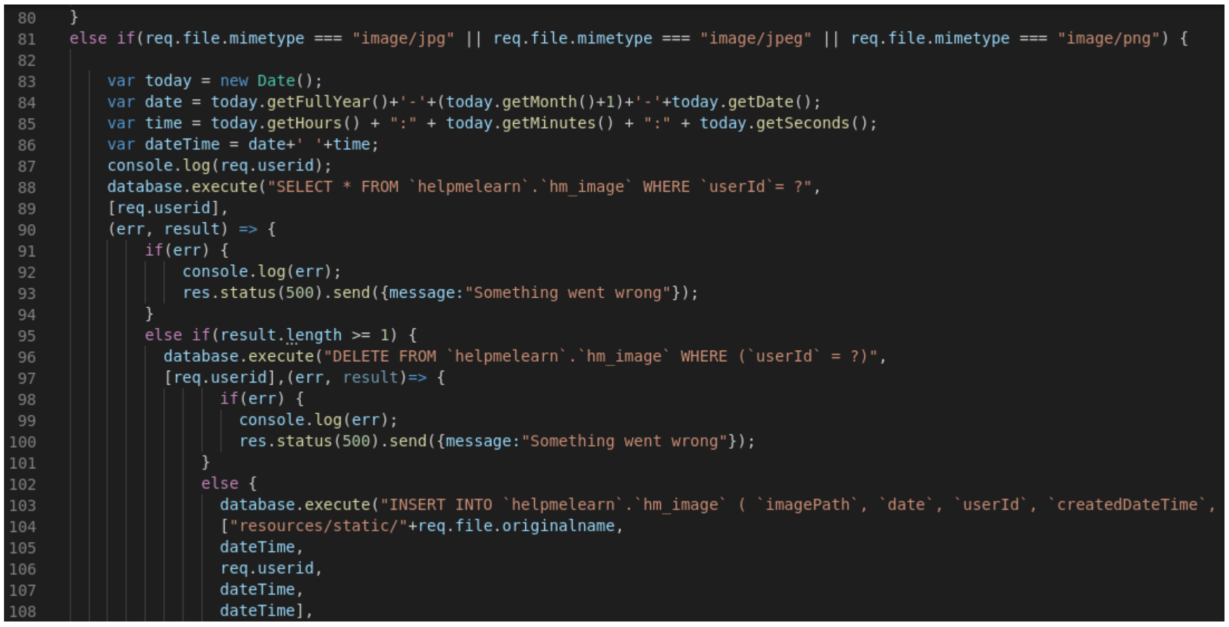

controller > JS uploadController.js > upload

```javascript
const uploadFile = require("../middleware/upload");
const database = require("../database");
const util = require("util");
require("dotenv").config();

const executeQuery = util.promisify(database.query).bind(database);

const upload = async (req, res) => {
  try {

    await uploadFile(req, res);

    if (req.file == undefined) {
      return res.status(400).send({ message: "Please upload a file!" });
    }

    var result = await executeQuery('SELECT id FROM hm_tutor_profile WHERE userId = ?', [req.userid]
    var tutorProfileId = result[0].id;

    if(req.file.mimetype === "application/pdf") {

      database.execute("SELECT * FROM `helpmelearn`.`hm_file` WHERE `tutorProfileId`= ?",
      [tutorProfileId],
      (err, result) => {
        if(err) {
          console.log(err);
          res.status(500).send({message:"Something went wrong"});
        }
```

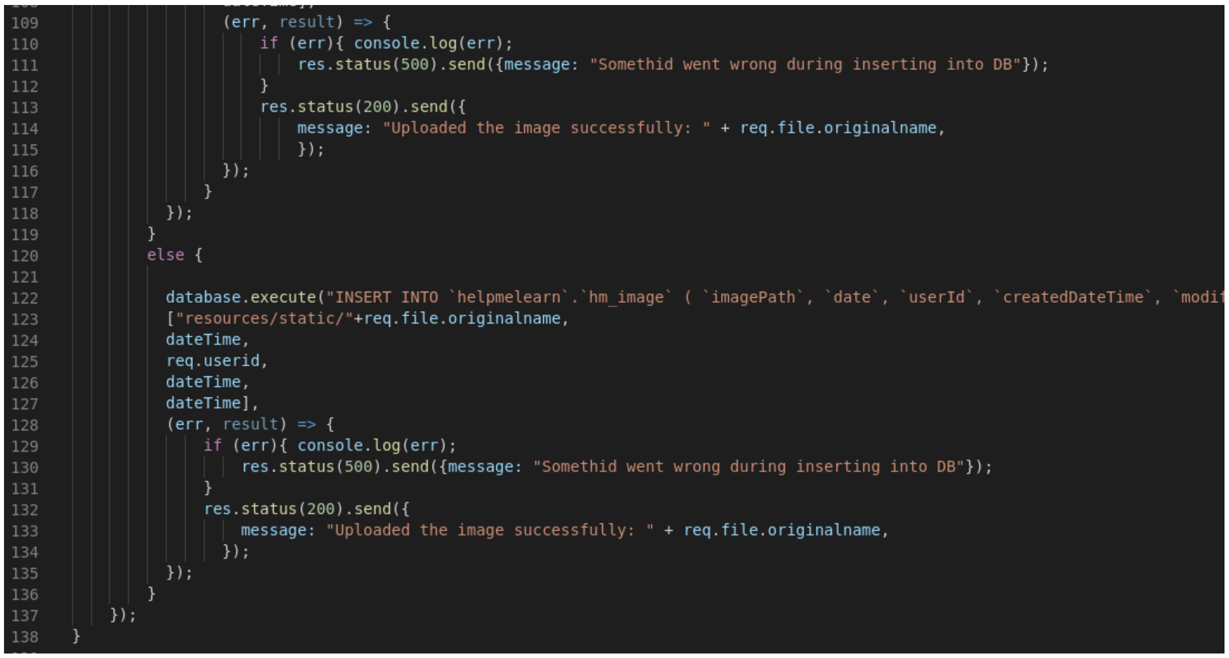

```javascript
        }
        else if(result.length >= 1) {
          database.execute("DELETE FROM `helpmelearn`.`hm_file` WHERE (`tutorProfileId` = ?)",
          [tutorProfileId],(err, result)=> {
              if(err) {
                console.log(err);
                res.status(500).send({message:"Something went wrong"});
              }
            else {
              database.execute("INSERT INTO `helpmelearn`.`hm_file` ( `tutorProfileId`, `fileN
              [tutorProfileId,
              req.file.originalname,
              0,
              "pdf",
              "resources/static/"+req.file.originalname],
              (err, result) => {

                if (err){
                  console.log(err);
                  res.status(500).send({message: "Somethid went wrong during inserting into DB
                }

                res.status(200).send({
                  message: "Uploaded the file successfully: " + req.file.originalname,
                });
              });
            }
          });
        }
```



