



**BSc (Hons) in Information Technology Specializing in Software  
Engineering  
Year 3 - 2022**

**SE3040 – Application Frameworks  
Group Project: Technical Report**

Group Name: SE3030\_WE\_68

**Group Member Details**

Student ID number	Name	SLIIT Email
IT20236564	Kumarathunga M.A.M.T	<a href="mailto:it20236564@my.sliit.lk">it20236564@my.sliit.lk</a>
IT20226046	Kavinda M.A.I.N	<a href="mailto:it20226046@my.sliit.lk">it20226046@my.sliit.lk</a>
IT20227036	M.M.M.B Moremada	<a href="mailto:it20227036@my.sliit.lk">it20227036@my.sliit.lk</a>
IT20134280	S.Ishalini	<a href="mailto:it20134280@my.sliit.lk">it20134280@my.sliit.lk</a>

Git Repository: <https://github.com/kavindamain99/Research-Project-Management-Tool-frontend>

Backend: <https://github.com/kavindamain99/Research-Project-Management-Tool-backend>

Group Presentation Video: [Recordings - OneDrive 1.mp4](#)

Project Start Date: 15 March 2022

Project End Date: 4 July 2022

**Project Declaration:**

I the undersigned solemnly declare that the project technical based on our group members own work carried out during the course of our study under the supervision of Mr. Thusithanana Thilakarathna.

A handwritten signature in black ink, appearing to read "Kumarathunga M.A.M.T.", is placed over a horizontal line.

Group Leader:

## Marking Rubric

Date	Weight	Marks (out of 5)			
Description		5	5	5	5
<b>1. SRS Document -</b>	<b>10</b>				
N/A	5				
N/A	5				
<b>2. Frontend Development</b>	<b>25</b>				
a. Viva	10				
b. demo					
1. Implemented all the components [2]					
2. UI/UX [08]					
3. Components [5]	15				
<b>3. Backend Development</b>	<b>25</b>				
a. Viva	10				
b. Demo					
1. Routes [5]					
2. DAL [5]					
3. DAO [5]	15				
<b>4. Apps are hosted</b>	<b>10</b>				
a. Frontend	5				
b. Backend	5				
<b>5. Unit Tests</b>	<b>10</b>				
a. Frontend	5				
b. Backend	5				
<b>6. Test Cases</b>	<b>5</b>				
<b>7. Technical Report</b>					
a. Screenshots of the UI [1]					
b. Screenshots of the MongoDB [2]					
c. Description [2]	5				
<b>Total</b>	<b>100</b>				
<b>Comments</b>					

**Mark out of 5**

- 0.0 - Section is not done
- 0.1 - There is something but entirely wrong or not relevant
- 1.0 - Relevant but wrong
- 2.0 - Slightly correct
- 3.0 - Average
- 4.0 - Good - Slight errors
- 5.0 - WOW (No errors)

**Report Template.**

1. Component Description (Member wise)
2. Screenshots of the Frontend (Member wise)
3. Screenshots/ Code Snippets of the Backend (Member wise)
4. Screenshots of the MongoDB Schemas (Member wise)
5. Proof of Hosting (Backend and Frontend)

When you write the report, create topics as given in the report template and subtopics from the member's SLIIT Student ID number. Please refer to the marking rubric as well.



## **Research Project Management Tool**

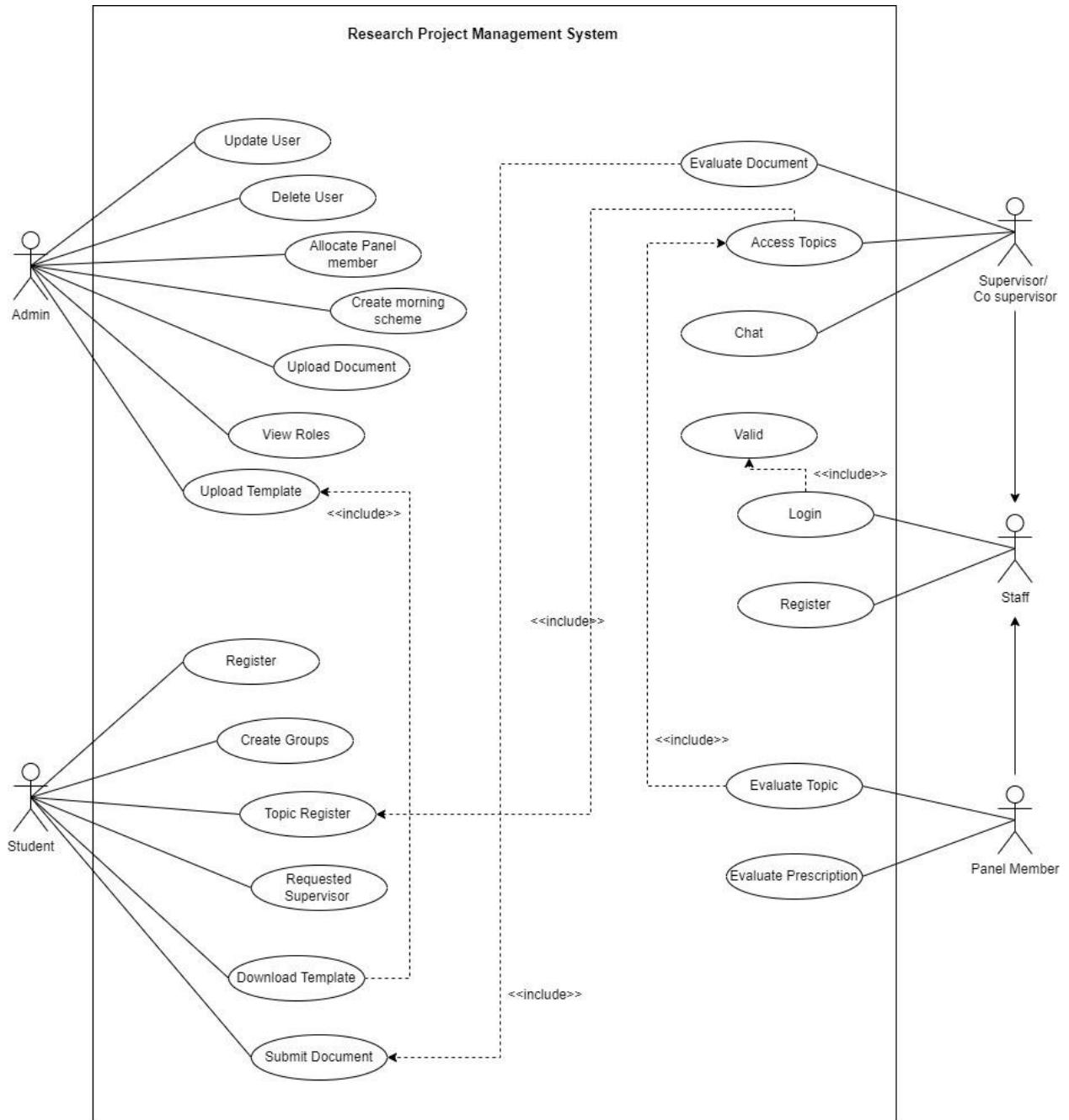
All SLIIT undergraduates must do a research project in their 4th year. This Research Project module is a 16 credit, two semesters long project. A student group must find a research topic in a specific research field and send it to a supervisor who has interests in the same research field. Once the supervisor accepted the topic, they must find a co-supervisor of the same research interest.

After finalizing the supervisor student must send a document including the topic details of the Research. Then topic evaluation panel will evaluate the topic and send feedback to the student group. If the topic is accepted, they can continue to do the project. If rejected, they must find a new topic and submit it back.

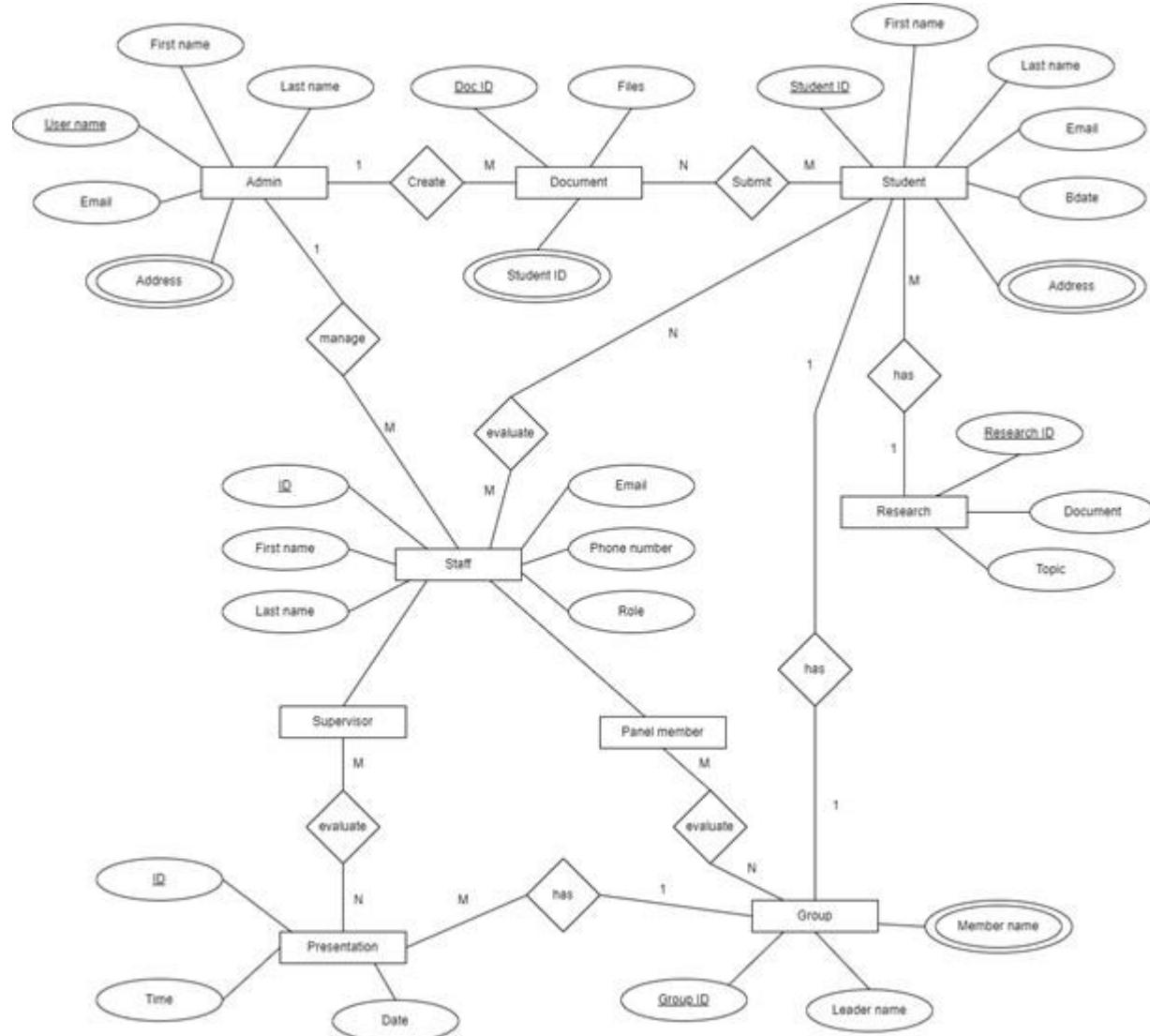
There are several evaluation stages. Document submissions are evaluated by the supervisor or the co-supervisor. Presentations will be evaluated by a separate panel. The final Thesis will be Double evaluated by the supervisor and a blind reviewer.

SLIIT Research project team is looking for a system, which has the capability of managing the Research project and automating certain tasks.

## Use Case Diagram:



## ER Diagram



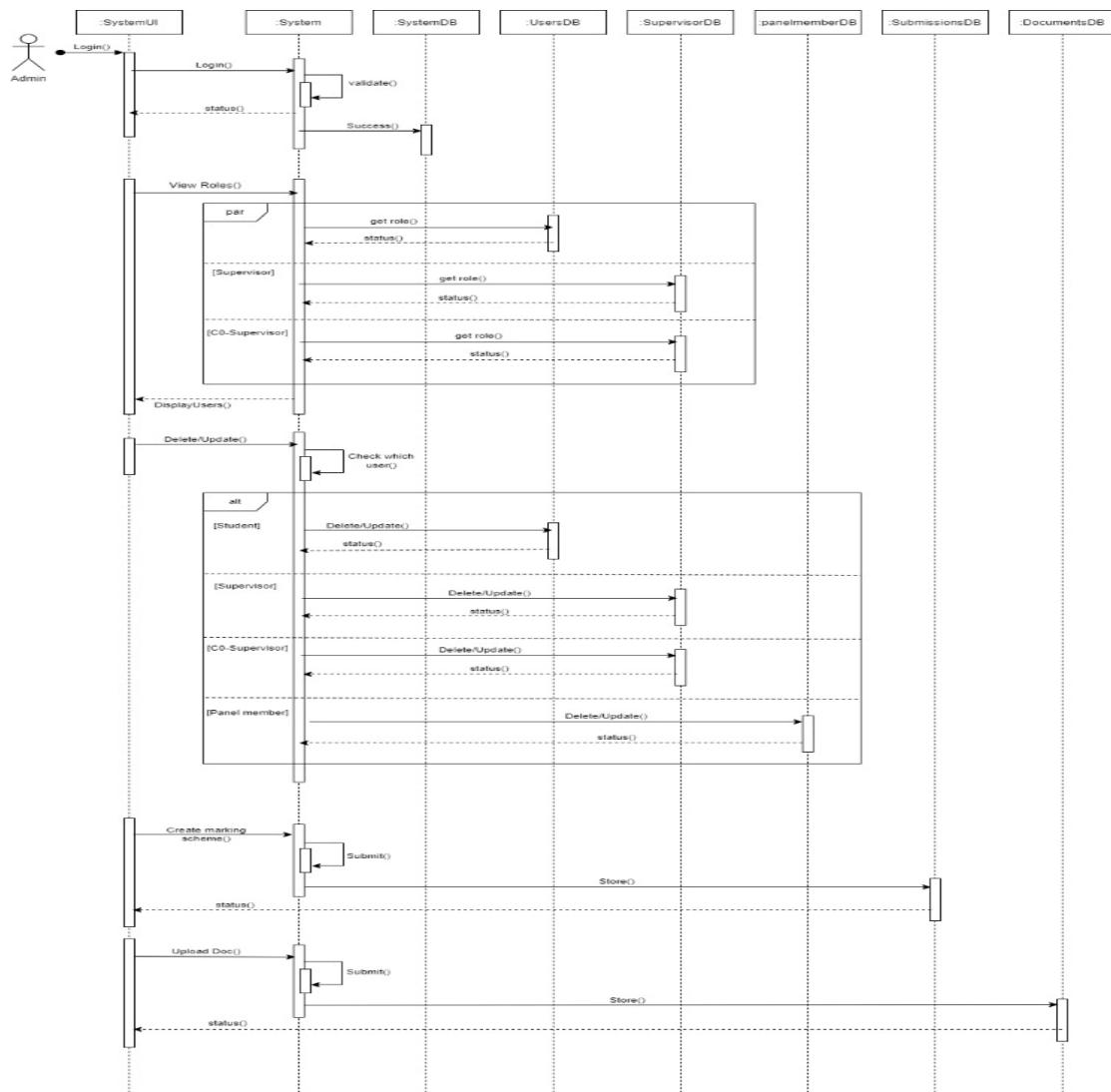
## Component Description

IT20226046 Kavinda M.A.I.N

Description:

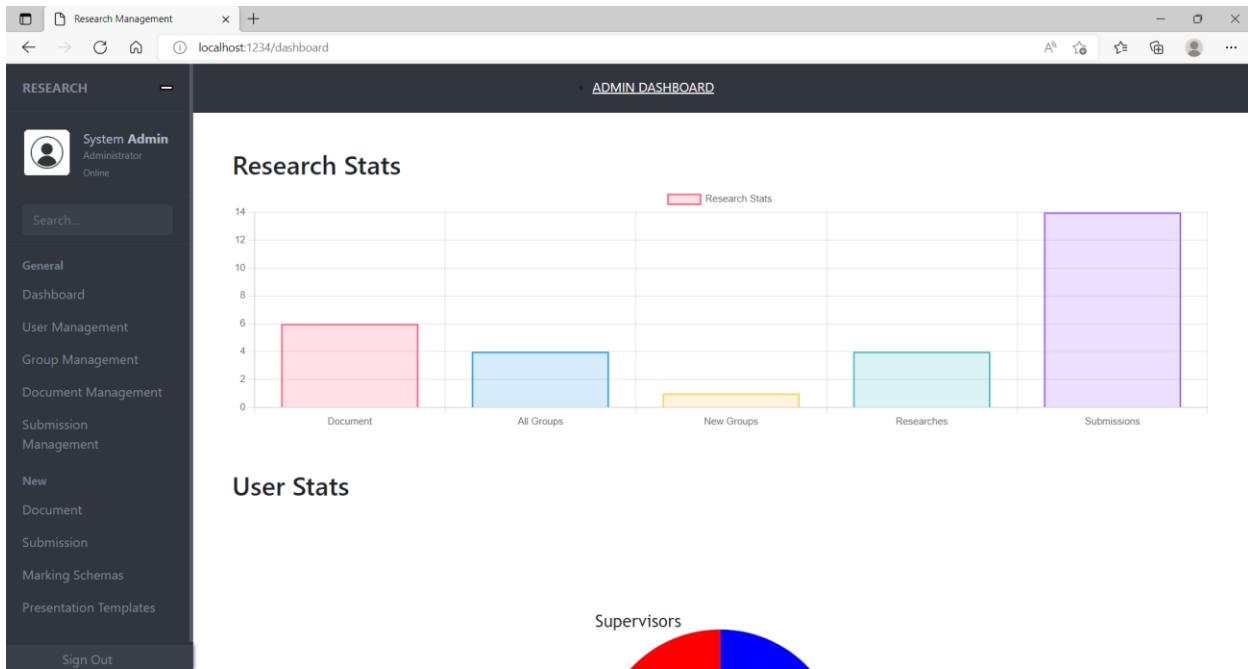
Admin Role

Admin is the owner of this system and who can view all of users and update or delete them. And also, their main function is the creation of submission part. Admin can create a submission type and create marking scheme for them. they can upload document or presentation templates to the system. Allocate panel members to student group is also a function of the administrator.



## Screenshots of the Frontend

### Admin dashboard



### Allocate panel members

The screenshot shows the Group Management page. The sidebar includes a search bar and links for General, Dashboard, User Management, Group Management, Document Management, Submission Management, New, Document, Submission, Marking Schemas, and Presentation Templates. The main content area displays an error message: 'Group : Error 403' with 'Group Id : RG2723'. It lists 'Group Members' with IDs 01, 02, 03, and 04. There is a text input field for changes and a section for 'Add Panel Members' with dropdown menus for selecting panel members.

**Group : Error 403**

**Group Id : RG2723**

**Group Members :**

- 01. IT20221021
- 02. IT202241121
- 03. IT202211012
- 04. IT202261032

Please enter Group Id for make changes

**Add Panel Members**

Please select Panel Member 1 ▾ Please select Panel Member 2 ▾ Please select Panel Member 3 ▾

Submit

## Create submission type

The screenshot shows a web-based administrative interface for 'Research Management'. The top navigation bar includes tabs for 'Research' and 'Admin Dashboard'. On the left, a sidebar menu lists various administrative functions: General, Dashboard, User Management, Group Management, Document Management, Submission Management, New, Document, Submission, Marking Schemas, and Presentation Templates. A 'Sign Out' button is at the bottom of the sidebar.

The main content area is titled 'Submissions' and contains a 'Manage Available Submissions' button. Below it, a form for 'Add New Submission' is displayed. The form fields include:

- Submission Name: A text input field.
- Description: A text area for notes.
- Submission Type: Radio buttons for '.Pdf', '.Ppt', and '.Docx'.
- Deadline: A date input field with a calendar icon.
- Degree: Radio buttons for 'Computing' and 'Business'.

