Sri Lanka Institute of Information Technology

B.Sc. Honors Degree in Information Technology

Specialized in Software Engineering

Final Examination
Year 3, Semester 1 (2021)

SE3040 – Application Frameworks

Group Project

Technical Report

Group ID: 2021S1_REG_WD_17

Group Name: Light Waves

Repository Link: https://github.com/TharushiNDewangi/AF-CFA.git

Student ID	Student Name
IT19123196	K.H.T.N Dewangi
IT19159072	A.N.S. Thenuwara
IT19129518	Rathnayake R.M.D.M
IT19140308	D.K.M Disanayake

For Evaluations:

Marks Allocation	Leader	Member 2	Member 3	Member 4	
User Interfaces look professional and consistency – 15					
Features are comprehensive and user friendly – 15					
Implementation of the RESTful web services – 15					
Implementing database and Mongo collections – 10					
Implementation of unit tests – 10					
Coding Standards and quality – 5					
Deployed in the cloud – 5					
Maintaining online git repository – 5					
Technical Report (Group Mark) – 5					
User guide (Group Mark) – 5					
Presentation – 10					

Comments:

Contents

Brief description of all the functions (member wise)

IT19123196

User Handling

There are 4 main user roles as admin, editor, reviewer and user but only user can register using registration form and user can register as a researcher , workshop presentations and attendee .in registration form user can select their user roles . users can access separate tasks related their user roles.

Only users can access and submit research papers, conduct workshops and present research related activities it is only visible reviewers and reviewer can approve these details then it shows all registered users

Editors can add conference details and edit. these conference details only visible to admin in admin dashboard and if admin approve conference details it visible all registered or non registered users These things control using middleware functions .

Approve workshops and research related activities

When User submit their content it is not visible any user in application it only visible reviewer and they can approve or reject it. if reviewer approve workshops and research papers they are visible to all registered users and send mail to content posted user . if it is rejected, it is not visible any where in the application and send mail to user says ‘ update and re submit your content ‘ .

IT19159072

Add Research related activities

Only researcher who have access to submit research papers and research related activities. When researcher registered to the system the research paper uploaded alongside the contact information. . Research paper presenters must pay if their papers approved to present them at the conference . If researcher did the payment he will get email that successfully received.

Add Workshops

Only workshop conductor who have access to submit workshop proposals and When workshop conductor registered to the system a proposal containing all the details about the workshop uploaded alongside the contact information. Workshop conductors don’t have to pay.

View Research related activities

All the submitted research papers and research related activities can view by reviewer. Research papers should approve or rejected by reviewer. Whenever reviewer approve research papers then research papers can view to the user. Only the approve papers can be view and also researcher get a email .Reviewers can view paper uploads in separate page. Researcher can download the research papers.

View Workshops

All the submitted workshop proposal can view by reviewer. Workshop proposals should approve or rejected by reviewer. Whenever reviewer approve Workshop proposals then Workshop proposals can view to the user approve or reject the Workshop proposals also researcher get a email .Reviewers can view workshop proposal uploads in separate page. Workshop conductor can download the workshop proposal.

IT19129518

Add Conference

Editor should log in using user credential which were given by Admin. An editor's roll is managing conferences. Only Editor has permission to add conferences to system and edit conferences .

View Conferences

After approval from admin, conferences view in conference list page to only editor and admin. After view conferences by attendees must pay if they want to register for the conference. If attendee did the payment, he will get email that successfully received.