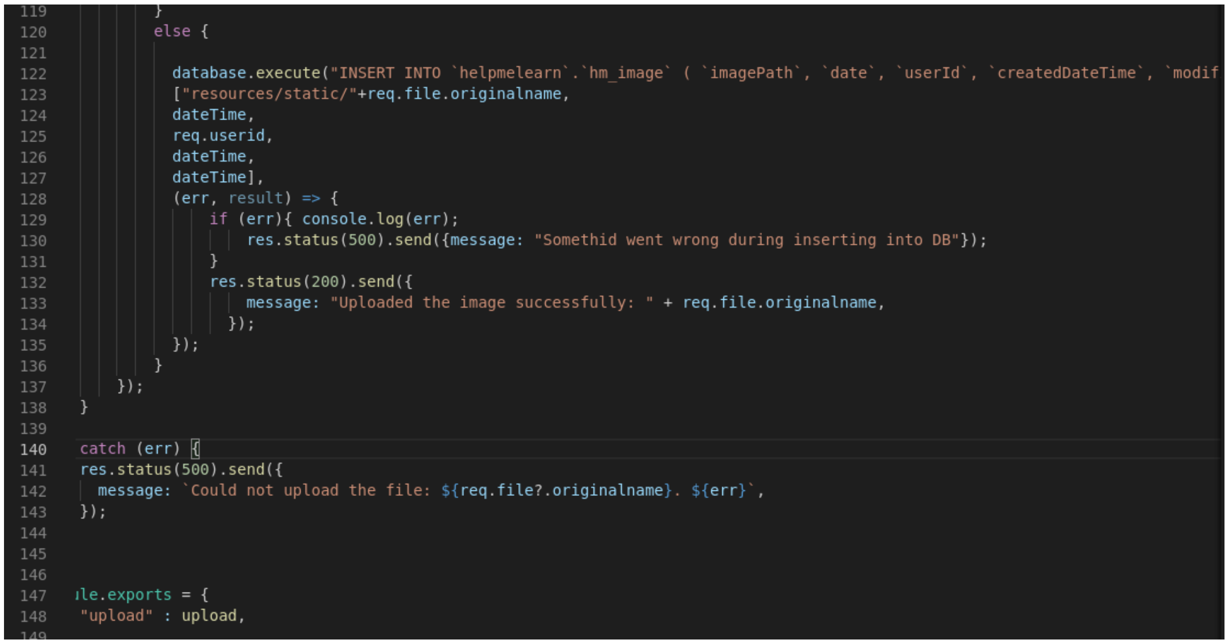
```
            }
            else {

              database.execute("INSERT INTO `helpmelearn`.`hm_file` ( `tutorProfileId`, `fileName`,
              [tutorProfileId,
              req.file.originalname,
              0,
              "pdf",
              "resources/static/"+req.file.originalname],
              (err, result) => {

                if (err){
                  console.log(err);
                  res.status(500).send({message: "Somethid went wrong during inserting into DB"});
                }

                res.status(200).send({
                  message: "Uploaded the file successfully: " + req.file.originalname,
                });
              });
            }

        });

    }
```



```
}
else if(req.file.mimetype === "image/jpg" || req.file.mimetype === "image/jpeg" || req.file.mimetype === "image/png") {

    var today = new Date();
    var date = today.getFullYear()+'-'+(today.getMonth()+1)+'-'+today.getDate();
    var time = today.getHours() + ":" + today.getMinutes() + ":" + today.getSeconds();
    var dateTime = date+' '+time;
    console.log(req.userid);
    database.execute("SELECT * FROM `helpmelearn`.`hm_image` WHERE `userId`= ?",
    [req.userid],
    (err, result) => {
        if(err) {
            console.log(err);
            res.status(500).send({message:"Something went wrong"});
        }
        else if(result.length >= 1) {
          database.execute("DELETE FROM `helpmelearn`.`hm_image` WHERE (`userId` = ?)",
          [req.userid],(err, result)=> {
                if(err) {
                  console.log(err);
                  res.status(500).send({message:"Something went wrong"});
                }
                else {
                  database.execute("INSERT INTO `helpmelearn`.`hm_image` ( `imagePath`, `date`, `userId`, `createdDateTime`,
                  ["resources/static/"+req.file.originalname,
                  dateTime,
                  req.userid,
                  dateTime,
                  dateTime],
```