## Delete / Update Users

This screenshot shows the 'User Management' section of the 'Research Management' application. The top navigation bar and sidebar are identical to the previous screenshot. The main content area is titled 'Search Users By Id' and displays a table of student records:

Student Name	Student ID	Email	Contact Number	Action	Action
Janith Liyanage	IT21229802	janith@abc.com	0717238069	<button>Update</button>	<button>Delete</button>
Kamil Mishara	IT21229802	kamil@abc.com	0717238069	<button>Update</button>	<button>Delete</button>
nadeera Gunawardhane	IT20222131	nadeera@gmail.com	0717664532	<button>Update</button>	<button>Delete</button>
Kavinda Munasinghe	IT20312134	kavindanim@gmail.com	0717664532	<button>Update</button>	<button>Delete</button>
Maheesh Fernando	IT20212151	maheesh@gmail.com	0717664532	<button>Update</button>	<button>Delete</button>
Ashan De silva	IT20436834	ashan@gmail.com	0717664532	<button>Update</button>	<button>Delete</button>

## Document management

The screenshot shows a web-based document management system. On the left, a sidebar menu lists various administrative functions: General, Dashboard, User Management, Group Management, Document Management (which is selected), Submission Management, New, Document, Submission, Marking Schemas, and Presentation Templates. The main content area is titled "ADMIN DASHBOARD". It features a search bar labeled "Search Document by name". Below the search bar is a table with columns: Document Name, Document Type, Degree, Action, and Action. The table contains six rows of document information:

Document Name	Document Type	Degree	Action	Action
Research Introduction	document	computing	Download	Delete
Student Group Registration	document	computing	Download	Delete
Student Research module delivery plan	document	computing	Download	Delete
SRS document marking schema	marking	computing	Download	Delete
SRS document marking schema	marking	computing	Download	Delete
Architecture plan marking schema	marking	computing	Download	Delete

## Submission management

The screenshot shows a web-based submission management system. The sidebar menu is identical to the one in the document management section. The main content area is titled "ADMIN DASHBOARD". It features a search bar labeled "Search Submission by name". Below the search bar are two distinct submission workflows:

**Submission : Research Plan Submission**  
Submit PDF document of Research workflow

Update Deadline  
Current Deadline : 2022-06-30T12:00  
New Deadline  
 [Calendar icon]

Update Delete

**Submission : SRS Document Submission**  
SRS Document for Research (Download your template)

Update Deadline  
Current Deadline : 2022-07-05T15:00

## Update user

The screenshot shows the 'Research Management' application's admin dashboard. The URL in the browser is `localhost:1234/student/update/628a39217644b8adfa31971a`. The left sidebar has a 'System Admin' profile and a search bar. The main area is titled 'ADMIN DASHBOARD' and contains fields for updating a user: First name (Janith), Last name (Liyanage), Email (janith@abc.com), Password (\*\*\*\*), Contact number (0717238069), and Student ID (IT21229802). A blue 'Update' button is at the bottom.

## Upload marking scheme

The screenshot shows the 'Research Management' application's admin dashboard. The URL in the browser is `localhost:1234/marking/insert`. The left sidebar has a 'System Admin' profile and a search bar. The main area is titled 'Marking Schema' and includes a 'Manage Available Schema' button. It has fields for Submission Name and Description, two checkboxes for 'Staff Allowed' and 'Student Allowed', three radio buttons for Degree (Computing, Business, Engineering), and a 'Document' section with a 'Choose File' button. A 'Submit' button is at the bottom.

## Upload presentation template

The screenshot shows a web browser window titled "Research Management" with the URL "localhost:1234/presentation/insert". The page is titled "Presentation Templates" and includes a "Manage Available Templates" button. It has fields for "Submission Name" and "Description", and checkboxes for "Staff Allowed" and "Student Allowed". There are radio buttons for "Degree" (Computing, Business, Engineering), and a "Document" section with a "Choose File" button. A "Submit" button is at the bottom.

RESEARCH ADMIN DASHBOARD

Presentation Templates

Manage Available Templates

Submission Name:

Description:

Staff Allowed  
 Student Allowed

Degree

Computing  
 Business  
 Engineering

Document :

Choose File No file chosen

Submit

## Upload documents

The screenshot shows a web browser window titled "Research Management" with the URL "localhost:1234/document/insert". The page is titled "Documents" and includes a "Manage Available Document" button. It has fields for "Document Name" and "Description", and checkboxes for "Staff Allowed" and "Student Allowed". There are radio buttons for "Degree" (Computing, Business, Engineering), and a "Document" section with a "Choose File" button. A "Submit" button is at the bottom.

RESEARCH ADMIN DASHBOARD

Documents

Manage Available Document

Document Name:

Description:

Staff Allowed  
 Student Allowed

Degree

Computing  
 Business  
 Engineering

Document :

Choose File No file chosen

Submit

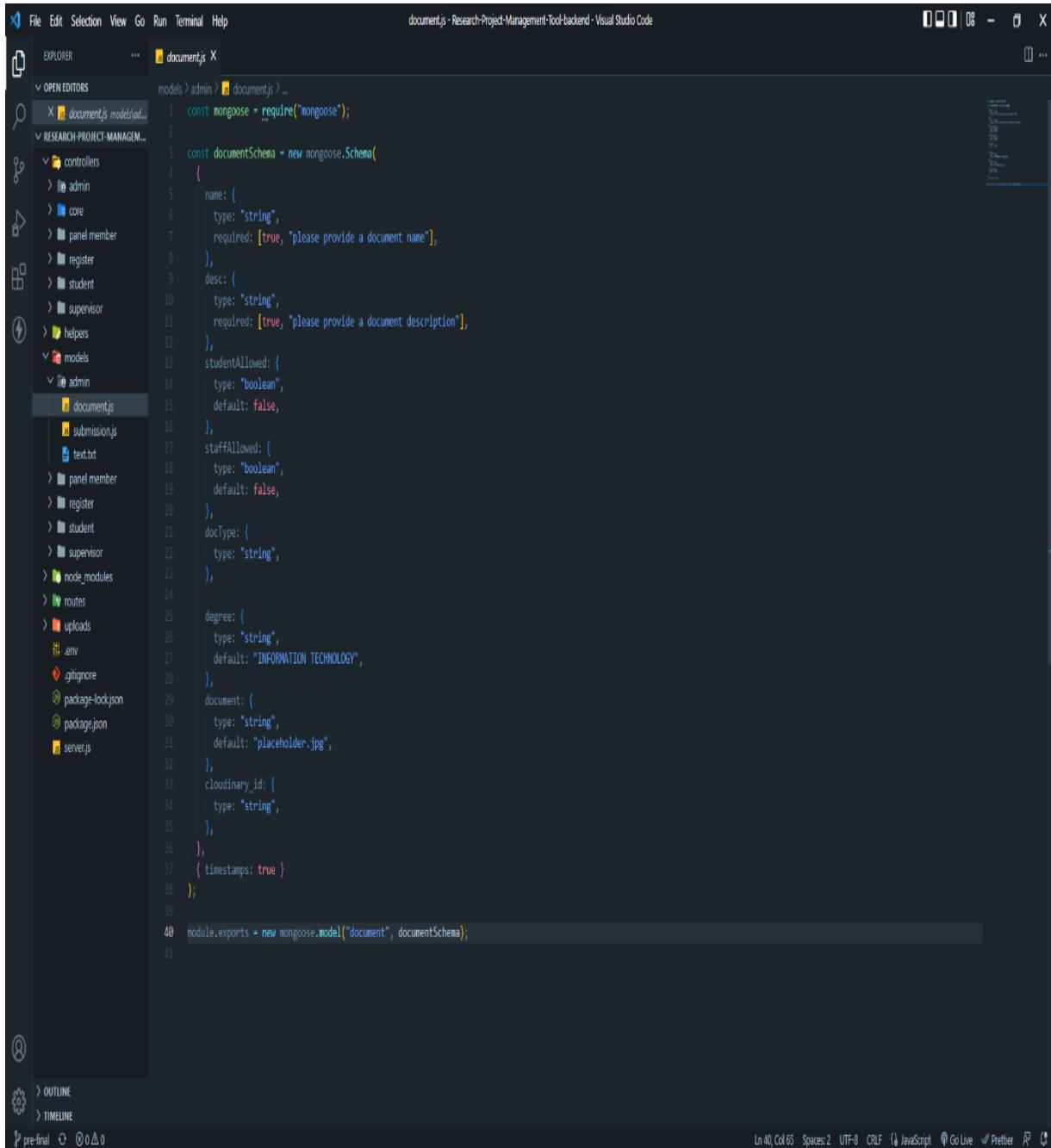
## View roles

The screenshot shows a web browser window titled "Research Management" with the URL "localhost:1234/user/management". The page is titled "ADMIN DASHBOARD" and features a sidebar on the left labeled "RESEARCH". The sidebar includes a user profile for "System Admin" (Administrator, Online), a search bar, and a list of general and research-related menu items: General, Dashboard, User Management, Group Management, Document Management, Submission Management, New, Document, Submission, Marking Schemas, and Presentation Templates. At the bottom of the sidebar is a "Sign Out" button.

The main content area is titled "User Roles" and displays four cards representing different roles:

- Students**: Includes a placeholder image of a person, a title, and a descriptive paragraph: "Some quick example text to build on the card title and make up the bulk of the card's content." A "View List" button is present.
- Supervisors**: Includes a placeholder image of a person, a title, and a descriptive paragraph: "Some quick example text to build on the card title and make up the bulk of the card's content." A "View List" button is present.
- Co Supervisors**: Includes a placeholder image of a person, a title, and a descriptive paragraph: "Some quick example text to build on the card title and make up the bulk of the card's content." A "View List" button is present.
- Panel Members**: Includes a placeholder image of a person, a title, and a descriptive paragraph: "Some quick example text to build on the card title and make up the bulk of the card's content." A "View List" button is present.

## Screenshots/ Code Snippets of the Backend:



The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** document.js - Research-Project-Management-Tool-backend - Visual Studio Code.
- Explorer:** Shows the project structure:
  - RESEARCH-PROJECT-MANAGEMENT-TOOL-BACKEND
  - controllers
  - core
  - panel member
  - register
  - student
  - supervisor
  - helpers
  - models
  - admin
    - document.js (selected)
    - submission.js
    - text.txt
  - panel member
  - register
  - student
  - supervisor
  - node\_modules
  - routes
  - uploads
  - .env
  - .gitignore
  - package-lock.json
  - package.json
  - server.js
- Code Editor:** The 'document.js' file is open, displaying the following code:

```
const mongoose = require("mongoose");

const documentSchema = new mongoose.Schema(
{
  name: {
    type: "string",
    required: [true, "please provide a document name"],
  },
  desc: {
    type: "string",
    required: [true, "please provide a document description"],
  },
  studentAllowed: {
    type: "boolean",
    default: false,
  },
  staffAllowed: {
    type: "boolean",
    default: false,
  },
  docType: {
    type: "string",
  },
  degree: {
    type: "string",
    default: "INFORMATION TECHNOLOGY",
  },
  document: {
    type: "string",
    default: "placeholder.jpg",
  },
  cloudinary_id: {
    type: "string",
  },
},
{ timestamps: true }
);

module.exports = new mongoose.model("document", documentSchema);
```
- Bottom Status Bar:** Ln 40, Col 65 Spaces: 2 CRLF ↴ JavaScript ⌂ Go Live ⌂ Prettier ⌂

```
File Edit Selection View Go Run Terminal Help
DOSWER
OPEN EDITORS ... documents.js
RESEARCH-PROJECT-MANAGEMENT
  controllers
    admin
      document.js
      controllers
      graph.js
      staff.js
      stGroup.js
      submission.js
      text.js
      user.js
    core
    panel-member
    register
    student
    supervisor
    helpers
    models
      admin
        document.js
        submission.js
        text.js
      panel-member
      register
      student
      supervisor
    routes
    uploads
    env
    ignore
    package-lock.json
    package.json
    server.js
document.js - Research-Project-Management-Tool-backend - Visual Studio Code
controllers > admin > document.js > (6) insertDocument
1 const Document = require('../models/admin/document');
2 const cloudinary = require('../helpers/admin/cloudinary');
3
4 const insertWorkingSchema = async (req, res) => {
5   try {
6     const result = await cloudinary.uploader.upload(req.file.path, {
7       folder: 'working',
8     });
9
10    let document = new Document({
11      name: req.body.name,
12      desc: req.body.desc,
13      studentAllowed: req.body.studentAllowed,
14      staffAllowed: req.body.staffAllowed,
15      docType: "working",
16      degree: req.body.degree,
17    });
18
19    document.result.secure_url =
20      cloudinary_id: result.public_id,
21
22    documentSaved = await document.save();
23
24    res.json(documentSaved);
25  } catch (error) {
26    res.json(error);
27  }
28}
29
30 const insertPresentation = async (req, res) => {
31   try {
32     const result = await cloudinary.uploader.upload(req.file.path, {
33       folder: 'presentation',
34     });
35
36    let document = new Document({
37      name: req.body.name,
38      desc: req.body.desc,
39      studentAllowed: req.body.studentAllowed,
40      staffAllowed: req.body.staffAllowed,
41      docType: "presentation",
42      degree: req.body.degree,
43    });
44
45    document.result.secure_url =
46      cloudinary_id: result.public_id,
47
48    documentSaved = await document.save();
49    res.json(documentSaved);
50  } catch (error) {
51    res.json(error);
52  }
53}
54
55 const insertDocument = async (req, res) => {
56   try {
57     const result = await cloudinary.uploader.upload(req.file.path, {
58       folder: 'document',
59     });
60
61    let document = new Document({
62      name: req.body.name,
63      desc: req.body.desc,
64    });
65
66    documentSaved = await document.save();
67    res.json(documentSaved);
68  } catch (error) {
69    res.json(error);
70  }
71}
72
73 module.exports = {
74   insertDocument,
75   updateDocument,
76   deleteDocument,
77   getDocumentsST,
78   getDocument,
79   insertWorkingSchema,
80   insertPresentation,
81   getDocumentAll,
82 };
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
```

```
File Edit Selection View Go Run Terminal Help
DOSWER
OPEN EDITORS ... documents.js
RESEARCH-PROJECT-MANAGEMENT
  controllers
    admin
      document.js
      controllers
      graph.js
      staff.js
      stGroup.js
      submission.js
      text.js
      user.js
    core
    panel-member
    register
    student
    supervisor
    helpers
    models
      admin
        document.js
        submission.js
        text.js
      panel-member
      register
      student
      supervisor
    routes
    uploads
    env
    ignore
    package-lock.json
    package.json
    server.js
document.js - Research-Project-Management-Tool-backend - Visual Studio Code
controllers > admin > document.js > (6) insertDocument
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
79
80
81
82
83
84
85
86
87
88
89
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
149
150
151
152
153
154
155
156
156
```

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** submission.js - Research Project Management Tool backend - Visual Studio Code
- Editor Area:** The main code editor displays the `submission.js` file. The code defines a Mongoose schema for a "submission" document, requiring fields like name, desc, type, and deadline.
- Explorer Panel:** On the left, the Explorer panel shows the project structure:
  - RESEARCH-PR...** (highlighted)
  - controllers**: controllers.js
  - admin**:
    - document.js
    - graph.js
    - stafflist.js
    - stGroup.js
    - submission.js (highlighted)
    - text.txt
    - userlist.js
  - core**: core.js
  - panel member**: panelMember.js
  - register**: register.js
  - student**: student.js
  - supervisor**: supervisor.js
  - helpers**: helpers.js
  - models**:
    - admin**:
      - document.js
      - submission.js (highlighted)
  - node\_modules**: node\_modules.js
  - routes**: routes.js
  - uploads**: uploads.js
  - .env**: .env
  - .gitignore**: .gitignore
  - package-lock.json**: package-lock.json
  - package.json**: package.json
  - server.js**: server.js
- Bottom Status Bar:** Line 1, Col 1 | Spaces 2 | CRLF | JavaScript | Go Live | ⌘Prettier

graphjs - Research-Project-Management-Tool-backend - Visual Studio Code

```

OPEN EDITORS
graphjs.controllers.admin.js
RESEARCH-PR...
  controllers
    admin
      document.js
      graph.js
      staffList.js
      stGroup.js
      submission.js
      text.txt
      userList.js
    core
    panel member
    register
    student
    supervisor
    helpers
    models
      admin
        document.js
        submission.js
        text.txt
      panel member
      register
      student
      supervisor
    code_modules
    routes
    uploads
    .env
    .gitignore
    package-lock.json
    package.json
    server.js

graphData = async (req, res) => {
  try {
    const count = await Document.find().count();
    const groups = await Group.find().count();
    const newGroups = await Group.find({ panelMember: { $ne: "member" } }).count();
    const submission = await Submission.find().count();
    const research = await Research.find().count();

    const data = {
      document: count,
      groups: groups,
      newGroups: newGroups,
      research: research,
      submission: submission,
    };
    res.json(data);
  } catch (error) {
    res.json(error);
  }
}

module.exports = { graphData };

```

OUTLINE

TIMELINE

In 1 Col 1 Spaces: 2 UTE-8 CR LF JavaScript Go Live Prettier

staffList.js

```

OPEN EDITORS
staffList.js
RESEARCH-PR...
  controllers
    admin
      document.js
      graph.js
      staffList.js
      stGroup.js
      submission.js
      text.txt
      userList.js
    core
    panel member
    register
    student
    supervisor
    helpers
    models
      admin
        document.js
        submission.js
        text.txt
      panel member
      register
      student
      supervisor
    code_modules
    routes
    uploads
    .env
    .gitignore
    package-lock.json
    package.json
    server.js

const Staff = require("../models/panel member/register");

const getStaffs = async (req, res) => {
  try {
    const staff = await Staff.find();
    res.send(staff);
  } catch (error) {
    res.send(error);
  }
};

const getStaff = async (req, res) => {
  const staffId = req.params.id;
  try {
    const staff = await Staff.findOne({ _id: staffId });
    if (staff) {
      res.send(staff);
    } else {
      res.send("staff not found");
    }
  } catch (error) {
    console.log(error);
  }
};

const updateStaff = async (req, res) => {
  const staffId = req.params.id;

  try {
    const staff = await Staff.findOneAndUpdate({ _id: staffId }, req.body);
    res.send(staff);
  } catch (error) {
    res.send(error);
  }
};

const deleteStaff = async (req, res) => {
  const staffId = req.params.id;

  try {
    const staff = await Staff.findOneAndDelete({ _id: staffId });
    if (!staff) {
      res.send({ msg: "Staff Not Found" });
    } else {
      res.send(staff);
    }
  } catch (error) {
    res.send(error);
  }
};

module.exports = {
  getStaff,
  getStaffs,
  deleteStaff,
  updateStaff,
};

```

OUTLINE

TIMELINE

stGroup.js

```

OPEN EDITORS
stGroup.js
RESEARCH-PR...
  controllers
    admin
      document.js
      graph.js
      staffList.js
      stGroup.js
      submission.js
      text.txt
      userList.js
    core
    panel member
    register
    student
    supervisor
    helpers
    models
      admin
        document.js
        submission.js
        text.txt
      panel member
      register
      student
      supervisor
    code_modules
    routes
    uploads
    .env
    .gitignore
    package-lock.json
    package.json
    server.js

const Group = require("../models/student/group");

const updatedGroup = async (req, res) => {
  const groupId = req.params.id;

  try {
    const group = await Group.findOneAndUpdate({ _id: groupId }, req.body);
    res.send(group);
  } catch (error) {
    res.send(error);
  }
};

const deleteGroup = async (req, res) => {
  const groupId = req.params.id;
  try {
    const group = await Group.findOneAndDelete({ _id: groupId });
    if (group) {
      res.send({ msg: "group Not Found" });
    } else {
      res.send(group);
    }
  } catch (error) {
    res.send(error);
  }
};

const getGroups = async (req, res) => {
  try {
    const group = await Group.find({ panelMember: { $ne: "member" } });
    res.send(group);
  } catch (error) {
    res.send(error);
  }
};

const getGroup = async (req, res) => {
  const groupId = req.params.id;
  try {
    const group = await Group.findOne({ _id: groupId });
    if (group) {
      res.json(group);
    } else {
      res.json("group not found");
    }
  } catch (error) {
    res.send(error);
  }
};

const getGroupHome = async (req, res) => {
  const stdId = req.params.id;
  try {
    const group = await Group.findOne({ studentId: stdId });
    if (group) {
      res.json(group);
    } else {
      res.json("group not found");
    }
  } catch (error) {
    res.send(error);
  }
};

module.exports = {
  deletedGroup,
  updatedGroup,
  getGroup,
  getGroupHome,
  getGroups,
};