IT19140308

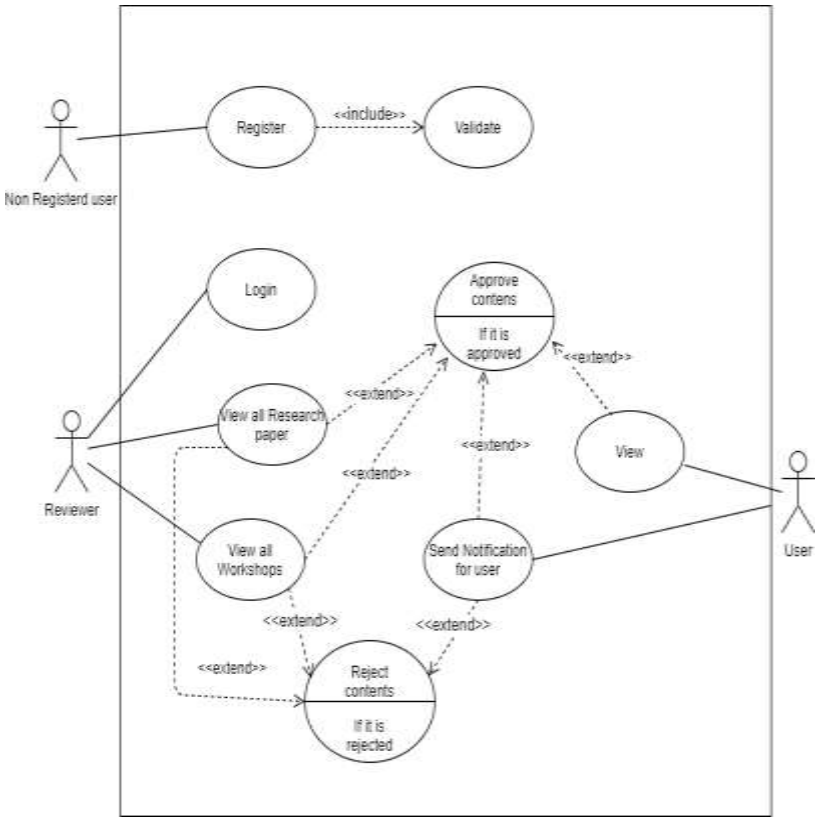
Approve Conferences

When editor submit their conferences it is not visible any user in application it only visible editor and they can approve or reject it. if editor approve conferences it is visible to all users include with nun registered users and send mail to content posted user. if it is rejected, it is not visible anywhere in the application and send mail to an editor says 'update and re submit your content'.

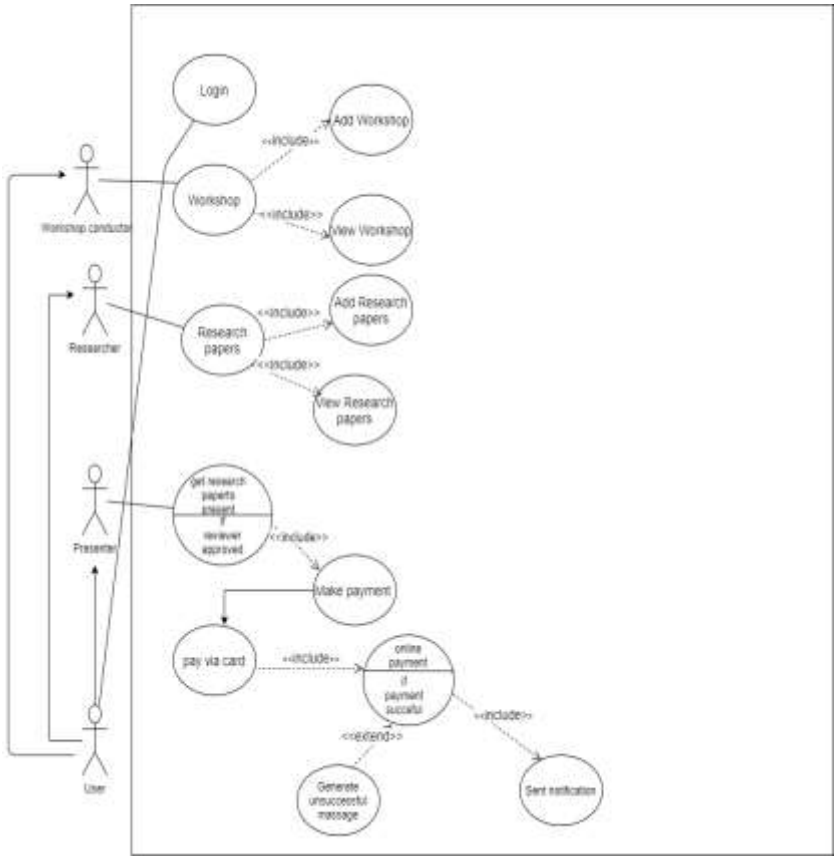
Use case Diagrams and the Component tree (member wise)