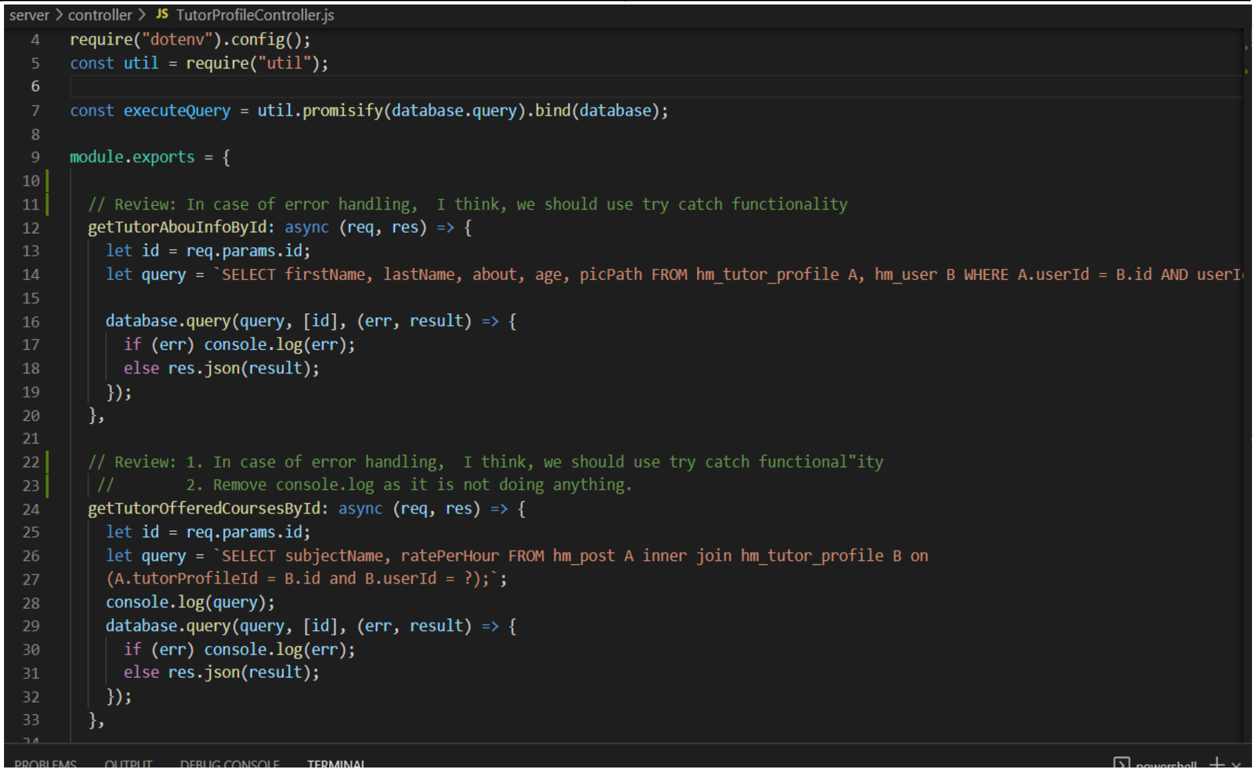


```
                (err, result) => {
                    if (err){ console.log(err);
                        res.status(500).send({message: "Somethid went wrong during inserting into DB"});
                    }
                    res.status(200).send({
                        message: "Uploaded the image successfully: " + req.file.originalname,
                        });
                });
            }
        });
    }
    else {

        database.execute("INSERT INTO `helpmelearn`.`hm_image` ( `imagePath`, `date`, `userId`, `createdDateTime`, `modif
        ["resources/static/"+req.file.originalname,
        dateTime,
        req.userid,
        dateTime,
        dateTime],
        (err, result) => {
            if (err){ console.log(err);
                res.status(500).send({message: "Somethid went wrong during inserting into DB"});
            }
            res.status(200).send({
                message: "Uploaded the image successfully: " + req.file.originalname,
                });
        });
    }
});
}
```



```
    }
    else {

        database.execute("INSERT INTO `helpmelearn`.`hm_image` ( `imagePath`, `date`, `userId`, `createdDateTime`, `modif
        ["resources/static/"+req.file.originalname,
        dateTime,
        req.userid,
        dateTime,
        dateTime],
        (err, result) => {
            if (err){ console.log(err);
                res.status(500).send({message: "Somethid went wrong during inserting into DB"});
            }
            res.status(200).send({
                message: "Uploaded the image successfully: " + req.file.originalname,
                });
        });
    }
});
}

catch (err) {
res.status(500).send({
  message: `Could not upload the file: ${req.file?.originalname}. ${err}`,
});



ıle.exports = {
 "upload" : upload,
```