```

OUTLINE

TIMELINE

```

// submission.js
controllers > admin > submission.js
1 const Submission = require("../models/admin/submission");
2
3 const insertSubmission = async (req, res) => {
4   try {
5     const submission = await Submission.create(req.body);
6     res.send(submission);
7   } catch (error) {
8     res.json({ msg: error.message });
9   }
10 }
11
12 const updateSubmission = async (req, res) => {
13   const subId = req.params.id;
14
15   try {
16     const submission = await Submission.findOneAndUpdate(
17       { _id: subId },
18       { $set: req.body },
19     );
20     res.json(submission);
21   } catch (error) {
22     res.send(error);
23   }
24 }
25
26 const deleteSubmission = async (req, res) => {
27   const subId = req.params.id;
28   try {
29     const submission = await Submission.findByIdAndDelete({ _id: subId });
30     if (!submission) {
31       res.send({ msg: "Submission Not Found" });
32     } else {
33       res.send(submission);
34     }
35   } catch (error) {
36     res.send(error);
37   }
38 }
39
40 const getSubmissions = async (req, res) => {
41   try {
42     const submission = await Submission.find();
43     res.send(submission);
44   } catch (error) {
45     res.send(error);
46   }
47 }
48
49 const getSubmission = async (req, res) => {
50   const subId = req.params.id;
51   try {
52     const submission = await Submission.findOne({ _id: subId });
53     if (submission) {
54       res.json(submission);
55     } else {
56       res.json("Submission not found");
57     }
58   } catch (error) {
59     res.send(error);
60   }
61 }
62
63 module.exports = {
64   insertSubmission,
65   deleteSubmission,
66   updateSubmission,
67   getSubmission,
68   getSubmissions,
69 };

```

```

// userList.js
controllers > admin > userList.js
1 const User = require("../models/register/register");
2 const Panel = require("../models/panelMember/panelMember");
3
4 const getStudents = async (req, res) => {
5   try {
6     const students = await User.find();
7     res.json(students);
8   } catch (error) {
9     console.log(error);
10  }
11 }
12
13 const getPanel = async (req, res) => {
14   try {
15     const panel = await Panel.findById();
16     res.json(panel);
17   } catch (error) {
18     console.log(error);
19   }
20 }
21
22 const getPanels = async (req, res) => {
23   const userId = req.params.id;
24   try {
25     const panel = await Panel.findOne({ _id: userId });
26     if (!panel) {
27       res.json(panel);
28     } else {
29       res.json("Student not found");
30     }
31   } catch (error) {
32     console.log(error);
33   }
34 }
35
36 const getStudent = async (req, res) => {
37   const userId = req.params.id;
38   try {
39     const student = await User.findOne({ _id: userId });
40     if (student) {
41       res.json(student);
42     } else {
43       res.json("Student not found");
44     }
45   } catch (error) {
46     console.log(error);
47   }
48 }
49
50 const updateStudent = async (req, res) => {
51   const studentId = req.params.id;
52   try {
53     const student = await User.findOneAndUpdate({ _id: studentId }, req.body);
54     res.json(student);
55   } catch (error) {
56     res.send(error);
57   }
58 }
59
60 const deleteStudent = async (req, res) => {
61   const studentId = req.params.id;
62   try {
63     const student = await User.findOneAndDelete({ _id: studentId });
64     if (!student) {
65       res.json({ msg: "Student Not Found" });
66     } else {
67       res.json(student);
68     }
69   } catch (error) {
70     res.send(error);
71   }
72 }
73
74 const deletePanel = async (req, res) => {
75   const panelId = req.params.id;
76   try {
77     const panel = await Panel.findOneAndDelete({ _id: panelId });
78     if (!panel) {
79       res.json({ msg: "Panel member Not Found" });
80     } else {
81       res.json(panel);
82     }
83   } catch (error) {
84     res.send(error);
85   }
86 }
87
88 module.exports = {
89   getStudent,
90   getStudents,
91   deleteStudent,
92   updateStudent,
93   getPanel,
94   getPanels,
95   deletePanel,
96 };
97
98

```

```

// userList.js
controllers > admin > userList.js
1 const student = await User.findOneAndUpdate({ _id: studentId }, req.body);
2 res.json(student);
3 } catch (error) {
4   res.send(error);
5 }
6
7 const deleteStudent = async (req, res) => {
8   const studentId = req.params.id;
9   try {
10     const student = await User.findOneAndDelete({ _id: studentId });
11     if (!student) {
12       res.json({ msg: "Student Not Found" });
13     } else {
14       res.json(student);
15     }
16   } catch (error) {
17     res.send(error);
18   }
19 }
20
21 const deletePanel = async (req, res) => {
22   const panelId = req.params.id;
23   try {
24     const panel = await Panel.findOneAndDelete({ _id: panelId });
25     if (!panel) {
26       res.json({ msg: "Panel member Not Found" });
27     } else {
28       res.json(panel);
29     }
30   } catch (error) {
31     res.send(error);
32   }
33 }
34
35 module.exports = {
36   getStudent,
37   getStudents,
38   deleteStudent,
39   updateStudent,
40   getPanel,
41   getPanels,
42   deletePanel,
43 };
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98

```

```
routes > admin > document.js > ...
1 const express = require("express");
2 const router = express.Router();
3 const authenticate = require("../helpers/registerMiddleware/auth");
4 const { upload } = require("../../../../helpers/admin/multer");
5 //const { upload } = require "../../../../helpers/admin/cloudinary";
6 const {
7   insertDocument,
8   updateDocument,
9   deleteDocument,
10  getDocumentsST,
11  getDocument,
12  insertMarkingSchema,
13  insertPresentation,
14  getDocumentAll,
15 } = require("../controllers/admin/document");
16
17 const { insertSubmission } = require("../controllers/student/submission");
18
19 router.post("/insert", upload.single("document"), insertDocument);
20 router.post(
21   "/student/insert/submission",
22   upload.single("document"),
23   insertSubmission
24 );
25
26 router.post("/insert/schema", upload.single("document"), insertMarkingSchema);
27 router.post(
28   "/insert/presentation",
29   upload.single("document"),
30   insertPresentation
31 );
32 router.get("/get", getDocumentsST);
33 router.get("/get/all", getDocumentAll);
34 router.put("/update/:id", updateDocument);
35
36 router.get("/get/:id", getDocument);
37 router.delete("/delete/:id", deleteDocument);
38 module.exports = router;
39
```

```
routes > admin > graph.js > ...
1 const express = require("express");
2 const router = express.Router();
3 const { graphData } = require("../controllers/admin/graph");
4
5 router.get("/graph", graphData);
6 module.exports = router;
7
```

EXPLORER ...

✓ OPEN EDITORS

- graph.js routes\admin
- X submission.js routes\admin

✓ RESEARCH-PR... [+] ⌂ ⌂ ⌂ ⌂

- models
- RESEARCH-PR... [+] ⌂ ⌂ ⌂ ⌂
- admin
  - document.js
  - submission.js
  - text.txt
- panel member
- register
- student
- supervisor
- node\_modules
- routes
  - admin
    - document.js
    - graph.js
    - staff.js
  - submission.js
  - userRouter.js
- auth
- core
- panel member
- student
- supervisor
- uploads
- .env

```
routes > admin > submission.js > ...
1 const express = require("express");
2 const router = express.Router();
3
4 const authenticate = require("../helpers/registerMiddleware/auth");
5
6 const {
7   insertSubmission,
8   deleteSubmission,
9   updateSubmission,
10  getSubmissions,
11  getSubmission,
12 } = require("../controllers/admin/submission");
13
14 router.post("/insert/submission", insertSubmission);
15 router.delete("/delete/submission/:id", deleteSubmission);
16 router.put("/update/submission/:id", updateSubmission);
17 router.get("/get/submission/:id", authenticate, getSubmission);
18 router.get("/get/submission", getSubmissions);
19
20 module.exports = router;
```

✓ OPEN EDITORS

- graph.js routes\admin
- X staff.js routes\admin

✓ RESEARCH-PR... [+] ⌂ ⌂ ⌂ ⌂

- student
- supervisor
- helpers
- admin
- core
- panel member
- registerMiddleware
- student
- supervisor
- models
- admin
  - document.js
  - submission.js
  - text.txt
- panel member
- register
- student
- supervisor
- node\_modules
- routes
- admin
  - document.js
  - graph.js
  - staff.js
- submission.js

```
routes > admin > staff.js > ...
1 const express = require("express");
2 const router = express.Router();
3 const authenticate = require("../helpers/registerMiddleware/auth");
4 const {
5   getStaff,
6   getStaffs,
7   deleteStaff,
8   updateStaff,
9 } = require("../controllers/admin/staffList");
10
11 router.get("/get/staff", authenticate, getStaffs);
12 router.get("/get/staff/:id", authenticate, getStaff);
13 router.put("/update/staff/:id", authenticate, updateStaff);
14 router.delete("/delete/staff/:id", authenticate, deleteStaff);
15
16 module.exports = router;
```

The screenshot shows a development environment in VS Code. The Explorer pane on the left displays a project structure for a Node.js application. The routes folder contains admin, auth, core, panel member, student, supervisor, and uploads subfolders. It also includes .env and .gitignore files. The routes\admin folder contains document.js, graph.js, staff.js, submission.js, and userRouter.js. The userRouter.js file is currently open in the Editor pane, showing its code. The Terminal pane at the bottom is empty.

```
graph.js userRouter.js
```

```
routes > admin > userRouter.js > ...
1 const express = require("express");
2 const router = express.Router();
3 const authenticate = require("../helpers/registerMiddleware/auth");
4 const {
5   getStudent,
6   getStudents,
7   deleteStudent,
8   updateStudent,
9   getPanel,
10  getPanels,
11  deletePanel,
12 } = require("../controllers/admin/userList");
13
14 router.get("/getStudents", getStudents);
15 router.get("/get/panel", getPanel);
16 router.get("/get/panel/:id", getPanels);
17 router.delete("/delete/panel/:id", deletePanel);
18
19 router.get("/getStudent/:id", getStudent);
20 router.put("/update/student/:id", updateStudent);
21 router.delete("/delete/student/:id", deleteStudent);
22
23 module.exports = router;
24
```

# Screenshots of the MongoDB Schemas

The screenshot shows the MongoDB Atlas interface. On the left, a sidebar lists databases (Movies) and collections (researchdb). Under researchdb, a sub-menu shows 'documents' selected, along with other options like groups, panelmembers, researches, results, staffs, studentsubmissions, submissions, supervisors, topics, and users. The main area has tabs for Find, Indexes, Schema Anti-Patterns, Aggregation, and Search Indexes. A 'FILTER' button with the query '{ field: 'value' }' is present. Below it, a section titled 'QUERY RESULTS: 1-4 OF 4' displays a single document:

```
_id: ObjectId("6298a5cca5d2c40aaa6b3106")
name: "af"
desc: "adsx"
studentAllowed: true
staffAllowed: true
docType: "document"
degree: "computing"
document: "https://res.cloudinary.com/dqq3b6lw3/image/upload/v1654171083/document..."
cloudinary_id: "document/hvx1s9k4uma737dkjugg"
createdAt: 2022-06-02T11:58:04.437+00:00
updatedAt: 2022-06-02T11:58:04.437+00:00
__v: 0
```

System Status: All Good

©2022 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales

This screenshot shows the same MongoDB Atlas interface. The sidebar now shows 'researchdb' expanded, with 'submissions' selected under it. Other visible items include documents, groups, panelmembers, researches, results, staffs, studentsubmissions, and users. The 'FILTER' button and 'QUERY RESULTS: 1-2 OF 2' section are identical to the first screenshot. The results show two documents:

```
_id: ObjectId("629896f2d19aed32af76d154")
name: "Testings"
desc: "awwd"
type: "pdf"
deadline: "2022-06-03T21:14"
degree: "computing"
__v: 0

_id: ObjectId("629925828b4230b350694461")
name: "Testing"
```

System Status: All Good

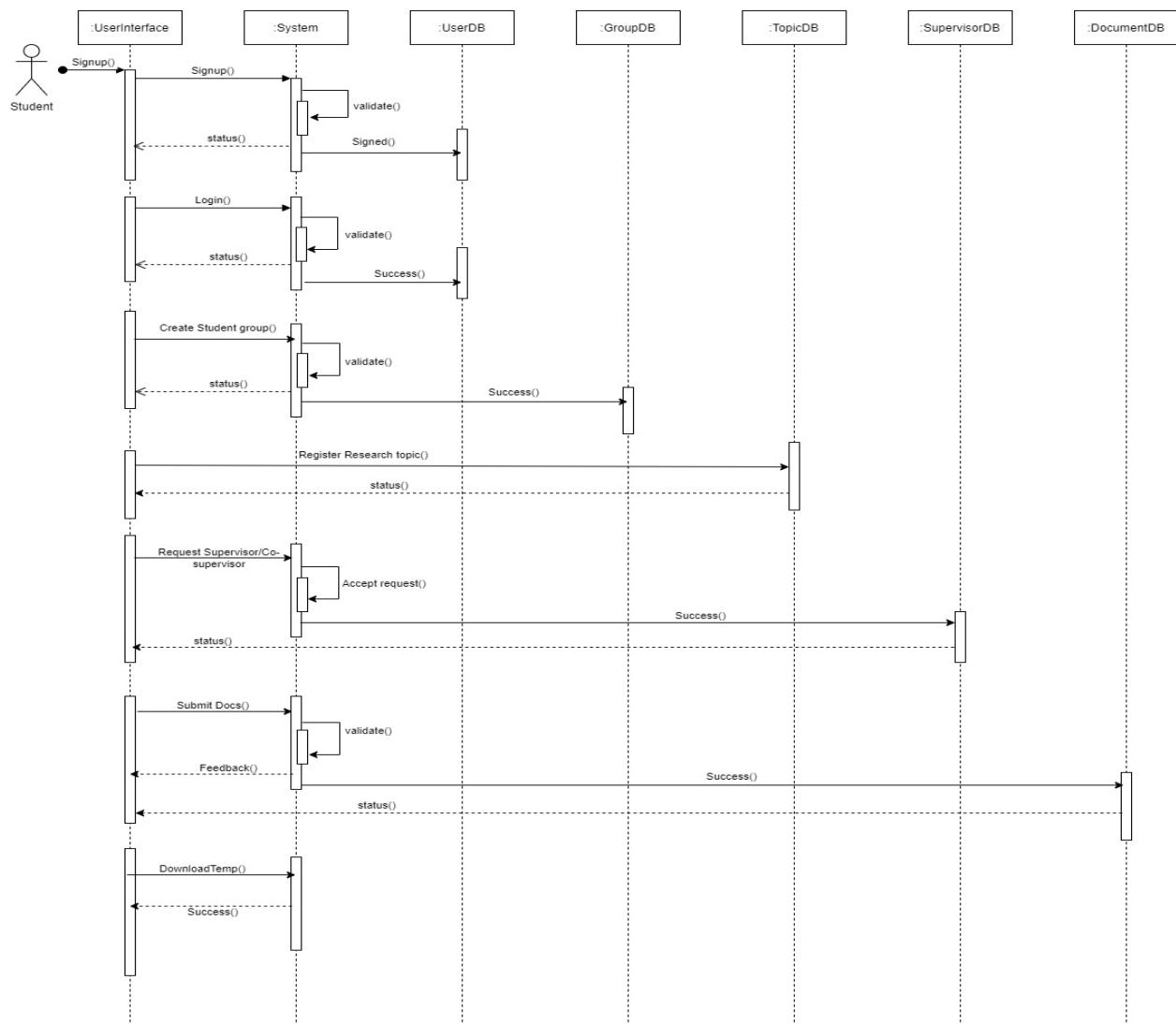
©2022 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales

## Component Description

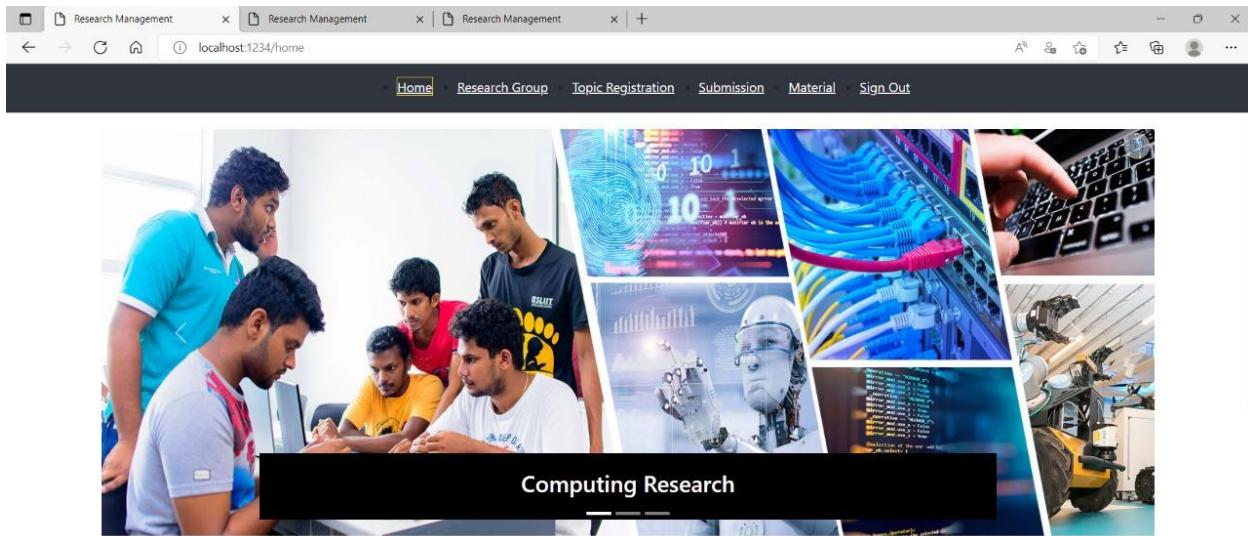
IT20134280 S. Ishalini

Student Role

Description: In this system other main role is a student. Initially student should register and login to the system for access it. They can create a student group and register their research topic. They can request supervisor and co-supervisor from this system who are interested in their research area. Students can download templates which have been published by the administrator. After doing their course work, they can submit their document.



## Screenshots of the Frontend



<b>Registered Group</b>	<b>Registered Topic</b>
-------------------------	-------------------------

A screenshot of a web browser showing a "Student Group Registration" form. The address bar shows "localhost:1234/group/registration". The page has a top navigation bar with links for Home, Research Group, Topic Registration, Submission, Material, and Sign Out. The main content area is a form titled "Student Group Registration" with fields for Group Name, Leader Id, Member 2 Id, Member 3 Id, Member 4 Id, and a "Register" button.