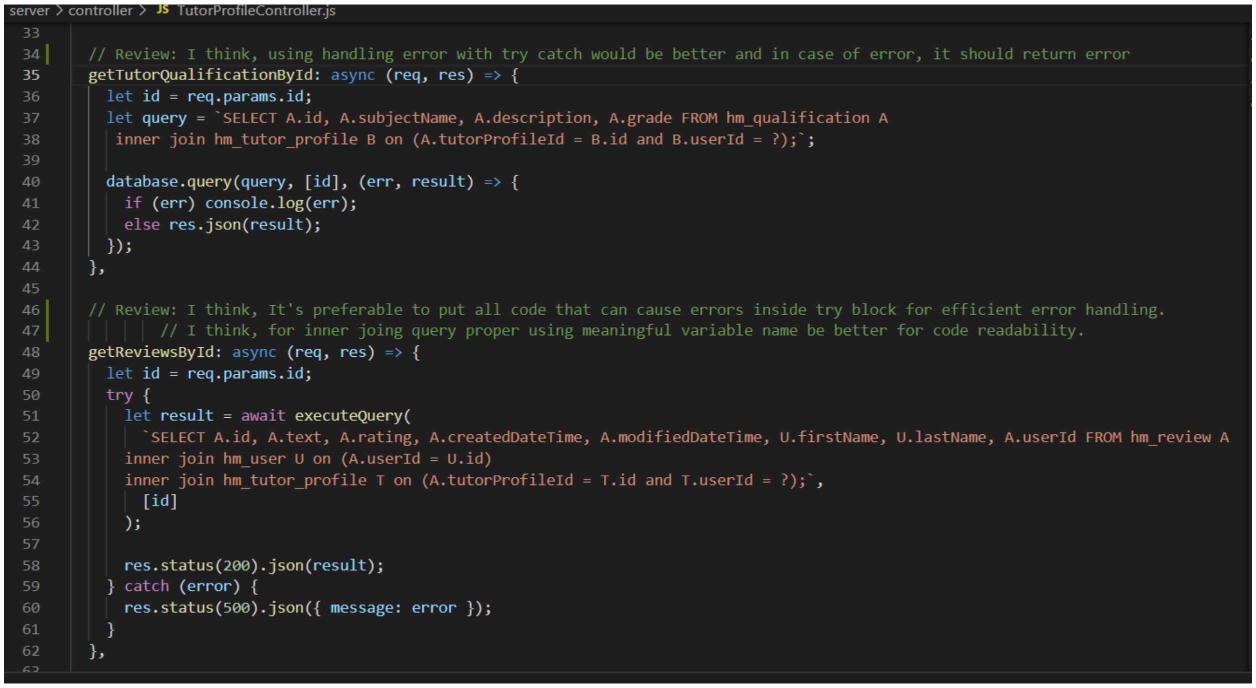
Developer	Chowdhury Amlan Barua
Reviewer	Hasib Iqbal
File Name	TutorprofilController.js



server > controller > JS TutorProfileController.js

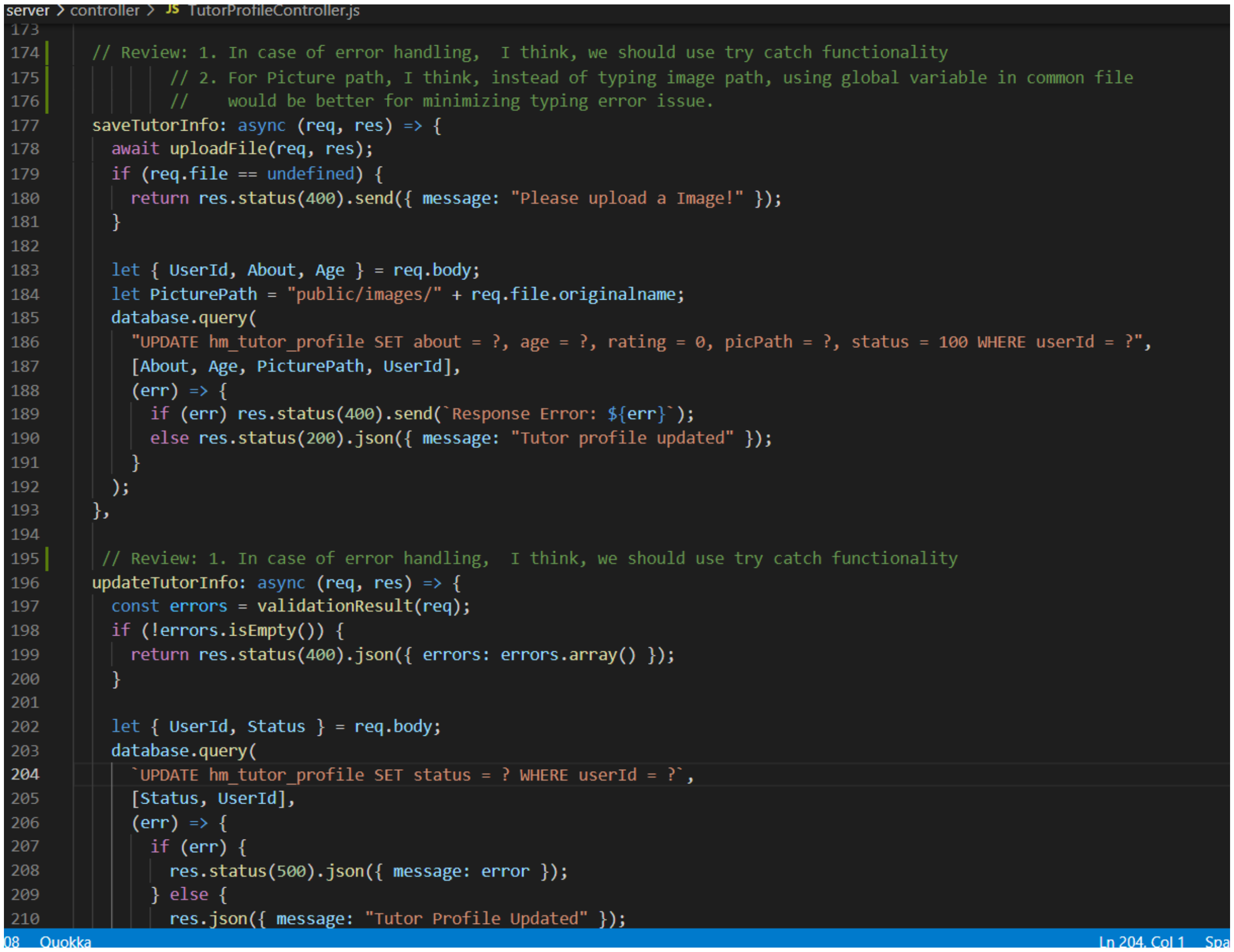
```javascript
require("dotenv").config();
const util = require("util");

const executeQuery = util.promisify(database.query).bind(database);

module.exports = {

  // Review: In case of error handling,  I think, we should use try catch functionality
  getTutorAbouInfoById: async (req, res) => {
    let id = req.params.id;
    let query = `SELECT firstName, lastName, about, age, picPath FROM hm_tutor_profile A, hm_user B WHERE A.userId = B.id AND userI

    database.query(query, [id], (err, result) => {
      if (err) console.log(err);
      else res.json(result);
    });
  },

  // Review: 1. In case of error handling,  I think, we should use try catch functional"ity
   //        2. Remove console.log as it is not doing anything.
  getTutorOfferedCoursesById: async (req, res) => {
    let id = req.params.id;
    let query = `SELECT subjectName, ratePerHour FROM hm_post A inner join hm_tutor_profile B on
    (A.tutorProfileId = B.id and B.userId = ?);`;
    console.log(query);
    database.query(query, [id], (err, result) => {
      if (err) console.log(err);
      else res.json(result);
    });
  },
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL



server > controller > JS TutorProfileController.js

```javascript

  // Review: I think, using handling error with try catch would be better and in case of error, it should return error
  getTutorQualificationById: async (req, res) => {
    let id = req.params.id;
    let query = `SELECT A.id, A.subjectName, A.description, A.grade FROM hm_qualification A
     inner join hm_tutor_profile B on (A.tutorProfileId = B.id and B.userId = ?);`;

    database.query(query, [id], (err, result) => {
      if (err) console.log(err);
      else res.json(result);
    });
  },

  // Review: I think, It's preferable to put all code that can cause errors inside try block for efficient error handling.
          // I think, for inner joing query proper using meaningful variable name be better for code readability.
  getReviewsById: async (req, res) => {
    let id = req.params.id;
    try {
      let result = await executeQuery(
        `SELECT A.id, A.text, A.rating, A.createdDateTime, A.modifiedDateTime, U.firstName, U.lastName, A.userId FROM hm_review A
      inner join hm_user U on (A.userId = U.id)
      inner join hm_tutor_profile T on (A.tutorProfileId = T.id and T.userId = ?);`,
        [id]
      );