Research Management   Research Management   Research Management

localhost:1234/topic/registration

Home   Research Group   Topic Registration   Submission   Material   Sign Out

### Research Topic Registration

Group Id

Topic

Description

Research Field

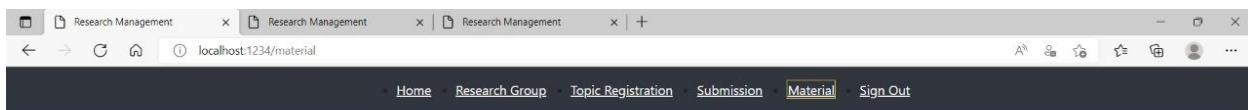
Supervisor Request

Co Supervisor Request

**Register**

### Registered Topic

Topic: IOT Miniaturization: Sensors, CPU, and network  
State: pending  
Status: Evaluation pending



## Document and Presentation Templates

### Research Introduction

Details : Introduce about student research module

[Download](#)

### Student Group Registration

Details : Details about Student Group registration process

[Download](#)

### Student Research module delivery plan

Details : Delivery plan of the research module in 4 th year

[Download](#)

### SRS document marking schema

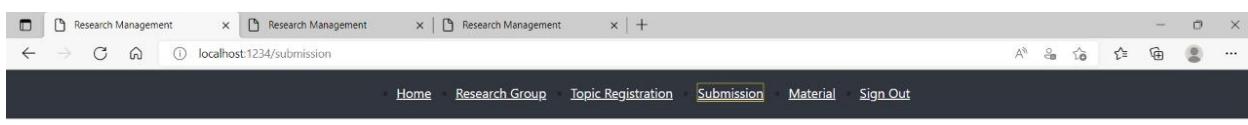
Details : SRS document evaluation sheet

[Download](#)

### SRS document marking schema

Details : SRS document evaluation sheet

[Download](#)



### Submission : Research Plan Submission

Details : Submit PDF document of Research workflow

Upload Type : pdf

Deadline : 2022-06-30T12:00

**Upload Here**

Group Id

Submission Type  
 Document  
 Presentation  
 Final Thesis

No file chosen

### Submission : SRS Document Submission

Details : SRS Document for Research (Download your template)

Upload Type : pdf

Deadline : 2022-07-05T15:00

**Upload Here**

A screenshot of a web browser window titled "Research Management" with three tabs open. The active tab is "localhost:1234/signup". The page has a dark header with "Sign In" and "Sign up" links. The main content area is titled "Student Sign Up" and contains fields for First name, Last name, Email, Password, Contact number, Student ID, and Degree selection (Computing, Business, Engineering). A blue "Sign Up" button is at the bottom.

First name

Last name

Email

Password

Contact number

Student ID

Degree

Computing

Business

Engineering

Sign Up

A screenshot of a web browser window titled "Research Management" with three tabs open. The active tab is "localhost:1234". The page has a dark header with "Sign In" and "Sign up" links. The main content area is titled "Student Sign In" and contains fields for Student Id and Password. A blue "Sign In" button is at the bottom.

Student Id

Password

Sign In

Screenshots/ Code Snippets of the Backend:

The screenshot shows a development environment in VS Code with the following layout:

- EXPLORER**: Shows the project structure with files like `graph.js`, `document.js`, `student`, `supervisor`, `helpers`, `models`, and `routes`.
- OPEN EDITORS**: Displays two code editors:
  - `graph.js`: A file containing logic for routes like `/admin` and `/student`.
  - `document.js`: A file defining a mongoose schema for `student` submissions.
- RESEARCH-PR...**: A search interface showing results for `student` and `document.js`.
- OUTLINE**: A tree view of the project structure.
- TIMELINE**: A history of changes made to the project.

**document.js (Editor)**

```
const mongoose = require("mongoose");
const documentSchema = new mongoose.Schema({
  studentId: {
    type: "string",
    required: [true, "Student ID must be provided"],
  },
  groupId: {
    type: "string",
    required: [true, "Group ID must be provided"],
  },
  topicId: {
    type: "string",
    required: [true, "Topic ID must be provided"],
  },
  submitTime: {
    type: "string",
    required: [true, "Submit ID must be provided"],
  },
  type: {
    type: String,
    required: true,
  },
  document: {
    type: "string",
    required: [true, "student must be provided"],
  },
  cloudinary_id: {
    type: "string",
    required: [true],
  },
  marked: {
    type: Boolean,
    default: false,
  },
  markedBy: {
    type: String,
    default: "pending",
  },
  timestamps: true
});
module.exports = mongoose.model("studentsubmissions", documentSchema);
```

OPEN EDITORS

RESEARCH-PR...

controllers > student > group.js > ...

```

1 const Group = require("../models/student/group");
2
3 const insertGroup = async (req, res) => {
4   try {
5     const group = await Group.create(req.body);
6     res.json(group);
7   } catch (error) {
8     res.send(error);
9   }
10 };
11
12 module.exports = { insertGroup };
13

```

group.js

research.js

submission.js

text.txt

supervisor

helpers

core

panel member

register

student

group.js

research.js

result.js

submission.js

text.txt

supervisor

node\_modules

routes

admin

core

panel member

registerMiddleware

student

supervisor

models

admin

EXPLORER

OPEN EDITORS

RESEARCH-PR...

models > student > group.js > ...

graph.js

group.js

```

1 const mongoose = require("mongoose");
2
3 const groupSchema = new mongoose.Schema({
4   groupName: {
5     type: "string",
6     required: [true, "group name must be provided"],
7   },
8   groupId: {
9     type: "string",
10 },
11   student1: {
12     type: "string",
13     required: [true, "student must be provided"],
14     unique: true,
15   },
16   student2: {
17     type: "string",
18     required: [true, "student must be provided"],
19     unique: true,
20   },
21   student3: {
22     type: "string",
23     required: [true, "student must be provided"],
24     unique: true,
25   },
26   student4: {
27     type: "string",
28     required: [true, "student must be provided"],
29     unique: true,
30   },
31   panelMember1: {
32     type: "string",
33     default: "member",
34   },
35   panelMember2: {
36     type: "string",
37     default: "member",
38   },
39   panelMember3: {
40     type: "string",
41     default: "member",
42   },
43 });
44
45 module.exports = new mongoose.model("group", groupSchema);
46

```

OUTLINE

The image shows a code editor interface with two main panes. The left pane displays a file tree (Outline) and the right pane shows a code editor with a schema definition.

**Outline (Left Pane):**

- OPEN EDITORS (1 UNSAVED)
  - graph.js
  - group.js
  - models/student
  - result.js
  - resultSchema
- RESEARCH-PROJECT-MANAGEMENT
  - admin
    - document.js
    - graph.js
    - staffList.js
    - stGroup.js
    - submission.js
    - text.txt
    - userList.js
  - core
  - panel member
  - register
  - student
  - supervisor
  - helpers
    - admin
    - core
    - panel member
    - registerMiddleware
    - student
    - supervisor
  - models
    - admin
    - panel member
    - register
  - student
    - document.js
    - group.js
    - research.js
    - result.js
    - submission.js
    - text.txt
  - supervisor
  - node\_modules
  - routes
    - admin
      - document.js
      - graph.js
      - staff.js
      - submission.js
      - userRouter.js
    - auth
    - core
    - panel member
    - student
    - supervisor
  - uploads
  - .env
  - .gitignore
  - package-lock.json
  - package.json
  - server.js

- OUTLINE
- TIMELINE

```
models > student > result.js > resultSchema
1  const mongoose = require("mongoose");
2
3  const resultSchema = new mongoose.Schema({
4    |
5    groupId: {
6      type: "string",
7      unique: true,
8    },
9    studentId: {
10      studentId: {
11        type: String,
12        default: "No ID",
13      },
14      documentMarks: {
15        type: Number,
16        default: 0,
17      },
18      presentationMarks: {
19        type: Number,
20        default: 0,
21      },
22      finalThesisMarks: {
23        type: Number,
24        default: 0,
25      },
26    },
27    student2: {
28      studentId: {
29        type: String,
30        default: "No ID",
31      },
32      documentMarks: {
33        type: Number,
34        default: 0,
35      },
36      presentationMarks: {
37        type: Number,
38        default: 0,
39      },
40      finalThesisMarks: {
41        type: Number,
42        default: 0,
43      },
44    },
45    student3: {
46      studentId: {
47        type: String,
48        default: "No ID",
49      },
50      documentMarks: {
51        type: Number,
52        default: 0,
53      },
54      presentationMarks: {
55        type: Number,
56        default: 0,
57      },
58      finalThesisMarks: {
59        type: Number,
60        default: 0,
61      },
62    },
63    student4: {
64      studentId: {
65        type: String,
66        default: "No ID",
67      },
68      documentMarks: {
69        type: Number,
70        default: 0,
71      },
72      presentationMarks: {
73        type: Number,
74        default: 0,
75      },
76      finalThesisMarks: {
77        type: Number,
78        default: 0,
79      },
80    },
81  },
82  {
83    timestamps: true
84  };
85});
```

The screenshot shows a code editor interface with the following details:

- OPEN EDITORS:** A sidebar listing files and folders:
  - models > student > submission.js
  - graph.js routes\admin
  - group.js models\student
  - result.js models\student
  - X submission.js models\st...
- RESEARCH-PR...**: A search bar with a dropdown menu showing recent results.
- File Content (submission.js):**

```
1 const mongoose = require("mongoose");
2
3 const submissionSchema = new mongoose.Schema(
4   {
5     studentId: {
6       type: "string",
7     },
8     groupId: {
9       type: "string",
10    },
11    type: {
12      type: "string",
13    },
14
15    topicId: {
16      type: "string",
17    },
18    submitTime: {
19      type: "string",
20    },
21    document: {
22      type: "string",
23      default: "placeholder.jpg",
24    },
25    marked: {
26      type: "boolean",
27      default: false,
28    },
29    markedBy: {
30      type: "string",
31      default: "pending",
32    },
33    clouddinary_id: {
34      type: "string",
35    },
36  },
37  { timestamps: true }
38 );
39
40 module.exports = new mongoose.model("studentSubmission", submissionSchema);
41
```

The screenshot shows a code editor interface with the following details:

- EXPLORER** sidebar: Shows the project structure with files like graph.js, group.js, and research.js.
- OPEN EDITORS** sidebar: Shows the current open files: graph.js, group.js, and research.js.
- research.js** content (highlighted in the editor):

```
models > student > research.js > ...
4   topicId: {
5     type: String,
6     default: "1234",
7   },
8   groupId: {
9     type: String,
10    required: [true, "Group id required"],
11 },
12   field: {
13     type: String,
14     required: [true, "Field required"],
15   },
16   studentId: {
17     type: String,
18   },
19   topic: {
20     type: String,
21     required: [true, "Topic required"],
22   },
23   description: {
24     type: String,
25     required: [true, "Description required"],
26   },
27   supervisorId: {
28     type: String,
29     required: true,
30     default: "Supervisor ID",
31   },
32   supervisorName: {
33     type: String,
34     required: true,
35     default: "Supervisor name",
36   },
37   role: {
38     type: String,
39     default: "Supervisor",
40     enum: ["Supervisor", "Co-supervisor"],
41   },
42   state: {
43     type: String,
44     default: "pending",
45     enum: ["accepted", "rejected", "pending"],
46   },
47   panelMemberId: {
48     type: String,
49     required: true,
50     default: "Panel member ID",
51   },
52   panelMemberName: {
53     type: String,
54     required: true,
55     default: "Panel member name",
56   },
57   evaluated: {
58     type: Boolean,
59     default: false,
60   },
61   evaluation: {
62     type: String,
63     default: "Evaluation pending",
64   },
65   supervisor: {
66     type: Boolean,
67     default: false,
68   },
69   coSupervisor: {
70     type: Boolean,
71     default: false,
72   },
73   },
74   },
75   { timestamps: true }
76   );
77
78 module.exports = new mongoose.model("topic", topicSchema);
```

```
research.js
controllers > student > research.js > ...
1 const Research = require("../models/student/research");
2 const Result = require("../models/student/result");
3
4 const insertResearch = async (req, res) => {
5   try {
6     const research = await Research.create(req.body);
7     res.send(research);
8   } catch (error) {
9     res.send(error);
10 }
11
12 const updateResearch = async (req, res) => {
13   const researchId = req.params.id;
14
15   try {
16     const research = await Research.findOneAndUpdate(
17       { _id: researchId },
18       req.body,
19     );
20     res.send(research);
21   } catch (error) {
22     res.send(error);
23   }
24 };
25
26 const deleteResearch = async (req, res) => {
27   const researchId = req.params.id;
28
29   try {
30     const research = await Research.findOneAndDelete({ _id: researchId });
31     if (!research) {
32       res.send({ msg: "research Not Found" });
33     } else {
34       res.send(research);
35     }
36   } catch (error) {
37     res.send(error);
38   }
39 };
40
41 const getResearchs = async (req, res) => {
42   try {
43     const research = await Research.find();
44     res.send(research);
45   } catch (error) {
46     res.send(error);
47   }
48 };
49
50 const getResearch = async (req, res) => {
51   const researchId = req.params.id;
52
53   try {
54     const research = await Research.findOne({ studentId: researchId });
55     if (research) {
56       res.json(research);
57     } else {
58       res.json("research not found");
59     }
60   } catch (error) {
61     res.send(error);
62   }
63 };
64
65 const getResult = async (req, res) => {
66   const groupId = req.params.id;
67
68   try {
69     const result = await Result.find({ groupId: groupId });
70     if (result) {
71       res.json(result);
72     } else {
73       res.json("no result found");
74     }
75   } catch (error) {
76     res.send(error);
77   }
78 };
79
80 module.exports = {
81   insertResearch,
82   deleteResearch,
83   updateResearch,
84   getResearchs,
85   getResearch,
86   getResult,
87 }
```

OPEN EDITORS

RESEARCH-PR... controllers student submission.js

```

1 const Submission = require("../models/student/submission");
2 const cloudinary = require("../helpers/admin/cloudinary");
3
4 const insertSubmission = async (req, res) => {
5   try {
6     const result = await cloudinary.uploader.upload(req.file.path, {
7       folder: "studentSubmission",
8     });
9
10    let document = new Submission({
11      studentId: req.body.studentId,
12      groupId: req.body.groupId,
13      topicId: req.body.topicId,
14      type: req.body.type,
15      submitTime: req.body.submitTime,
16      document: result.secure_url,
17      cloudinary_id: result.public_id,
18    });
19
20    console.log(document);
21    documentSaved = await document.save();
22    res.json(documentSaved);
23  } catch (error) {
24    res.json(error);
25  }
26};
27
28 module.exports = { insertSubmission };
29

```

models student

node\_modules routes

routes admin

student

OPEN EDITORS

RESEARCH-PR... routes\student group.js

```

1 const express = require("express");
2 const router = express.Router();
3
4 const authenticate = require("../helpers/registerMiddleware/auth");
5
6 const {
7   deleteGroup,
8   updateGroup,
9   getGroups,
10  getGroup,
11  getGroupHome,
12 } = require("../controllers/admin/stGroup");
13
14 const { insertGroup } = require("../controllers/student/group");
15
16 router.post("/insert/group", insertGroup);
17 router.delete("/delete/group/:id", deleteGroup);
18 router.put("/update/group/:id", updateGroup);
19 router.get("/get/group/:id", getGroup);
20 router.get("/get/group/home/:id", getGroupHome);
21
22 router.get("/get/group", getGroups);
23
24 module.exports = router;
25

```

group.js research.js text.txt supervisor

uploads .env .gitignore

```

1 const express = require("express");
2 const router = express.Router();
3 const authenticate = require("../helpers/registerMiddleware/auth");
4 const {
5   insertResearch,
6   deleteResearch,
7   updateResearch,
8   getResearchs,
9   getResearch,
10  getResult,
11 } = require("../controllers/student/research");
12
13 router.post("/insert/research", insertResearch);
14 router.delete("/delete/research/:id", authenticate, deleteResearch);
15 router.put("/update/research/:id", authenticate, updateResearch);
16 router.get("/get/research/:id", getResearch);
17 router.get("/get/research", getResearch);
18 router.get("/get/result/:id", getResult);
19
20 module.exports = router;
21

```

The Explorer sidebar shows the project structure:

- RESEARCH-PR...
- controllers
  - admin
  - core
  - panel member
  - register
  - student
  - supervisor
- helpers
  - admin
  - core
  - panel member
  - registerMiddleware
  - student
  - supervisor
- models
  - admin
  - panel member
  - register
  - student
  - supervisor
- node\_modules
- routes
  - admin
  - auth
  - core
  - panel member
  - student
    - group.js
    - research.js
    - text.txt
  - supervisor
- uploads
- .env
- .gitignore
- package-lock.json
- package.json
- server.js

## Screenshots of the MongoDB Schemas

**researchdb**

- documents
- groups
- panelmembers
- researches
- results
- staffs
- studentsubmissions**
- submissions
- supervisors
- topics
- users

**studentsubmissions**

```

_id: ObjectId("6297ce5c7c464a044da2e7d1")
studentId: "2"
groupId: "qu"
topicId: ""
submTitle: ""
document: "https://res.cloudinary.com/dqg3b6lw3/image/upload/v1654115934/studentS..."
cloudinary_id: "studentSubmission/cm8jdnzu1jigbrz17js"
createdAt: 2022-06-01T20:38:52.814+00:00
updatedAt: 2022-06-01T20:38:52.814+00:00
__v: 0

```

## researchdb

documents

### groups

panelmembers

researches

results

staffs

studentsubmissions

submissions

supervisors

topics

users

FILTER { field: 'value' }

QUERY RESULTS: 1-4 OF 4

```
_id: ObjectId("6298d8eaffbf22af38a12e6c")
groupName: "Error 404"
groupId: "RG8845"
student1: "IT202201021"
student2: "IT202201121"
student3: "IT202201012"
student4: "IT202201032"
panelMember1: "PMR0001"
panelMember2: "PMR0002"
panelMember3: "PMR0003"
__v: 0
```

id: ObjectId("6298d8eaffbf22af38a12e6c")

System Status: All Good

©2022 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales

## researchdb

documents

groups

panelmembers

researches

results

staffs

studentsubmissions

submissions

supervisors

### topics

users

FILTER { field: 'value' }

QUERY RESULTS: 1-4 OF 4

```
_id: ObjectId("629786f63f7122f06988c71b")
topicId: "1234"
groupId: "RG8845"
field: "Computational biology"
studentId: "2"
topic: "1"
description: "Lorem Ipsum is simply dummy text of the printing and typesetting industry. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum."
supervisorId: "SVR0001"
supervisorName: "Saoirse Ronan"
role: "supervisor"
state: "accepted"
panelMemberId: "PMR0001"
panelMemberName: "Saoirse Ronan"
evaluated: true
evaluation: "Evaluation pending"
createdAt: 2022-06-01T15:34:14.912+00:00
```

System Status: All Good

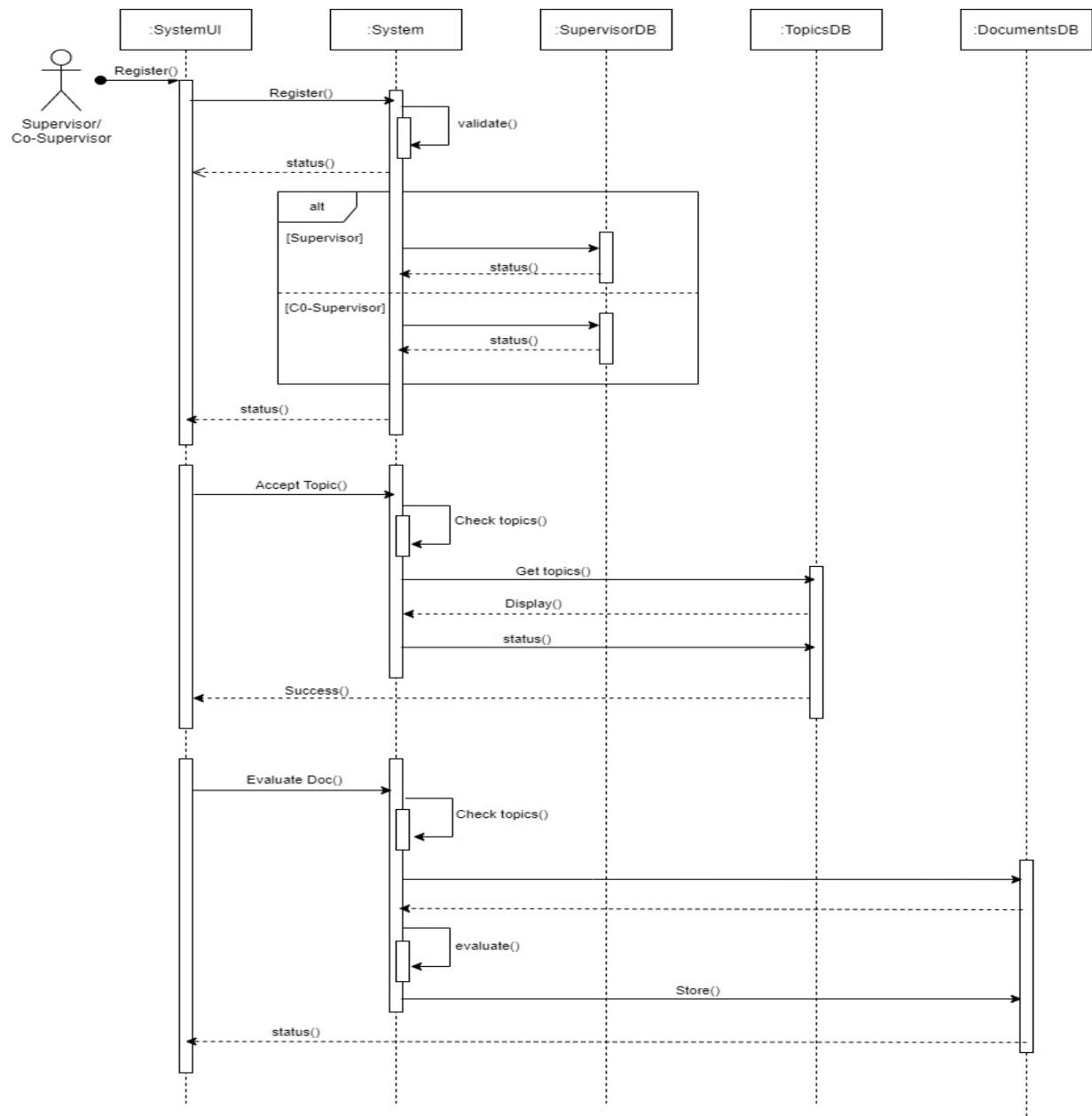
©2022 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales

## Component Description

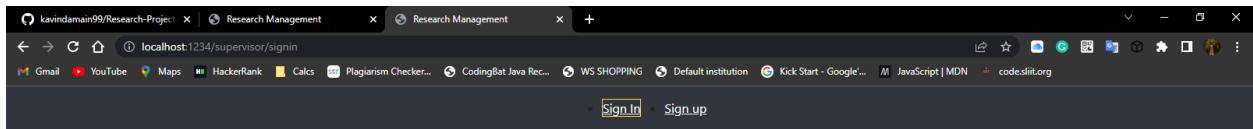
IT20227036 Moremada M.M.M.B

Supervisor / Co-supervisor Role

Description: Generally, supervisor and co-supervisor has same functions in the system. All should register and login to the system initially. The main task of supervisor and co-supervisor is accepting student's topics and give feedback for them and also communicate with them. They Should evaluate Documents submitted by groups using the provided marking scheme.