      res.status(200).json(result);
    } catch (error) {
      res.status(500).json({ message: error });
    }
  },
```

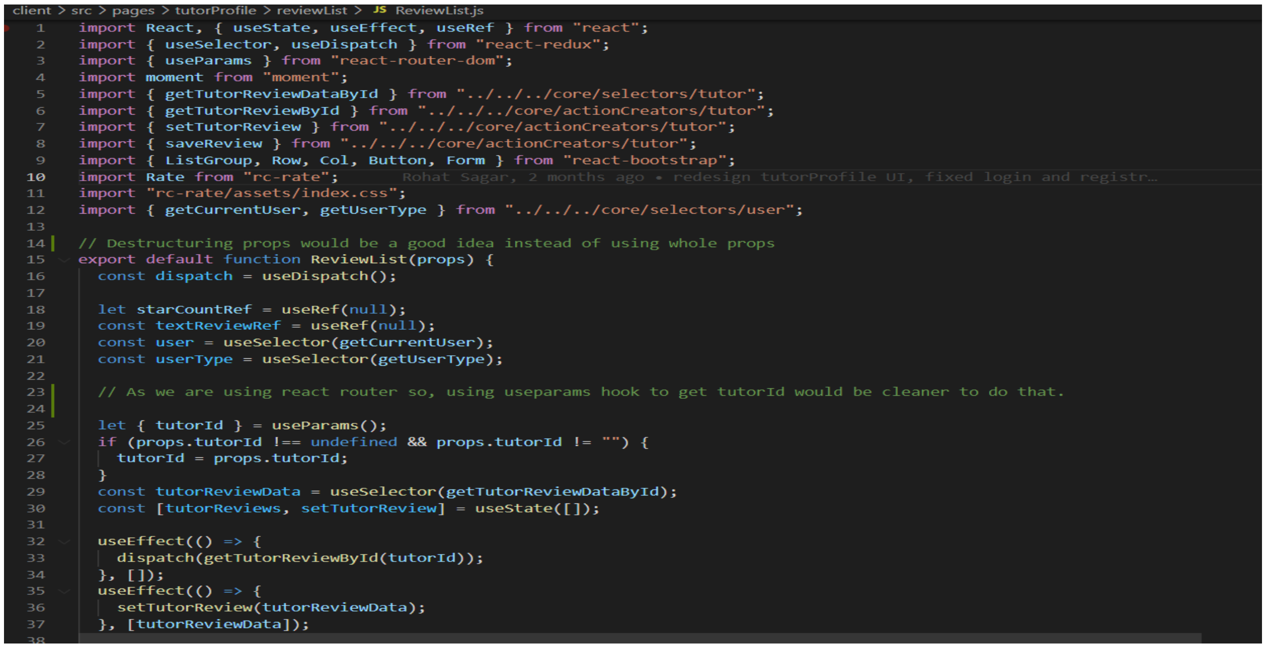


server > controller > JS TutorProfileController.js

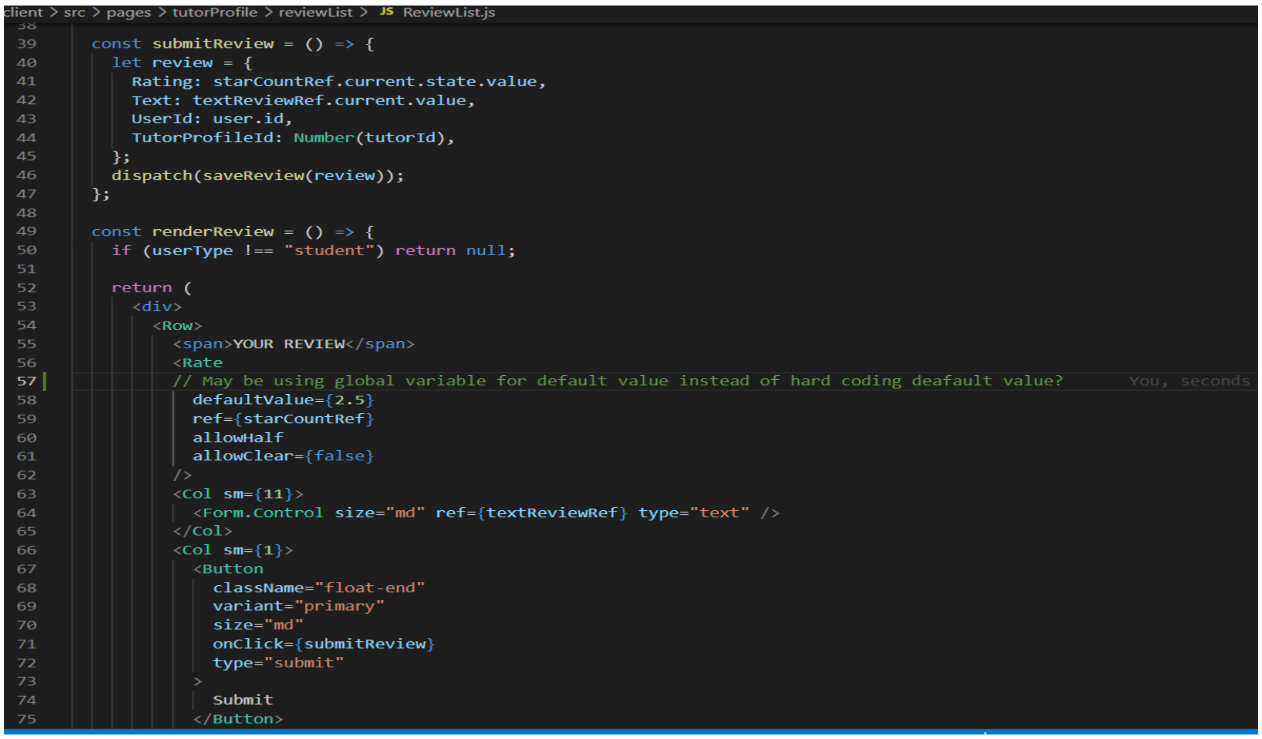
```
  // Review: 1. In case of error handling,  I think, we should use try catch functionality
          // 2. For Picture path, I think, instead of typing image path, using global variable in common file
          //    would be better for minimizing typing error issue.
  saveTutorInfo: async (req, res) => {
    await uploadFile(req, res);
    if (req.file == undefined) {
      return res.status(400).send({ message: "Please upload a Image!" });
    }

    let { UserId, About, Age } = req.body;
    let PicturePath = "public/images/" + req.file.originalname;
    database.query(
      "UPDATE hm_tutor_profile SET about = ?, age = ?, rating = 0, picPath = ?, status = 100 WHERE userId = ?",
      [About, Age, PicturePath, UserId],
      (err) => {
        if (err) res.status(400).send(`Response Error: ${err}`);
        else res.status(200).json({ message: "Tutor profile updated" });
      }
    );
  },