## Screenshots of the Frontend

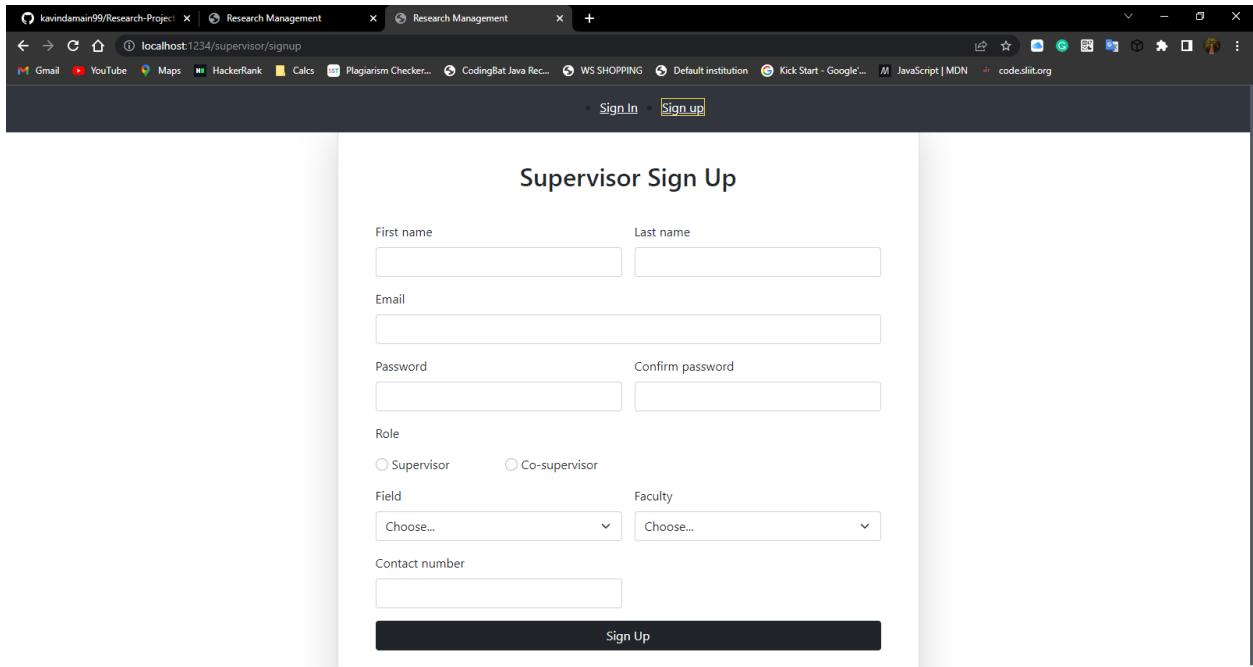


### Supervisor Sign In

Email

Password

**Sign In**



The screenshot shows a web application interface for managing research topics. At the top, there are three tabs: 'Pending' (1), 'Accepted' (2), and 'Rejected' (1). Each tab displays a list of topics with their details and submission history.

- Pending:** IOT Service Orchestration and Routing
- Accepted:** IOT Miniaturization: Sensors, CPU, and network
- Rejected:** IoT applications

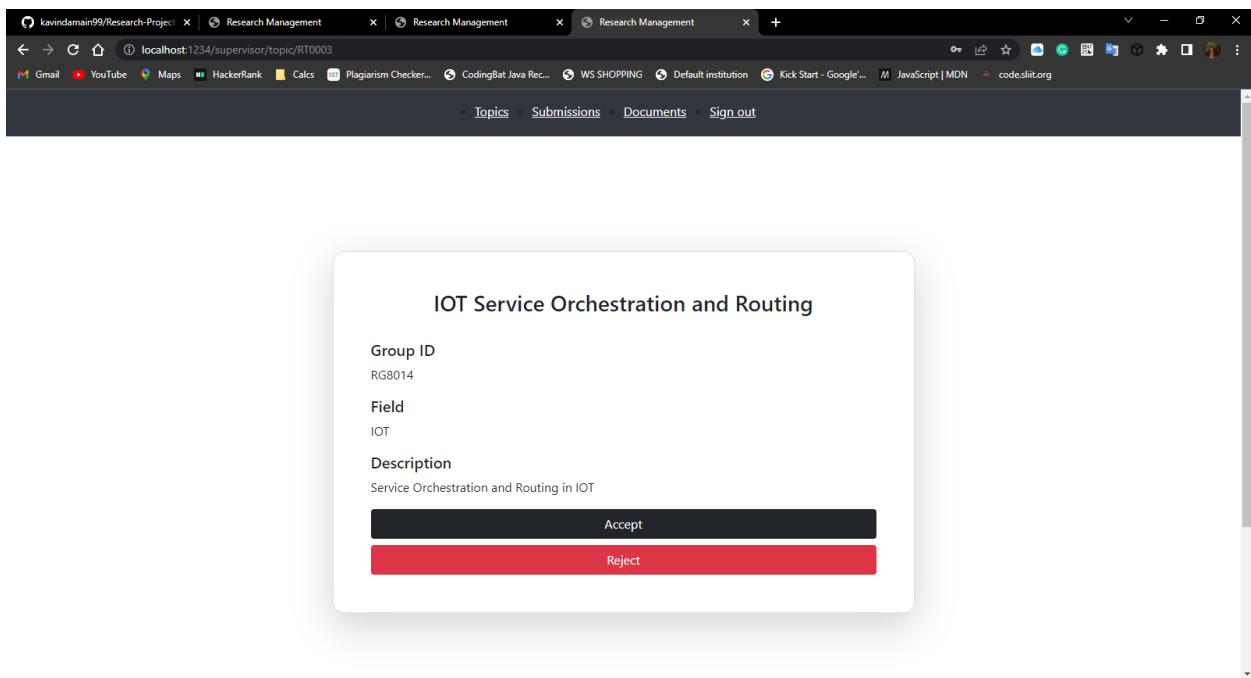
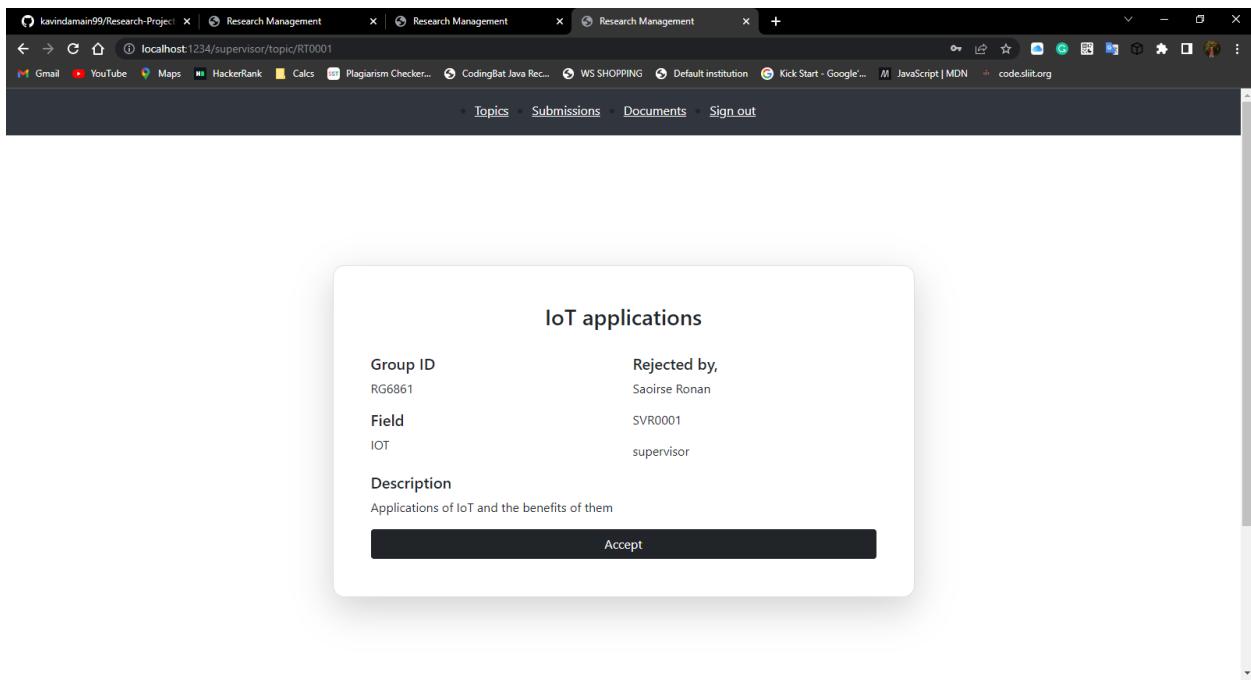
Each topic entry includes a 'View details' button.

Copyright © 2022 All Rights Reserved.

The screenshot shows a detailed view of a research topic titled "IOT Miniaturization: Sensors, CPU, and network". The page includes the following information:

- Group ID:** RG8013
- Field:** IOT
- Description:** IOT Miniaturization: Sensors, CPU, and network
- Accepted by:** Saoirse Ronan (SVR0001, supervisor)

A red 'Reject' button is visible at the bottom of the topic details.



Screenshot of a web browser showing a list of research documents under the 'Documents' tab.

The page title is 'localhost:1234/supervisor/documents'. The navigation bar includes 'Topics', 'Submissions', 'Documents', 'Sign out', and 'Chat'.

The main content area displays five document entries:

- Research Introduction document**: Introduce about student research module. Uploaded on 2022-06-04 at 09:44:52. View Document button.
- Student Research module delivery plan document**: Delivery plan of the research module in 4 th year. Uploaded on 2022-06-04 at 09:46:47. View Document button.
- SRS document marking schema marking**: SRS document evaluation sheet. Uploaded on 2022-06-04 at 09:51:51. View Document button.
- Student Group Registration document**: Details about Student Group registration process. Uploaded on 2022-06-04 at 09:45:52. View Document button.
- SRS document marking schema marking**: SRS document evaluation sheet. Uploaded on 2022-06-04 at 09:51:51. View Document button.

Screenshot of a web browser showing a list of student documents under the 'Documents' tab.

The page title is 'localhost:1234/supervisor/student/documents'. The navigation bar includes 'Topics', 'Submissions', 'Documents', 'Sign out', and 'Chat'. A 'Settings' icon is also present in the top right.

The main content area displays two document entries:

- RG8013 document**: Submitted on 2022-06-04 at 16:09:51. View Document button.
- Final Theses**: 0 items.

At the bottom, a dark footer bar contains the text 'Copyright © 2022 All Rights Reserved.'

kavindomain99/Research-Project | Research Management | Research Management | Settings

localhost:1234/supervisor/student/document/629b83cf8a7e0d1421051fc7/RG8013

Gmail YouTube Maps HackerRank Calcs Plagiarism Checker... CodingBat Java Rec... WS SHOPPING Default institution Kick Start - Google... JavaScript | MDN codes.sliit.org

Topics Submissions Documents Sign out Chat

### Document

Group name - IOT Heroes  
Group ID - RG8013  
[View document](#)

Student1 - IT20312134

Student2 - IT20331232

Student3 - IT20341532

Student4 - IT20341262

kavindomain99/Research-Project | Research Management | Research Management | Settings

localhost:1234/supervisor/chat

Gmail YouTube Maps HackerRank Calcs Plagiarism Checker... CodingBat Java Rec... WS SHOPPING Default institution Kick Start - Google... JavaScript | MDN codes.sliit.org

Topics Submissions Documents Sign out Chat

Search groups

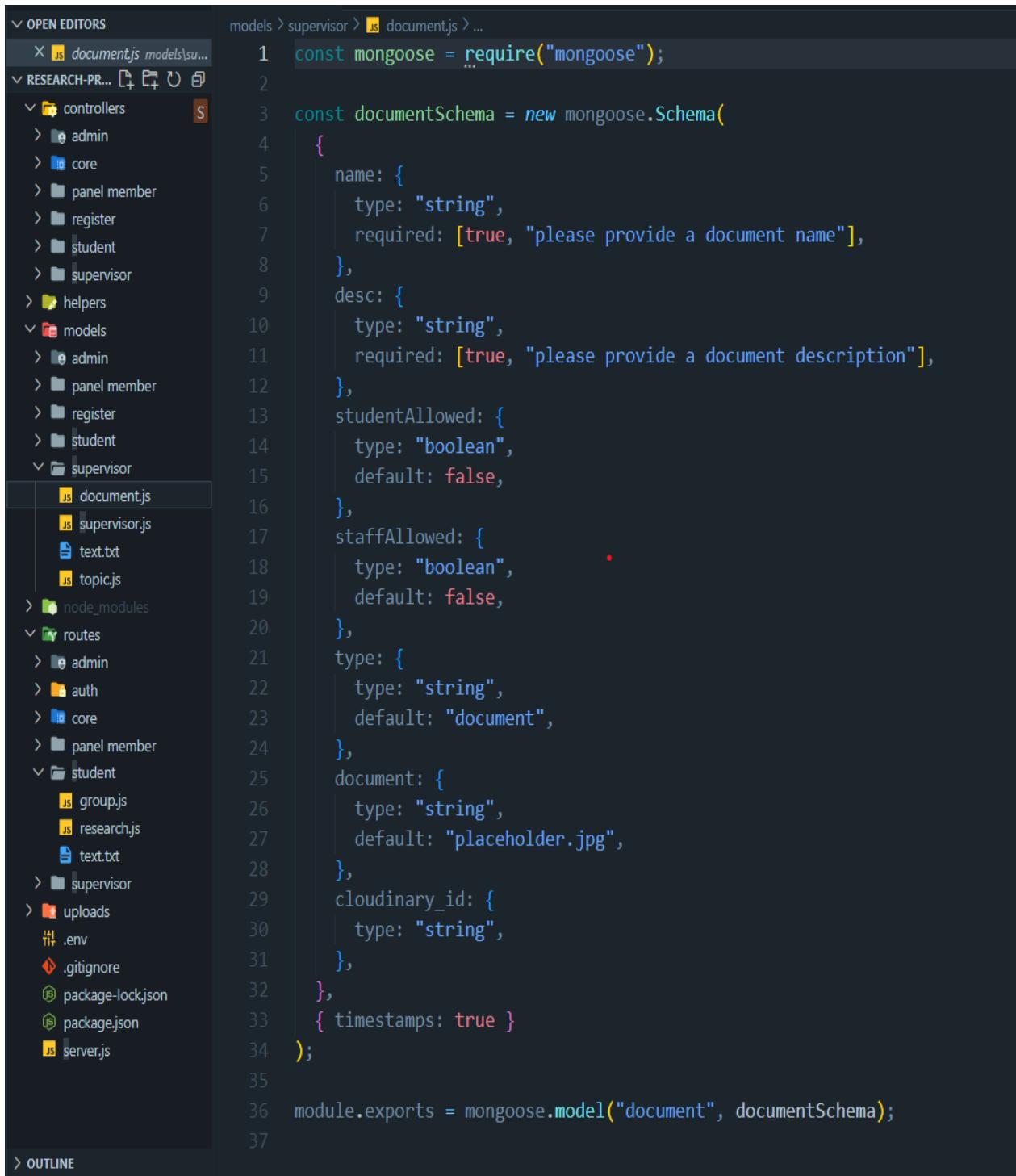
- RG0001
- RG0003
- RG0006
- RG0006
- RG0008
- RG0009
- RG0011

• Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has

• Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has

Type something...

## Screenshots/ Code Snippets of the Backend:



The screenshot shows a code editor interface with the following details:

- OPEN EDITORS:** A list of open files including `document.js`, `supervisor.js`, and `server.js`.
- RESEARCH-PR...**: A search bar with a magnifying glass icon.
- File Explorer:** Shows the project structure:
  - `models`:
    - `supervisor`:
      - `document.js` (selected)
      - `supervisor.js`
      - `text.txt`
      - `topic.js`
    - `node_modules`
    - `routes`:
      - `admin`
      - `auth`
      - `core`
      - `panel member`
      - `student`:
        - `group.js`
        - `research.js`
        - `text.txt`
      - `supervisor`
    - `uploads`
    - `.env`
    - `.gitignore`
    - `package-lock.json`
    - `package.json`
    - `server.js`
  - Document Preview:** The content of `document.js` is displayed:

```
1 const mongoose = require("mongoose");
2
3 const documentSchema = new mongoose.Schema(
4   {
5     name: {
6       type: "string",
7       required: [true, "please provide a document name"],
8     },
9     desc: {
10       type: "string",
11       required: [true, "please provide a document description"],
12     },
13     studentAllowed: {
14       type: "boolean",
15       default: false,
16     },
17     staffAllowed: {
18       type: "boolean",
19       default: false,
20     },
21     type: {
22       type: "string",
23       default: "document",
24     },
25     document: {
26       type: "string",
27       default: "placeholder.jpg",
28     },
29     cloudinary_id: {
30       type: "string",
31     },
32     { timestamps: true }
33   );
34
35 module.exports = mongoose.model("document", documentSchema);
36
```
  - Bottom Bar:** Includes buttons for `OUTLINE`, `SEARCH`, and `REFRESH`.

The screenshot shows the VS Code interface with the following details:

- Explorer View:** Shows the project structure. The root folder contains files like `node_modules`, `routes`, `server.js`, and `uploads`. Inside `models`, there are `admin`, `core`, `panel member`, `register`, `student`, and `supervisor` subfolders. The `supervisor` folder contains `document.js` and `supervisor.js`.
- Editor View:** Displays the `supervisor.js` file content. The code defines a Mongoose schema for a `supervisor` document. It includes fields for `id`, `firstName`, `lastName`, `role`, `email`, `password`, `field`, `degree`, and `contactNumber`. The `email` field has a regular expression validation for email addresses. The `password` field requires at least 8 characters. The `contactNumber` field requires a 10-digit number.

```
const mongoose = require("mongoose");
const supervisorSchema = new mongoose.Schema(
{
    id: {
        type: String,
        required: [true, "Id required"],
    },
    firstName: {
        type: String,
        required: [true, "Please provide a first name"],
    },
    lastName: {
        type: String,
        required: [true, "Please provide a last name"],
    },
    role: {
        type: String,
        default: "supervisor",
        enum: ["supervisor", "co-supervisor"],
        required: [true, "Please select a role"],
    },
    email: {
        type: String,
        required: [true, "Please provide an email"],
        match: [
            /^[a-zA-Z0-9.!#$%&'*+=^_`{|~}]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*$/,
            "Please provide an valid email",
        ],
    },
    password: {
        type: String,
        required: [true, "Please provide a password"],
        minlength: [8, "Password must have atleast 8 characters"],
    },
    field: {
        type: String,
        required: [true, "Please select a field"],
    },
    degree: {
        type: String,
        enum: ["computing", "engineering", "business"],
        required: [true, "Please select a degree"],
    },
    contactNumber: {
        type: String,
        required: [true, "Contact number required"],
        minlength: [10, "Contact number must have 10 digits"],
        maxlength: [10, "Contact number must have 10 digits"],
    },
},
{ timestamps: true }
);
module.exports = mongoose.model("supervisor", supervisorSchema);
```

```
const mongoose = require("mongoose");

const supervisorSchema = new mongoose.Schema({
  id: {
    type: String,
    required: [true, "Id required"],
  },
  firstName: {
    type: String,
    required: [true, "Please provide a first name"],
  },
  lastName: {
    type: String,
    required: [true, "Please provide a last name"],
  },
  role: {
    type: String,
    default: "supervisor",
    enum: ["supervisor", "co-supervisor"],
    required: [true, "Please select a role"],
  },
  email: {
    type: String,
    required: [true, "Please provide an email"],
    match: [
      /^[a-zA-Z0-9.!#$%&'*+=?^_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*$/,
      "Please provide a valid email",
    ],
  },
  password: {
    type: String,
    required: [true, "Please provide a password"],
    minlength: [8, "Password must have atleast 8 characters"],
  },
  field: {
    type: String,
    required: [true, "Please select a field"],
  },
  degree: {
    type: String,
    enum: ["computing", "engineering", "business"],
    required: [true, "Please select a degree"],
  },
  contactNumber: {
    type: String,
    required: [true, "Contact number required"],
    minlength: [10, "Contact number must have 10 digits"],
    maxlength: [10, "Contact number must have 10 digits"],
  },
  timestamps: true
});

module.exports = mongoose.model("supervisor", supervisorSchema);
```

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER**: Shows the project structure with files like supervisor.js, supervisorSchema, and supervisorModel.
- OPEN EDITORS**: Shows supervisor.js open in the editor.
- models > supervisor > supervisor.js**: The current file being edited.
- Content of supervisor.js:**

```
1 const mongoose = require("mongoose");
2
3 const supervisorSchema = new mongoose.Schema(
4 {
5   id: {
6     type: String,
7     required: [true, "Id required"],
8   },
9   firstName: {
10     type: String,
11     required: [true, "Please provide a first name"],
12   },
13   lastName: {
14     type: String,
15     required: [true, "Please provide a last name"],
16   },
17   role: {
18     type: String,
19     default: "supervisor",
20     enum: ["supervisor", "co-supervisor"],
21     required: [true, "Please select a role"],
22   },
23   email: {
24     type: String,
25     required: [true, "Please provide an email"],
26     match: [
27       /^[a-zA-Z0-9.!#$%&*+=?^`{|~}]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*$/,
28       "Please provide a valid email",
29     ],
30   },
31   password: {
32     type: String,
33     required: [true, "Please provide a password"],
34     minlength: [8, "Password must have atleast 8 characters"],
35   },
36   field: {
37     type: String,
38     required: [true, "Please select a field"],
39   },
40   degree: {
41     type: String,
42     enum: ["computing", "engineering", "business"],
43     required: [true, "Please select a degree"],
44   },
45   contactNumber: {
46     type: String,
47     required: [true, "Contact number required"],
48     minlength: [10, "Contact number must have 10 digits"],
49     maxlength: [10, "Contact number must have 10 digits"],
50   },
51 },
52 { timestamps: true });
53
54
55 module.exports = mongoose.model("supervisor", supervisorSchema);
```

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER**: Shows the project structure with files like supervisor.js, supervisorSchema, and supervisorModel.
- OPEN EDITORS**: Displays the supervisor.js file content.
- File Content (supervisor.js)**:

```
1 const mongoose = require("mongoose");
2
3 const supervisorSchema = new mongoose.Schema(
4 {
5   id: {
6     type: String,
7     required: [true, "Id required"],
8   },
9   firstName: {
10     type: String,
11     required: [true, "Please provide a first name"],
12   },
13   lastName: {
14     type: String,
15     required: [true, "Please provide a last name"],
16   },
17   role: {
18     type: String,
19     default: "supervisor",
20     enum: ["supervisor", "co-supervisor"],
21     required: [true, "Please select a role"],
22   },
23   email: {
24     type: String,
25     required: [true, "Please provide an email"],
26     match: [
27       /^[a-zA-Z0-9.!#$%&*+=?^`{|}~]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*$/,
28       "Please provide a valid email",
29     ],
30   },
31   password: {
32     type: String,
33     required: [true, "Please provide a password"],
34     minlength: [8, "Password must have atleast 8 characters"],
35   },
36   field: {
37     type: String,
38     required: [true, "Please select a field"],
39   },
40   degree: {
41     type: String,
42     enum: ["computing", "engineering", "business"],
43     required: [true, "Please select a degree"],
44   },
45   contactNumber: {
46     type: String,
47     required: [true, "Contact number required"],
48     minlength: [10, "Contact number must have 10 digits"],
49     maxlength: [10, "Contact number must have 10 digits"],
50   },
51 },
52 { timestamps: true });
53
54 module.exports = mongoose.model("supervisor", supervisorSchema);
```

```
controllers > supervisor > supervisor.js > ...
69      }
70    } catch (error) {
71      return response
72        .status(400)
73        .json({ message: "Field not found", error: error });
74    }
75  };
76
77 /**
78 * update the state of topic to accepted or rejected
79 */
80 const updatestate = async (request, response) => {
81   const state = request.params.state;
82   const evaluated = request.topic.evaluated;
83
84   if (state === "accepted" || state === "rejected") {
85     if (!evaluated) {
86       const topic = request.topic;
87       const supervisor = request.supervisor;
88
89       topic.state = state;
90       topic.supervisorId = supervisor.id;
91       topic.supervisorName = supervisor.firstName + " " + supervisor.lastName;
92       topic.role = supervisor.role;
93
94       const updatedTopic = new Topic(topic);
95       try {
96         await updatedTopic.save();
97         return response.status(200).json({ message: "Update succesfull" });
98       } catch (error) {
99         console.log(error);
100        return response
101          .status(400)
102          .json({ message: "Update failed", error: error });
103      }
104    } else {
105      return response
106        .status(400)
107        .json({ error: "Cannot change state after evaluation" });
108    }
109  } else {
110    return response.status(400).json({ error: "Invalid state" });
111  }
112}
113
114 /**
115 * get the documents uploaded by admin
116 */
117 const getdocuments = async (request, response) => {
118   try {
119     const documents = await Document.find({ staffAllowed: true });
120     return response.status(200).json(documents);
121   } catch (error) {
122     return response
123       .status(400)
124       .json({ message: "Unable to fetch documents", error: error });
125   }
126 }
127
128 /**
129 * middleware
130 * get the group IDs that the supervisor is assigned to
131 */
132 const groupIDs = async (request, response, next) => {
133   try {
134     const groupIDs = await Topic.find({
135       supervisorId: request.supervisor.id,
136     }).select("groupID -_id");
137
138     var IDArray = [];

```

```
var IDArray = [];

groupIDs.forEach((id) => {
  IDArray.push(id.groupId);
});

request.groupIDs = IDArray;
next();
} catch (error) {
  return response
    .status(400)
    .json({ error: error, message: "Unable to fetch relevant group IDs" });
}

/**
 * get the groups that the supervisor is assigned to
 */
const getAssignedGroups = async (request, response) => {
  try {
    const groups = await Group.find({ groupId: request.groupIDs });
    return response.status(200).json(groups);
  } catch (error) {
    return response
      .status(400)
      .json({ error: error, message: "Unable to fetch relevant groups" });
  }
};

/**
 * get submissions from the students groups that the supervisor is assigned to
 */
const getstudentDocuments = async (request, response) => {
  const type = request.params.type;

  const filter = {
    groupId: request.groupIDs,
    type: type,
    marked: false,
  };

  try {
    const documents = await StudentDocument.find(filter);
    return response.status(200).json(documents);
  } catch (error) {
    return response
      .status(400)
      .json({ error: error, message: "Unable to fetch documents" });
  }
};

/**
 * get a single submission from the relevant groups
 */
const getStudentDocument = async (request, response) => {
  const id = request.params.id;

  try {
    const document = await StudentDocument.findById(id);
    return response.status(200).json(document);
  } catch (error) {
    return response
      .status(400)
      .json({ error: error, message: "Unable to fetch the document" });
  }
};

/**
 * get a single group
 */
```

```
controllers > supervisor | supervisor.js > ...
205  /**
206   * get a single group
207  */
208 const getGroup = async (request, response) => {
209   const groupId = request.params.id;
210
211   try {
212     const group = await Group.findOne({ groupId: groupId });
213     return response.status(200).json(group);
214   } catch (error) {
215     return response.status(400).json({ error: "Unable to find the group" });
216   }
217 }
218
219 /**
220  * evaluate document by providing marks for each members
221 */
222 const evaluateDocument = async (request, response) => {
223   const marks = request.body.marks;
224   const groupId = request.params.id;
225
226   /**
227    * If no records having the group ID available in the collection create new document
228    * If document exist having the group ID update document
229   */
230
231   try {
232     const result = await Result.findOne({ groupId: groupId });
233     if (result) {
234       result.student1.documentMarks = marks.student1;
235       result.student2.documentMarks = marks.student2;
236       result.student3.documentMarks = marks.student3;
237       result.student4.documentMarks = marks.student4;
238
239     try {
240       await result.save();
241       return response.status(200).json({ message: "Evaluation succesfull" });
242     } catch (error) {
243       return response
244         .status(400)
245         .json({ error: "Unable to submit the evaluation" });
246     }
247   } else {
248     var newResult = new Result();
249
250     try {
251       const group = await Group.findOne({ groupId: groupId });
252
253       newResult.groupId = groupId;
254       newResult.student1.studentId = group.student1;
255       newResult.student1.documentMarks = marks.student1;
256       newResult.student2.studentId = group.student2;
257       newResult.student2.documentMarks = marks.student2;
258       newResult.student3.studentId = group.student3;
259       newResult.student3.documentMarks = marks.student3;
260       newResult.student4.studentId = group.student4;
261       newResult.student4.documentMarks = marks.student4;
262
263     try {
264       await newResult.save();
265       return response
266         .status(200)
267         .json({ message: "Evaluation succesfull" });
268     } catch (error) {
269       return response
270         .status(400)
271         .json({ error: "Unable to submit the evaluation" });
272     }
273   } catch (error) {
274     return response.status(400).json({ error: "Unable to find the group" });
275   }
276 }
```

The screenshot shows the VS Code interface with the following details:

- Explorer View:** Shows the project structure. The current file, `supervisor.js`, is open in the editor.
- Editor View:** Displays the code for `supervisor.js`. The code handles evaluating final theses by providing marks for each member of a group. It uses `async/await` and `try/catch` blocks to interact with a database and return responses.

```
controllers > supervisor > supervisor.js > ...
  ...
    ...
  }
} catch (error) {
  return response.status(400).json({ error: "Unable to find a result" });
}

};

const evaluateFinalThesis = async (request, response) => {
  const marks = request.body;
  const groupId = request.params.id;

  /**
   * If no records having the group ID available in the collection create new document
   * If document exist having the group ID update document
   */

  try {
    const result = await Result.findOne({ groupId: groupId });
    if (result) {
      result.student1.finalThesisMarks = marks.student1;
      result.student2.finalThesisMarks = marks.student2;
      result.student3.finalThesisMarks = marks.student3;
      result.student4.finalThesisMarks = marks.student4;

      try {
        await result.save();
        return response.status(200).json({ message: "Evaluation succesfull" });
      } catch (error) {
        return response
          .status(400)
          .json({ error: "Unable to submit the evaluation" });
      }
    } else {
      var newResult = new Result();

      try {
        const group = await Group.findOne({ groupId: groupId });

        newResult.groupId = groupId;
        newResult.student1.studentId = group.student1;
        newResult.student1.finalThesisMarks = marks.student1;
        newResult.student2.studentId = group.student2;
        newResult.student2.finalThesisMarks = marks.student2;
        newResult.student3.studentId = group.student3;
        newResult.student3.finalThesisMarks = marks.student3;
        newResult.student4.studentId = group.student4;
        newResult.student4.finalThesisMarks = marks.student4;

        console.log(newResult);

        try {
          await newResult.save();
          return response
            .status(200)
            .json({ message: "Evaluation succesfull" });
        } catch (error) {
          return response
            .status(400)
            .json({ error: "Unable to submit the evaluation" });
        }
      } catch (error) {
        return response.status(400).json({ error: "Unable to find the group" });
      }
    }
  } catch (error) {
    return response.status(400).json({ error: "Unable to find a result" });
  }
}
```

- Bottom Status Bar:** Shows the file name as `supervisor.js`, the status as `11:01`, and other standard status bar icons.

The screenshot shows a code editor interface with a dark theme. On the left, there is a tree view of the project structure:

- OPEN EDITORS: supervisor.js
- RESEARCH-PR... (marked with a red error icon)
- controllers
- admin
- core
- panel member
- register
- student
- supervisor
  - registration.js
  - supervisor.js
- text.txt
- helpers
- models
- node modules
- routes
  - admin
  - auth
  - core
  - panel member
  - student
    - group.js
    - research.js
    - text.txt
  - supervisor
- uploads
- .env
- .gitignore
- package-lock.json
- package.json
- server.js

The main editor area displays the content of supervisor.js:

```
331     .status(200)
332         .json({ message: "Evaluation succesfull" });
333     } catch (error) {
334         return response
335             .status(400)
336                 .json({ error: "Unable to submit the evaluation" });
337     }
338 } catch (error) {
339     return response.status(400).json({ error: "Unable to find the group" });
340 }
341 }
342 } catch (error) {
343     return response.status(400).json({ error: "Unable to find a result" });
344 }
345 };
346 /**
347 * update document if the document is evaluated
348 */
349 const updateDocumentState = async (request, response) => {
350     const documentId = request.params.id;
351
352     try {
353         const document = await StudentDocument.findById(documentId);
354         if (document) {
355             document.marked = true;
356
357             try {
358                 await document.save();
359                 return response.status(200).json({ message: "Update succesfull" });
360             } catch (error) {
361                 return response.status(400).json({ error: "Update failed" });
362             }
363         }
364     } catch (error) {
365         return response.status(400).json({ error: "Unable to find document" });
366     }
367 }
368 };
369
370 module.exports = {
371     topicById,
372     getSingleTopic,
373     getAllTopics,
374     updateState,
375     getDocuments,
376     groupIDs,
377     getAssignedGroups,
378     getStudentDocuments,
379     getStudentDocument,
380     getGroup,
381     evaluateDocument,
382     evaluateFinalThesis,
383     updateDocumentstate,
384 };
385
```

The screenshot shows a code editor interface with a dark theme. The left sidebar displays a project structure:

- OPEN EDITORS
- RESEARCH-PL...
- controllers
- admin
- core
- panel member
- register
- student
- supervisor
- helpers
- models
- node\_modules
- routes
- admin
- auth
- core
- panel member
- student
- supervisor
- supervisor.js
- text.txt
- uploads
- .env
- .gitignore
- package-lock.json
- package.json
- server.js

The main editor area contains the content of the `supervisor.js` file:

```
const express = require("express");
const {
  supervisorAuthorization,
} = require("../helpers/supervisor/authorization.js");
const {
  signUp,
  signIn,
} = require("../controllers/supervisor/registration.js");
const {
  topicById,
  getSingleTopic,
  getAllTopics,
  updateState,
  getDocuments,
  groupIDs,
  getAssignedGroups,
  getStudentDocuments,
  getStudentDocument,
  getGroup,
  evaluateDocument,
  evaluateFinalThesis,
  updateDocumentState,
} = require("../controllers/supervisor/supervisor.js");
const router = express.Router();

router.post("/supervisor/signup", signUp);
router.post("/supervisor/signin", signIn);
router.post(
  "/supervisor/evaluation/document/:id",
  supervisorAuthorization,
  evaluateDocument
);
router.post(
  "/supervisor/evaluation/finalthesis/:id",
  supervisorAuthorization,
  evaluateFinalThesis
);
router.get(
  "/supervisor/topic/:id",
  supervisorAuthorization,
  topicById,
  getSingleTopic
);
router.get("/supervisor/topics/:state", supervisorAuthorization, getAllTopics);
router.get(
  "/supervisor/groups",
  supervisorAuthorization,
  groupIDs,
  getAssignedGroups
);
router.get("/supervisor/documents", supervisorAuthorization, getDocuments);
router.get(
  "/supervisor/student/documents/:type",
  supervisorAuthorization,
  groupIDs,
  getStudentDocuments
);
router.get(
  "/supervisor/student/document/:id",
  supervisorAuthorization,
  getStudentDocument
);
router.get("/supervisor/student/group/:id", supervisorAuthorization, getGroup);
router.put(
  "/supervisor/topic/:id/:state",
  supervisorAuthorization,
  topicById,
  updateState
);
router.put(
  "/supervisor/student/document/:id",
  supervisorAuthorization,
  updateDocumentState
);
module.exports = router;
```

At the bottom of the editor, there are status bars for OUTLINE, TIMELINE, and a file named `pre-final`.