   // Review: 1. In case of error handling,  I think, we should use try catch functionality
  updateTutorInfo: async (req, res) => {
    const errors = validationResult(req);
    if (!errors.isEmpty()) {
      return res.status(400).json({ errors: errors.array() });
    }

    let { UserId, Status } = req.body;
    database.query(
      `UPDATE hm_tutor_profile SET status = ? WHERE userId = ?`,
      [Status, UserId],
      (err) => {
        if (err) {
          res.status(500).json({ message: error });
        } else {
          res.json({ message: "Tutor Profile Updated" });
```

08 Quokka   Ln 204, Col 1

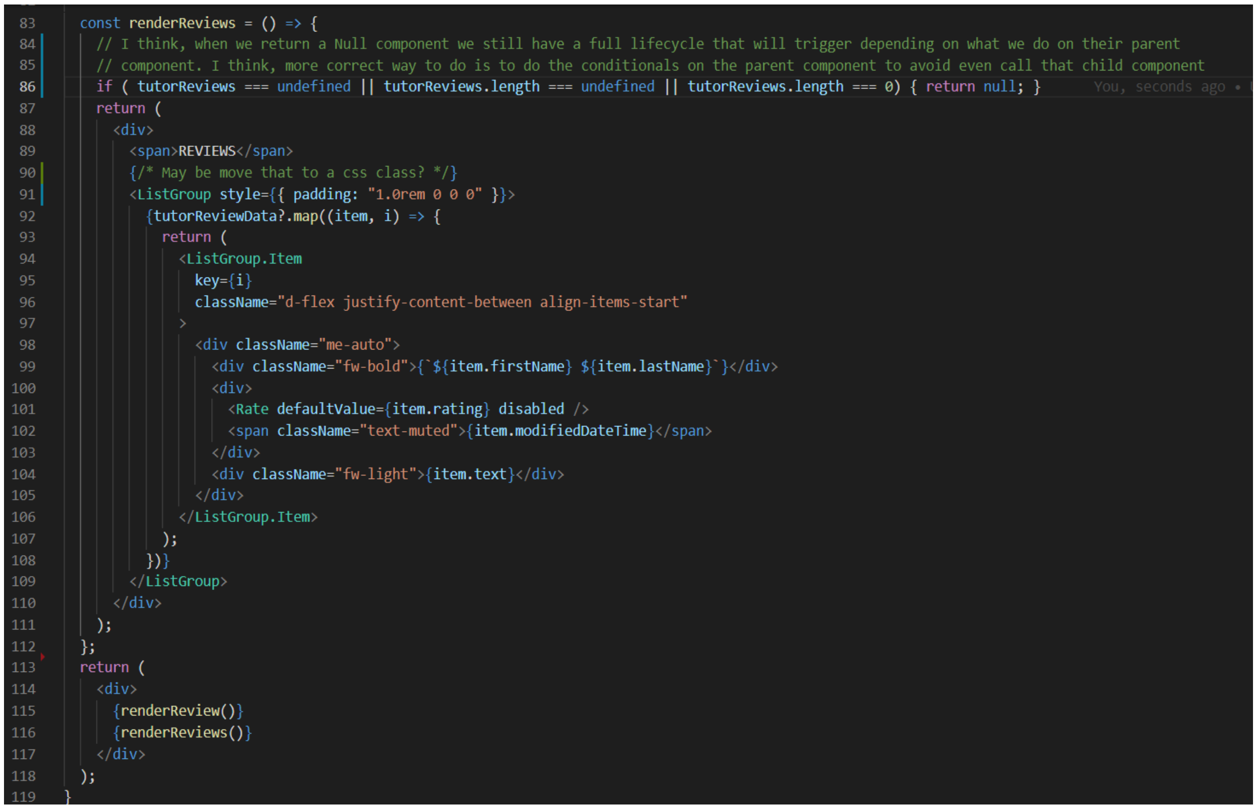
Developer	Chowdhury Amlan Barua
Reviewer	Mohmmad Rakibul Hasan
File Name	ReviewList.js


client > src > pages > tutorProfile > reviewList > JS ReviewList.js

```javascript
import React, { useState, useEffect, useRef } from "react";
import { useSelector, useDispatch } from "react-redux";
import { useParams } from "react-router-dom";
import moment from "moment";
import { getTutorReviewDataById } from "../../../core/selectors/tutor";
import { getTutorReviewById } from "../../../core/actionCreators/tutor";
import { setTutorReview } from "../../../core/actionCreators/tutor";
import { saveReview } from "../../../core/actionCreators/tutor";
import { ListGroup, Row, Col, Button, Form } from "react-bootstrap";
import Rate from "rc-rate";        Rohat Sagar, 2 months ago • redesign tutorProfile UI, fixed login and registr…
import "rc-rate/assets/index.css";
import { getCurrentUser, getUserType } from "../../../core/selectors/user";

// Destructuring props would be a good idea instead of using whole props
export default function ReviewList(props) {
  const dispatch = useDispatch();

  let starCountRef = useRef(null);
  const textReviewRef = useRef(null);
  const user = useSelector(getCurrentUser);
  const userType = useSelector(getUserType);

  // As we are using react router so, using useparams hook to get tutorId would be cleaner to do that.