## Screenshots of the MongoDB Schemas

▼ **researchdb**

- documents
- groups
- panelmembers
- researches
- results
- staffs
- studentsubmissions
- submissions
- supervisors**
- topics
- users

**FILTER** { field: 'value' }

**QUERY RESULTS: 1-2 OF 2**

```
_id: ObjectId("6297b835fa2cb02d9babca9")
firstName: "Saoirse"
lastName: "Ronan"
role: "supervisor"
email: "saoirse@gmail.com"
password: "$2b$10$LeGEN4h.3pYTrJ/NMHox.j9fJWlfnuaFt/.Wl16G5Hc855ZDu3j0"
field: "Computational biology"
degree: "computing"
contactNumber: "0712346789"
id: "SVR0001"
createdAt: 2022-06-01T19:04:21.664+00:00
updatedAt: 2022-06-01T19:04:21.664+00:00
__v: 0
```

▼ **researchdb**

- documents
- groups
- panelmembers
- researches
- results**
- staffs
- studentsubmissions
- submissions
- supervisors
- topics
- users

**FILTER** { field: 'value' }

**QUERY RESULTS: 1-3 OF 3**

```
_id: ObjectId("6299dbe018e2b3af14403838")
> student1: Object
> student2: Object
> student3: Object
> student4: Object
groupId: "RG6861"
createdAt: 2022-06-03T10:01:04.466+00:00
updatedAt: 2022-06-03T10:01:04.466+00:00
__v: 0
```

```
_id: ObjectId("6299ea6db48d0062e37f5fb4")
> student1: Object
> student2: Object
```

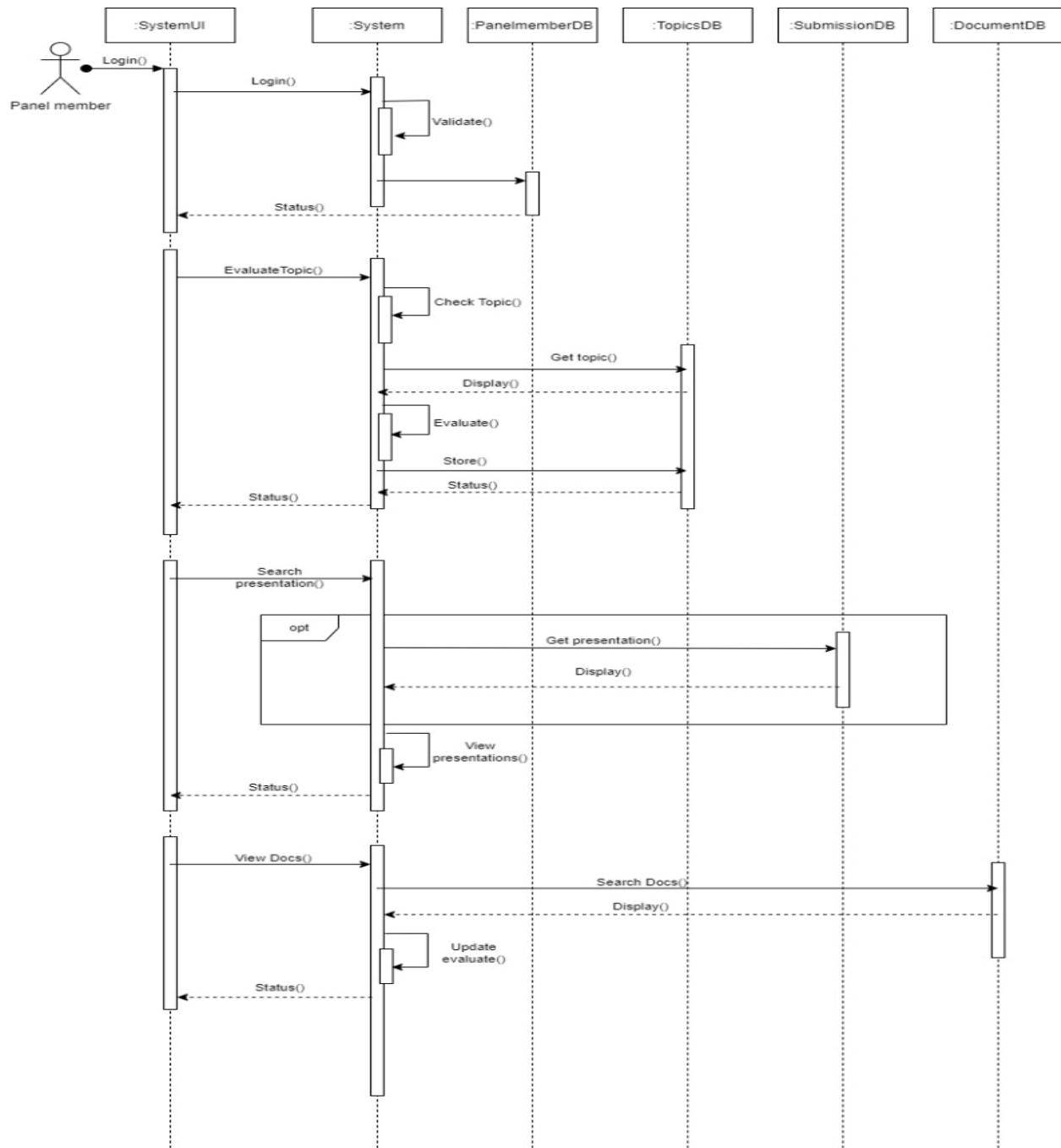
System Status: All Good

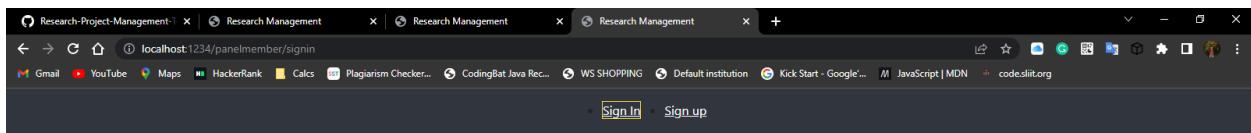
## Component Description

IT20236564 Kumarathunga M.A.M.T

Panel Member Role

Description: Panel member add by the administrator. They have several tasks in this system and their main task is to evaluated presentations. They have evaluated student's topics also and evaluate student's presentation according to the provided marking scheme.

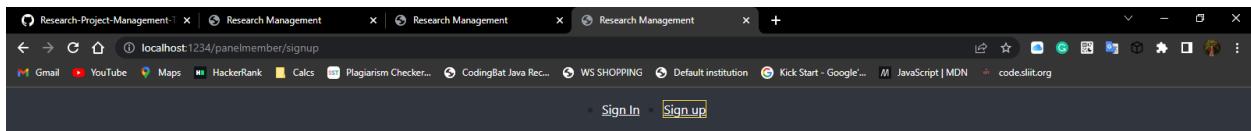




### Panel Member Sign In

Email

Password



### Panel Member Sign Up

First name <input type="text"/>	Last name <input type="text"/>
Email <input type="text"/>	
Password <input type="password"/>	Confirm password <input type="password"/>
Faculty <input type="text" value="Choose..."/>	Contact number <input type="text"/>

Screenshot of the 'Documents' section of the Research Project Management system.

The page title is 'Research Management' and the URL is 'localhost:1234/panelmember/documents'. The navigation bar includes 'Topics', 'Submissions', 'Documents', and 'Sign out'.

A search bar at the top right contains the placeholder 'Search documents'.

The main content area displays four document entries in a grid:

- Research Introduction**  
document  
Introduce about student research module  
[View Document](#)  
Uploaded on 2022-06-04 at 09:44:52
- Student Group Registration**  
document  
Details about Student Group registration process  
[View Document](#)  
Uploaded on 2022-06-04 at 09:45:52
- Student Research module delivery plan**  
document  
Delivery plan of the research module in 4 th year  
[View Document](#)  
Uploaded on 2022-06-04 at 09:46:47
- SRS document marking schema**  
marking  
SRS document evaluation sheet  
[View Document](#)  
Uploaded on 2022-06-04 at 09:51:51
- SRS document marking schema**  
marking  
SRS document evaluation sheet  
[View Document](#)
- Architecture plan marking schema**  
marking  
Architecture plan evaluation sheet  
[View Document](#)

Screenshot of the 'Topics' section of the Research Project Management system.

The page title is 'Research Management' and the URL is 'localhost:1234/panelmember/topics'. The navigation bar includes 'Topics', 'Submissions', 'Documents', and 'Sign out'.

The main content area shows two categories of topics:

- Pending** (1 item):  
IOT Miniaturization: Sensors, CPU, and network  
IOT Miniaturization: Sensors, CPU, and network  
Submitted on 2022-06-04 at 09:59:25  
Accepted on 2022-06-04 at 16:19:05  
[Evaluate](#)
- Evaluated** (0 items)

IOT Miniaturization: Sensors, CPU, and network

Group ID: RG8013 Accepted By: Saoirse Ronan

Field: IOT Supervisor: SVR0001

Description: IOT Miniaturization: Sensors, CPU, and network

Evaluation

Evaluate

RG6861 presentation

Uploaded on 2022-06-02 at 11:05:39

View

Copyright © 2022 All Rights Reserved.

The screenshot shows a web browser window with three tabs labeled "Research Management". The active tab displays a presentation slide with the following content:

**Root code**  
RG6861

[View presentation](#)

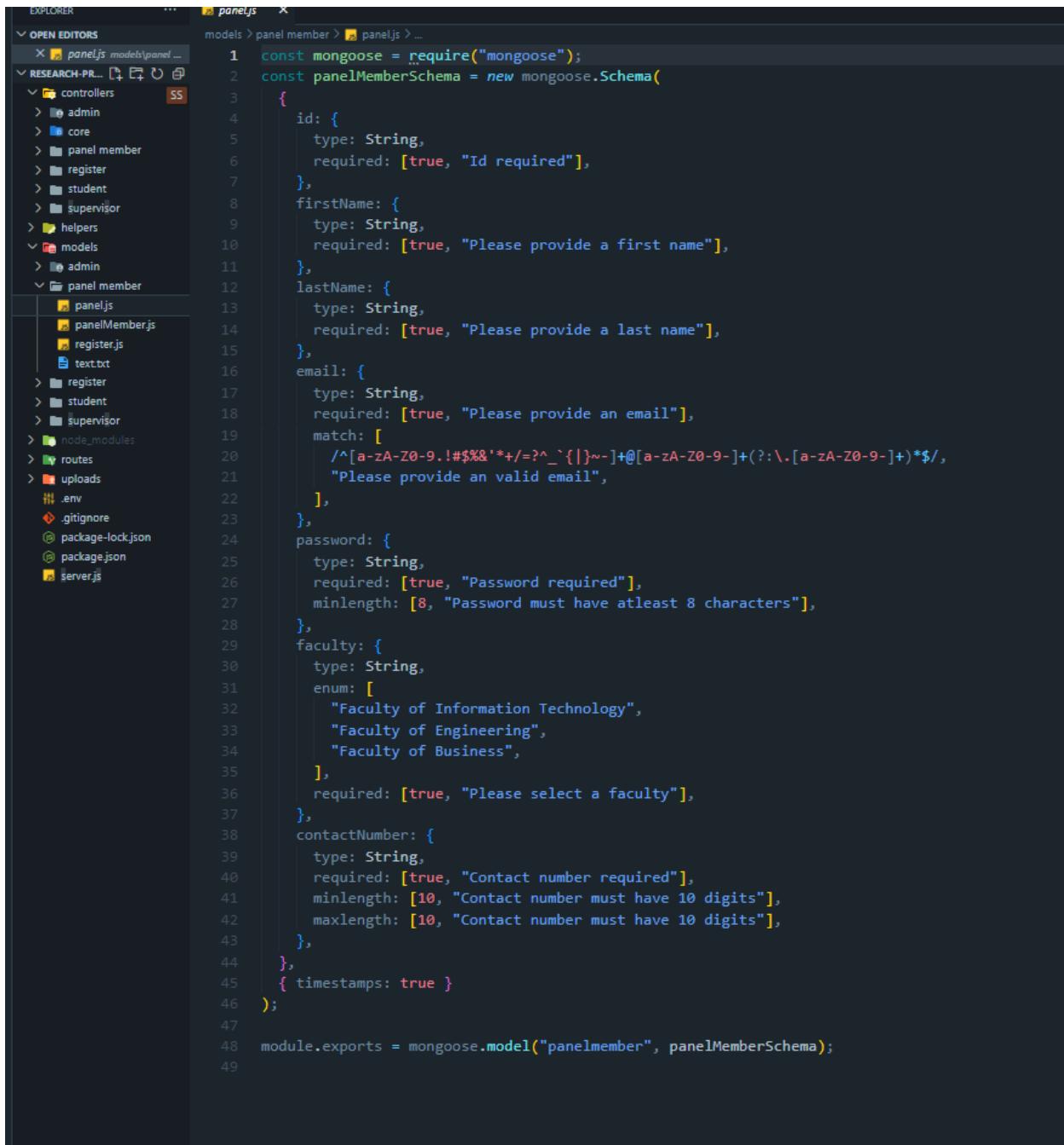
Student1 - IT202121334

Student2 - IT231225131

Student3 - IT231121531

Student4 - IT231121431

## Screenshots/ Code Snippets of the Backend:



```
const mongoose = require("mongoose");
const panelMemberSchema = new mongoose.Schema(
{
  id: {
    type: String,
    required: [true, "Id required"],
  },
  firstName: {
    type: String,
    required: [true, "Please provide a first name"],
  },
  lastName: {
    type: String,
    required: [true, "Please provide a last name"],
  },
  email: {
    type: String,
    required: [true, "Please provide an email"],
    match: [
      /^[a-zA-Z0-9.!#$%&'*+=?^_`{|~}-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*$/,
      "Please provide an valid email",
    ],
  },
  password: {
    type: String,
    required: [true, "Password required"],
    minlength: [8, "Password must have atleast 8 characters"],
  },
  faculty: {
    type: String,
    enum: [
      "Faculty of Information Technology",
      "Faculty of Engineering",
      "Faculty of Business",
    ],
    required: [true, "Please select a faculty"],
  },
  contactNumber: {
    type: String,
    required: [true, "Contact number required"],
    minlength: [10, "Contact number must have 10 digits"],
    maxlength: [10, "Contact number must have 10 digits"],
  },
},
{ timestamps: true }
);
module.exports = mongoose.model("panelmember", panelMemberSchema);
```

The screenshot shows a code editor interface with a dark theme. On the left is a file tree (left sidebar) and a search bar. The main area displays a code file.

**File Tree (Left Sidebar):**

- OPEN EDITORS
- RESEARCH-PR...
- controllers
- models
- panel member
- register
- student
- supervisor
- helpers
- admin
- panel member
- panel.js
- panelMember.js
- register.js
- text.txt
- register
- student
- supervisor
- node\_modules
- routes
- uploads
- .env
- .gitignore
- package-lock.json
- package.json
- server.js

**Code Editor Content:**

```
const mongoose = require("mongoose");

const panelMemberSchema = new mongoose.Schema(
{
  id: {
    type: String,
    required: [true, "Id required"],
  },
  firstName: {
    type: String,
    required: [true, "Please provide a first name"],
  },
  lastName: {
    type: String,
    required: [true, "Please provide a last name"],
  },
  email: {
    type: String,
    required: [true, "Please provide an email"],
    match: [
      /^[a-zA-Z0-9.!#$%&'*+=?^`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*$/,
      "Please provide a valid email",
    ],
  },
  password: {
    type: String,
    required: [true, "Password required"],
    minlength: [8, "Password must have atleast 8 characters"],
  },
  degree: {
    type: String,
    enum: ["computing", "engineering", "business"],
    required: [true, "Please select a degree"],
  },
  contactNumber: {
    type: String,
    required: [true, "Contact number required"],
    minlength: [10, "Contact number must have 10 digits"],
    maxlength: [10, "Contact number must have 10 digits"],
  },
  timestamps: true
};

module.exports = mongoose.model("panelmember", panelMemberSchema);
```

The screenshot shows a code editor interface with a dark theme. On the left is the 'EXPLORER' sidebar, which lists several files and folders under 'ONLINE EDITORS' and 'RESEARCH\_PNL'. The 'register.js' file is open in the main editor area. The code defines a MongoDB schema for a 'staff' collection. It includes validation rules for fields like 'firstName', 'lastName', 'email', 'password', 'contactNumber', 'staffId', and 'role'. It also includes a pre-save middleware function that hashes the password using bcrypt.

```
const mongoose = require("mongoose");
const bcrypt = require("bcryptjs");

const staffSchema = new mongoose.Schema({
  firstName: {
    type: "string",
    required: [true, "please Enter name"],
  },
  lastName: {
    type: "string",
    required: [true, "please Enter name"],
  },
  email: {
    type: "string",
    required: [true, "please Enter email"],
    validate: [
      // validate email
      { $regex: `^(([^<>(){}\\.,;:\\$@"]*(\\.[^<>(){}\\.,;:\\$@"]*)*)|(.+*))@((\\[[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}\\.)|(([a-zA-Z-0-9]+\\.)+[a-zA-Z]{2,}))$`},
      { $error: "please provide valid email" },
    ],
  },
  password: {
    type: "string",
    required: [true, "please Enter password"],
    minlength: [4, "must be least 4 characters"],
  },
  contactNumber: {
    type: "string",
    required: [true, "Please enter a contact number"],
    minlength: [10, "must be 10 characters"],
    maxlength: [10, "must be 10 characters"],
  },
  staffId: {
    type: "string",
    required: [true, "Please enter a staff id"],
    unique: [true, "student id already exists"],
  },
  role: {
    type: "string",
    required: [true, "Please select a role"],
    default: "panel",
  },
});

//hashing password using accessing the mongoose creatae middleware
staffSchema.pre('save', async function (next) {
  try {
    const salt = await bcrypt.genSalt(10);
    const hashedPassword = await bcrypt.hash(this.password, salt);
    this.password = hashedPassword;
    next();
  } catch (error) {
    next(error);
  }
});

//return first name
staffSchema.methods.getId = function () {
  return this.studentId;
};

module.exports = mongoose.model("staff", staffSchema);
```

```
const Topic = require("../models/student/research");
const Group = require("../models/student/group.js");
const StudentDocuments = require("../models/student/document.js");
const Result = require("../models/student/result.js");
const Document = require("../models/admin/document");

const topicById = async (request, response, next) => {
  const topicId = request.params.id;
  const filter = {
    topicId: topicId,
    state: "accepted",
  };
  try {
    const topic = await Topic.findOne(filter);
    if (topic) {
      request.topic = topic;
      next();
    } else {
      return response
        .status(400)
        .json({ message: "Topic could not be found", error: error });
    }
  } catch (error) {
    return response
      .status(400)
      .json({ message: "Topic could not be found", error: error });
  }
};

const groupIDs = async (request, response, next) => {
  const panelMemberId = request.panelMember.id;

  try {
    const groupIDs = await Group.find({
      $or: [
        { panelMember1: panelMemberId },
        { panelMember2: panelMemberId },
        { panelMember3: panelMemberId },
      ],
    }).select("groupId _id");
    var IDArray = [];

    groupIDs.forEach((id) => {
      IDArray.push(id.groupId);
    });

    request.groupIDs = IDArray;
    next();
  } catch (error) {
    return response
      .status(400)
      .json({ error: error, message: "Unable to fetch relevant group IDs" });
  }
};

const getSingleTopic = async (request, response) => {
  return response.status(200).json(request.topic);
};
```

```
48     next();
49   } catch (error) {
50     return response
51       .status(400)
52       .json({ error: error, message: "Unable to fetch relevant group IDs" });
53   }
54 };
55
56 const getSingleTopic = async (request, response) => {
57   return response.status(200).json(request.topic);
58 };
59
60 const getAllTopics = async (request, response) => {
61   const state = request.params.state;
62   const filter = {
63     state: "accepted",
64     evaluated: state,
65     groupId: request.groupIDs,
66   };
67   try {
68     const topics = await Topic.find(filter);
69     return response.status(200).json(topics);
70   } catch (error) {
71     return response
72       .status(400)
73       .json({ message: "Topics not found", error: error });
74   }
75 };
76
77 const evaluateTopic = async (request, response) => {
78   const topic = request.topic;
79
80   topic.panelMemberId = request.panelMember.id;
81   topic.panelMemberName =
82     request.panelMember.firstName + " " + request.panelMember.lastName;
83   topic.evaluated = true;
84   topic.evaluation = request.body.evaluation;
85
86   const evaluatedTopic = new Topic(topic);
87   try {
88     await evaluatedTopic.save();
89     return response.status(200).json({ message: "Update succesfull" });
90   } catch (error) {
91     return response
92       .status(400)
93       .json({ message: "Update failed", error: error });
94   }
95 };
96
97 const getDocuments = async (_request, response) => {
98   try {
99     const documents = await Document.find({ staffAllowed: true });
100    return response.status(200).json(documents);
101  } catch (error) {
102    return response
103      .status(400)
104      .json({ message: "Unable to fetch documents", error: error });
105  }
}
```

```
File Edit Selection View Go Run Terminal Help
panelMember.js - Research-Pro

EXPLORER
OPEN EDITORS
RESEARCH-PR...
controllers > panel member > panelMember.js ...
98     try {
99         const documents = await Document.find({ staffAllowed: true });
100        return response.status(200).json(documents);
101    } catch (error) {
102        return response
103            .status(400)
104            .json({ message: "Unable to fetch documents", error: error });
105    };
106}
107
108const getStudentPresentations = async (request, response) => {
109    const filter = {
110        groupId: request.groupIDs,
111        type: "presentation",
112        marked: false,
113    };
114
115    try {
116        const presentations = await StudentDocuments.find(filter);
117        return response.status(200).json(presentations);
118    } catch (error) {
119        return response
120            .status(400)
121            .json({ error: error, message: "Unable to fetch presentations" });
122    };
123}
124
125const getStudentPresentation = async (request, response) => {
126    const presentationId = request.params.id;
127
128    try {
129        const presentation = await StudentDocuments.findById(presentationId);
130        return response.status(200).json(presentation);
131    } catch (error) {
132        return response
133            .status(400)
134            .json({ error: error, message: "Unable to the fetch presentation" });
135    };
136}
137
138const getGroup = async (request, response) => {
139    const groupId = request.params.id;
140
141    try {
142        const group = await Group.findOne({ groupId: groupId });
143        return response.status(200).json(group);
144    } catch (error) {
145        return response.status(400).json({ error: "Unable to find the group" });
146    };
147}
148
149const evaluate = async (request, response) => {
150    const marks = request.body;
151    const groupId = request.params.id;
152
153    console.log(marks);
154
155    try {
```

```
panelMember.js - Research-P

File Edit Selection View Go Run Terminal Help
EXPLORER ... panelMember.js X
RESEARCH-PR... controllers > panel member > panelMember.js ...
OPEN EDITORS
RESEARCH-PR... controllers admin core panel member
panelMember.js registration.js text.txt
register student supervisor helpers models routes uploads .env .gitignore package-lock.json package.json server.js
153 console.log(marks);
154
155 try {
156   const result = await Result.findOne({ groupId: groupId });
157   if (result) {
158     result.student1.presentationMarks = marks.student1;
159     result.student2.presentationMarks = marks.student2;
160     result.student3.presentationMarks = marks.student3;
161     result.student4.presentationMarks = marks.student4;
162
163     try {
164       await result.save();
165       return response.status(200).json({ message: "Evaluation succesfull" });
166     } catch (error) {
167       return response
168         .status(400)
169         .json({ error: "Unable to submit the evaluation" });
170     }
171   } else {
172     var newResult = new Result();
173
174     try {
175       const group = await Group.findOne({ groupId: groupId });
176
177       newResult.groupId = groupId;
178       newResult.student1.studentId = group.student1;
179       newResult.student1.presentationMarks = marks.student1;
180       newResult.student2.studentId = group.student2;
181       newResult.student2.presentationMarks = marks.student2;
182       newResult.student3.studentId = group.student3;
183       newResult.student3.presentationMarks = marks.student3;
184       newResult.student4.studentId = group.student4;
185       newResult.student4.presentationMarks = marks.student4;
186
187       console.log(newResult);
188
189     try {
190       await newResult.save();
191       return response
192         .status(200)
193         .json({ message: "Evaluation succesfull" });
194     } catch (error) {
195       return response
196         .status(400)
197         .json({ error: "Unable to submit the evaluation" });
198     }
199     } catch (error) {
200       return response.status(400).json({ error: "Unable to find the group" });
201     }
202   }
203 } catch (error) {
204   return response.status(400).json({ error: "Unable to find a result" });
205 }
206 };
207
208 /**
209  * update state of the presentation to marked

```

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help
- Explorer:** Shows the project structure:
  - RESEARCH-PR...
  - controllers
    - panelMember.js
    - admin
    - core
    - panel member
      - panelMember.js
      - registration.js
      - text.txt
    - register
    - student
    - supervisor
  - helpers
  - models
  - node\_modules
  - routes
  - uploads
  - .env
  - .gitignore
  - package-lock.json
  - package.json
  - server.js
- Editor:** The active file is "panelMember.js" located at "controllers > panel member". The code is as follows:

```
198     }
199     } catch (error) {
200       return response.status(400).json({ error: "Unable to find the group" });
201     }
202   }
203   } catch (error) {
204     return response.status(400).json({ error: "Unable to find a result" });
205   }
206 };
207 /**
208 * update state of the presentation to marked
209 */
210 const updatePresentationState = async (request, response) => {
211   const presentationId = request.params.id;
212
213   try {
214     const presentation = await StudentDocuments.findById(presentationId);
215     if (presentation) {
216       presentation.marked = true;
217
218       try {
219         await presentation.save();
220         return response.status(200).json({ message: "Update succesfull" });
221       } catch (error) {
222         console.log(error);
223         return response.status(400).json({ error: "Update failed" });
224       }
225     }
226   } catch (error) {
227     confirm.log(error);
228     return response.status(400).json({ error: "Unable to find presentation" });
229   }
230 };
231 };
232
233 module.exports = {
234   topicById,
235   getSingleTopic,
236   getAllTopics,
237   evaluateTopic,
238   groupIDs,
239   getDocuments,
240   getStudentPresentations,
241   getStudentPresentation,
242   getGroup,
243   evaluate,
244   updatePresentationState,
245 };
246
```

```
const express = require("express");
const {
  panelMemberAuthorization,
  topicById,
  getAllTopics,
  evaluateTopic,
  groupIDs,
  getDocuments,
  getStudentPresentations,
  getStudentPresentation,
  getGroup,
  evaluate,
  updatePresentationState,
} = require("../helpers/supervisor/authorization.js");
const router = express.Router();

router.post("/panelmember/signup", signUp);
router.post("/panelmember/signin", signIn);
router.post("/panelmember/evaluation/:id", panelMemberAuthorization, evaluate);

router.get(
  "/panelmember/topic/:id",
  panelMemberAuthorization,
  topicById,
  getSingleTopic
);
router.get(
  "/panelmember/topics/:state"
);
```

```
File Edit Selection View Go Run Terminal Help
panelMember.js - Research-Project-Management-Tool-backend - Visual Studio Code
EXPLORER      ...
OPEN EDITORS   panelMember.js X
RESEARCH-PR... routes > panel member > panelMember.js > ...
controllers    SS
  admin
  core
  panel member
  register
  student
  supervisor
  helpers
  models
  node_modules
routes
  admin
  auth
  core
  panel member
    panelMember.js
    text.txt
    student
    supervisor
    uploads
    .env
    .gitignore
    package-lock.json
    package.json
    server.js
routes
  panelMember.get(
    "/panelmember/topics/:state",
    panelMemberAuthorization,
    groupIDs,
    getAllTopics
  );
  router.get("/panelmember/documents", panelMemberAuthorization, getDocuments);
  router.get(
    "/panelmember/student/presentations",
    panelMemberAuthorization,
    groupIDs,
    getStudentPresentations
  );
  router.get(
    "/panelmember/student/presentation/:id",
    panelMemberAuthorization,
    getStudentPresentation
  );
  router.get(
    "/panelmember/student/group/:id",
    panelMemberAuthorization,
    getGroup
  );
  router.put(
    "/panelmember/topic/:id",
    panelMemberAuthorization,
    topicById,
    evaluateTopic
  );
  router.put(
    "/panelmember/student/presentaton/:id",
    panelMemberAuthorization,
    updatePresentationState
  );
module.exports = router;
```

## Screenshots of the MongoDB Schemas

The screenshot shows the MongoDB Compass interface. On the left, the database structure is outlined under 'researchdb':

- documents
- groups
- panelmembers** (highlighted in green)
- researches
- results
- staffs
- studentsubmissions
- submissions
- supervisors
- topics
- users

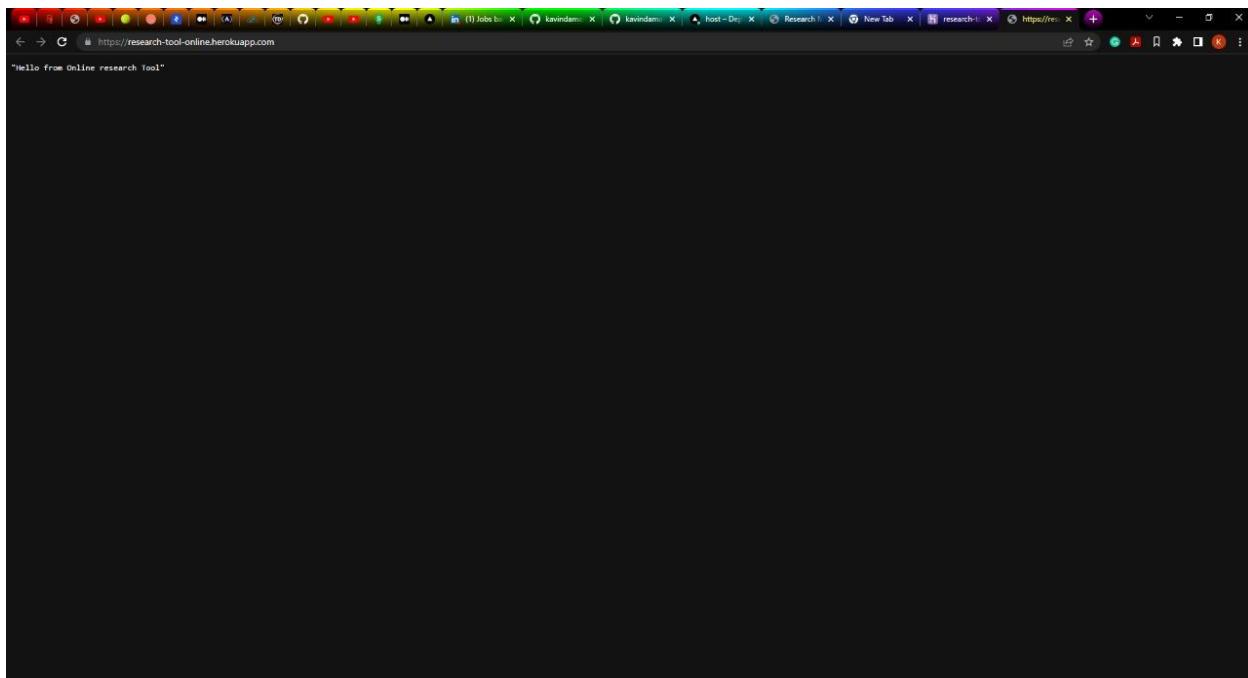
On the right, the 'panelmembers' collection is selected. A 'FILTER' button with the placeholder '{ field: 'value' }' is at the top. Below it, the results are displayed:

**QUERY RESULTS: 1-3 OF 3**

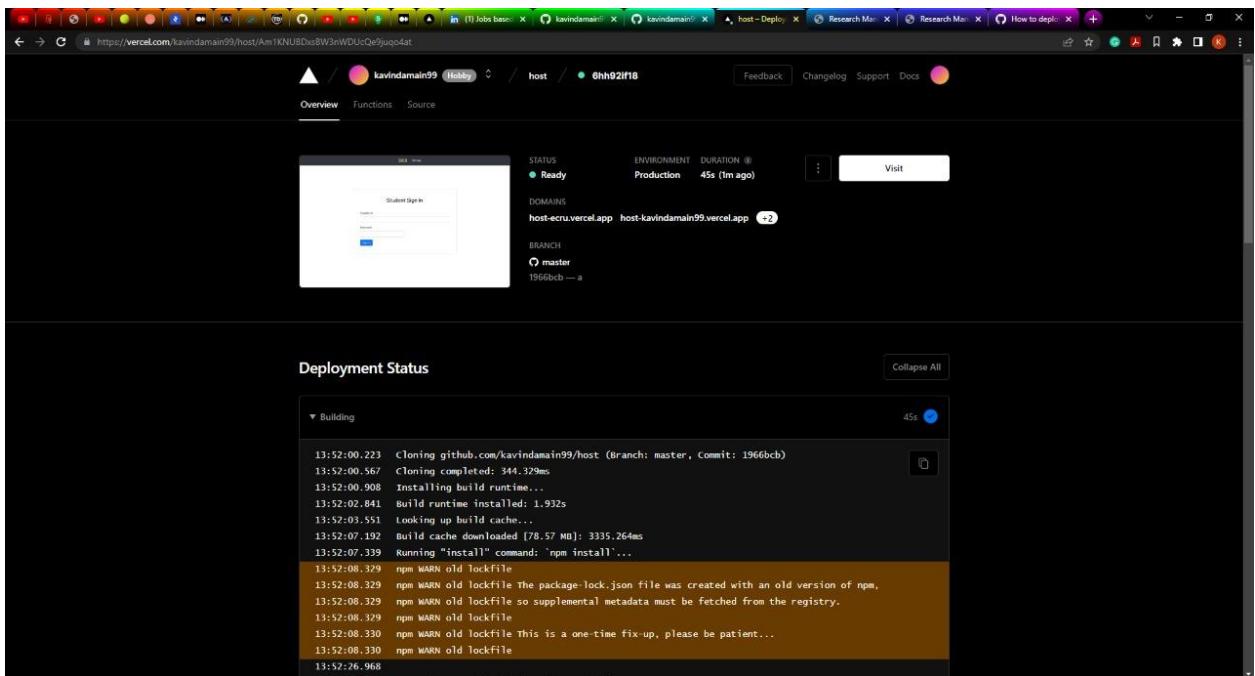
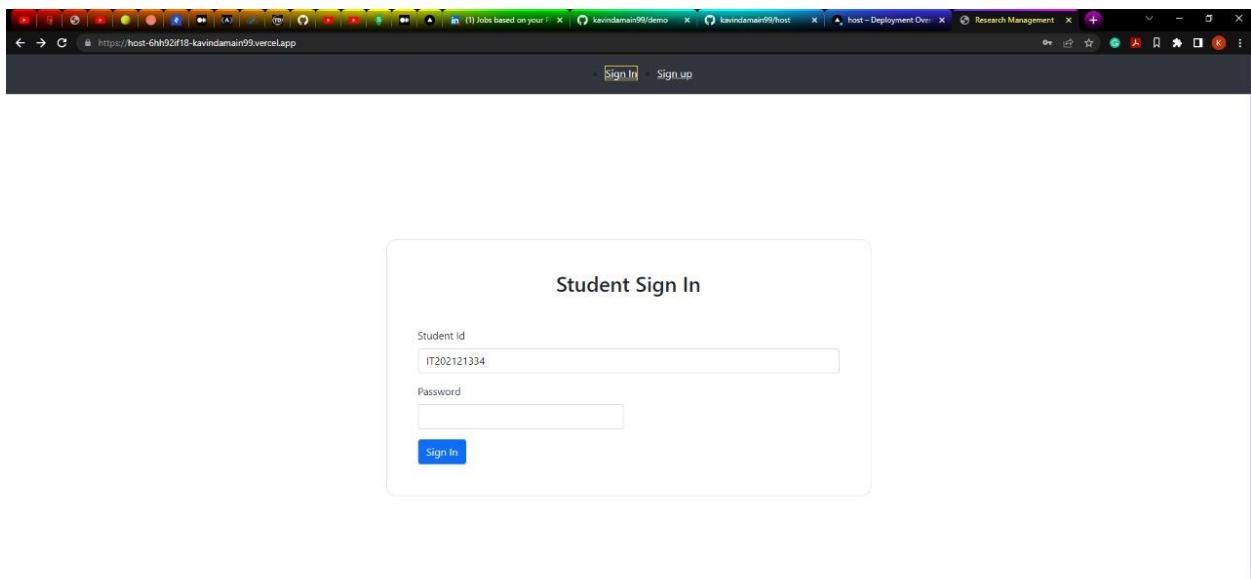
>	<pre>_id: ObjectId("6297b93a1743629d704ebcaf") firstName: "Saoirse" lastName: "Ronan" email: "saoirse@gmail.com" password: "\$2b\$10\$CHvxmi8Ij6wFYkmvoLRKcelrBwU/whBUZTwedY0kERod90ZdHie80" degree: "computing" contactNumber: "0712346789" id: "PMR0001" createdAt: 2022-06-01T19:08:42.600+00:00 updatedAt: 2022-06-01T19:08:42.600+00:00 __v: 0</pre>
	<pre>_id: ObjectId("6297b93a1743629d704ebcb2")</pre>

System Status: All Good

## Proof of Hosting (Backend and Frontend)

A screenshot of the Heroku dashboard for the application "research-tool-online". The dashboard shows the following details:

- Personal > research-tool-online**: GitHub user kavindanim99/Research-Project-Management-Tool-backend
- Installed add-ons: \$0.00/month**: No add-ons installed.
- Dyno formation: \$0.00/month**: Using free dynos. Dyno configuration: web: npm start, ON.
- Collaborator activity:** kavindanim@gmail.com (4 deploys)
- Latest activity:** A list of deployment logs:
  - kavindanim@gmail.com: Deployed 0ffed450 Today at 11:35 AM v6 [Compare diff](#)
  - kavindanim@gmail.com: Build succeeded Today at 11:34 AM [View build log](#)
  - kavindanim@gmail.com: Deployed 0ffed450 Today at 12:04 AM v5 [Compare diff](#)
  - kavindanim@gmail.com: Build succeeded Today at 12:04 AM [View build log](#)
  - kavindanim@gmail.com: Deployed 4e42786e Today at 12:00 AM v4 [Compare diff](#)
  - kavindanim@gmail.com: Build succeeded Today at 12:00 AM [View build log](#)
  - kavindanim@gmail.com: Deployed 4e42786e Yesterday at 11:59 PM v3
  - kavindanim@gmail.com: Build succeeded Yesterday at 11:59 PM [View build log](#)
  - kavindanim@gmail.com: Enable Logplex Yesterday at 11:58 PM - v2



## Test Cases

<b>Member ID :</b> IT20226046	
<b>Member Name:</b> Kavinda M.A.I.N	
<b>Testing Function 1:</b> Insert a submission	
<b>Test case ID :</b> 01	<b>Test designed by:</b> IT20226046
<b>Test priority (High/Medium/Low):</b> Medium	Test executed by: Kumarathunga M.A.M.T
<b>Test Description:</b> Admin can decide the submission type of research paper.	

Test steps	Test data	Expected result	Actual result	Status
Login as admin	Email, password	Display admin page	Display admin page	Pass
Go to the dashboard	N/A	Display dashboard	Display dashboard	Pass
Select the submission	N/A	Display submission insert option	Display submission insert option	Pass
Insert a submission	N/A	Ask submission type	Ask submission type	Pass
Add submission type	Pdf, doc	Successfully added	Successfully added	Pass

<b>Member ID :</b> IT20134280	
<b>Member Name:</b> S. Ishalini	
<b>Testing Function 1:</b> Insert a submission	
<b>Test case ID :</b> 01	<b>Test designed by:</b> IT20134280
<b>Test priority (High/Medium/Low):</b> Medium	Test executed by: Kumarathunga M.A.M.T
<b>Test Description:</b> Student can register their student group.	

Test steps	Test data	Expected result	Actual result	Status
Register to the System	First name, last name, email, password, phone number	Display the login page	Display the login page	Pass
Login	Student ID, password	Display student page	Display student page	Pass
Select Topic Registration	Group Id, topic, description, research field	Successfully Submitted and display the submitted details	Successfully Submitted and display the submitted details	Pass

<b>Member ID :</b> IT20227036	
<b>Member Name:</b> Moremad M.M.M.B	
<b>Testing Function 1:</b>	
<b>Test case ID :</b> 01	<b>Test designed by:</b> IT20227036
<b>Test priority (High/Medium/Low):</b> Medium	Test executed by: Kumarathunga M.A.M.T
<b>Test Description:</b> Supervisor can be accept the student's research topic	

Test steps	Test data	Expected result	Actual result	Status
Register to the System	First name, last name, email, password, phone number	Display the login page	Display the login page	Pass
Login	Student ID, password	Display student page	Display student page	Pass
Select Topic	N / A	Display pending topics, accepted topics and rejected topics	Display pending topics, accepted topics and rejected topics	Pass
Select pending Topics	N / A	Display about the topic	Display about the topic	Pass
Select Accept Button	N / A	Successful message and display at the topic pages as a accepted topic	Successful message and display at the topic pages as a accepted topic	Pass

<b>Member ID :</b> IT20236564	
<b>Member Name:</b> Kumarathunga M.A.M.T	
<b>Testing Function 1:</b> Insert a submission	
<b>Test case ID :</b> 01	<b>Test designed by:</b> IT20235453
<b>Test priority (High/Medium/Low):</b> Medium	Test executed by: Kavinda M.A,I.N
<b>Test Description:</b> Panel member can see the students' final submissions and presentations.	

Test steps	Test data	Expected result	Actual result	Status
Login	Student ID, password	Display student page	Display student page	Pass
Go to Documents page	N/A	Display all the documents which were submitted by the student groups.	Display all the documents which were submitted by the student groups.	Pass
Select View Documents	N / A	Display Document's details	Display Document's details	Pass