  let { tutorId } = useParams();
  if (props.tutorId !== undefined && props.tutorId != "") {
    tutorId = props.tutorId;
  }
  const tutorReviewData = useSelector(getTutorReviewDataById);
  const [tutorReviews, setTutorReview] = useState([]);

  useEffect(() => {
    dispatch(getTutorReviewById(tutorId));
  }, []);
  useEffect(() => {
    setTutorReview(tutorReviewData);
  }, [tutorReviewData]);
```


client > src > pages > tutorProfile > reviewList > JS ReviewList.js

```javascript
  const submitReview = () => {
    let review = {
      Rating: starCountRef.current.state.value,
      Text: textReviewRef.current.value,
      UserId: user.id,
      TutorProfileId: Number(tutorId),
    };
    dispatch(saveReview(review));
  };

  const renderReview = () => {
    if (userType !== "student") return null;

    return (
      <div>
        <Row>
          <span>YOUR REVIEW</span>
          <Rate
          // May be using global variable for default value instead of hard coding deafault value?        You, seconds
            defaultValue={2.5}
            ref={starCountRef}
            allowHalf
            allowClear={false}
          />
          <Col sm={11}>
            <Form.Control size="md" ref={textReviewRef} type="text" />
          </Col>
          <Col sm={1}>
            <Button
              className="float-end"
              variant="primary"
              size="md"
              onClick={submitReview}
              type="submit"
            >
              Submit
            </Button>
```



```
  const renderReviews = () => {
    // I think, when we return a Null component we still have a full lifecycle that will trigger depending on what we do on their parent
    // component. I think, more correct way to do is to do the conditionals on the parent component to avoid even call that child component
    if ( tutorReviews === undefined || tutorReviews.length === undefined || tutorReviews.length === 0) { return null; }       You, seconds ago •
    return (
      <div>
        <span>REVIEWS</span>
        {/* May be move that to a css class? */}
        <ListGroup style={{ padding: "1.0rem 0 0 0" }}>
          {tutorReviewData?.map((item, i) => {
            return (
              <ListGroup.Item
                key={i}
                className="d-flex justify-content-between align-items-start"
              >
                <div className="me-auto">
                  <div className="fw-bold">{`${item.firstName} ${item.lastName}`}</div>
                  <div>
                    <Rate defaultValue={item.rating} disabled />
                    <span className="text-muted">{item.modifiedDateTime}</span>
                  </div>
                  <div className="fw-light">{item.text}</div>
                </div>
              </ListGroup.Item>
            );
          })}
        </ListGroup>
      </div>
    );
  };
  return (
    <div>
      {renderReview()}
      {renderReviews()}
    </div>
  );
}
```

Self Check on best practices for security

List of major assets that we should protect

- Passwords
- Admin routes
- Private routes

List of major threats for each asset above

- **Passwords:** should be encrypted, otherwise if the database is hacked and the passwords are stored as a plain text, then all the accounts will be exposed.
- **Admin routes:** admin functionalities are critical and only authorized users who have admin privilege can access these routes, otherwise, any user can manipulate the posts and site users improperly.
- **Private routes:** only authenticated users should access these routes like the chatting and adding posts routes.

For each asset, how we may protect it

- **Passwords:** using encryption, so all passwords are hashed and saved in the database.
- **Admin routes:** verifying the token sent in the request header and checking the user role before allowing him to access the API using the "checkAdmin" middleware.
- **Private routes:** verifying the token sent in the request header and checking if it’s valid before allowing him to access the API using the "checkAuth" middleware.
- On the client side the admin and private routes are not accessed by not logged in users.

Confirm that you encrypt PW in the DB

We are saving the passwords hashed and not as plain text in the database using the Blowfish Cipher which is one way hashing and cannot be converted to the plain text password.

Confirm Input data validation

- **Valid Email address:** a checking function is used in the signup to verify that the email address has a valid format and is related to Fulda/San Francisco university.
- **Strong Rules for passwords:** a checking function is used in the signup to enforce the user to choose a strong password that complies with the rules of the website (8 min length, one small, one capital one digit and one special char).
- **Limit the search field for up to 40 characters max:** limiting the size of the input field to 40 characters max, and preventing the user to exceed this number.

Self-check: Adherence to original Non-functional specs

List of non functional requirements	Done	On Track
Application shall be developed, tested and deployed using tools and servers approved by Class CTO and as agreed in Milestone 0. Application delivery shall be from chosen cloud server	✔	
Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of two major browsers	✔	
All or selected application functions must render well on mobile devices	✔	
Data shall be stored in the database on the team's deployment cloud server	✔	
No more than 50 concurrent users shall be accessing the application at any time	✔	
Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users.	✔	
The language used shall be English (no localization needed)	✔	
Application shall be very easy to use and intuitive	✔	
Application should follow established architecture patterns		✔
Application code and its repository shall be easy to inspect and maintain	✔	
No email clients shall be allowed.	✔	
Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated in UI.	✔	
Site security: basic best practices shall be applied (as covered in the class) for main data items	✔	
Application shall be media rich (images, video etc.). Media formats shall be standard as used in the market today		✔
Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development	✔	
For code development and management, as well as documentation like formal milestones required in the class, each team shall use their own GitHub to be set-up by class instructors and started by each team during Milestone 0	✔	
The application UI (WWW and mobile) shall prominently display the following exact text on all pages "Fulda University of Applied Sciences Software Engineering Project, Fall 2021 For Demonstration Only" at the top of the WWW page.	✔	