Use case Diagrams

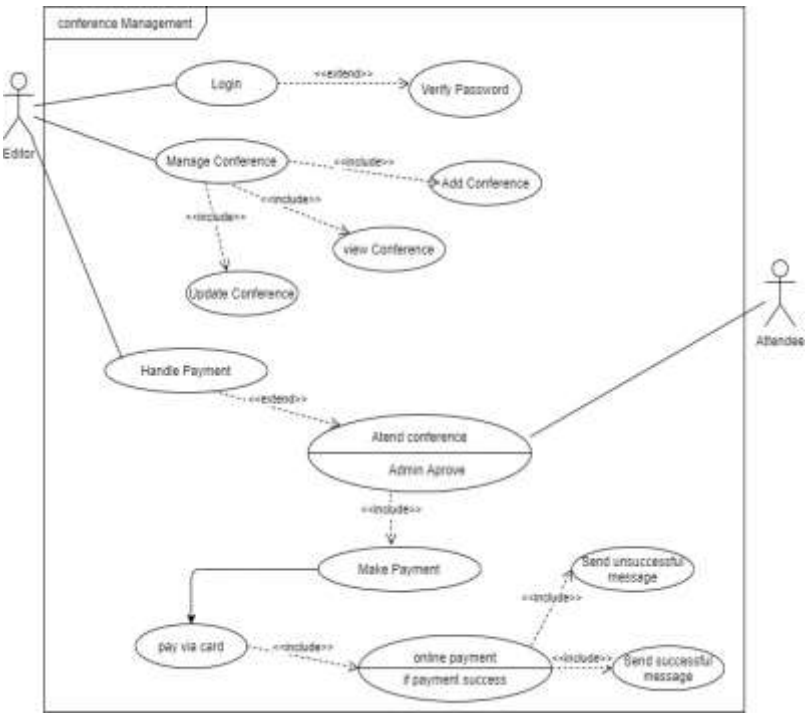
IT19123196


Non Registerd user
Register
<<include>>
Validate
Login
Approve contens
If it is approved
<<extend>>
View all Research paper
<<extend>>
<<extend>>
<<extend>>
View
Reviewer
User
View all Workshops
Send Notfication for user
<<extend>>
<<extend>>
<<extend>>
Reject contents
If it is rejected

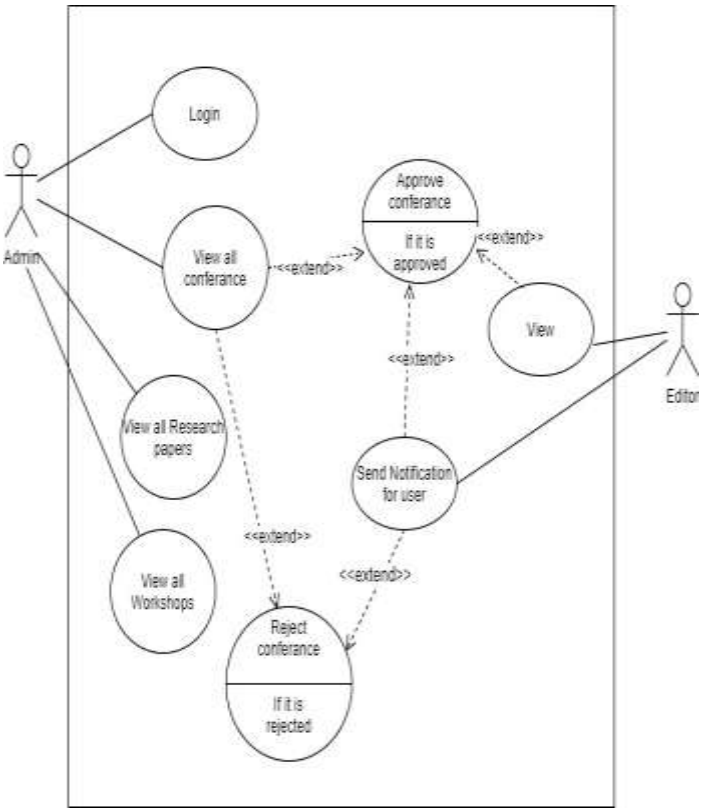
IT19159072



IT19129518



IT19140308



Component Tree

IT19123196

1.HomePage

1.Navbar

2.ButtonCollection

1.Signin

1.Home(Registered user view)

1.For Admin

3.For Admin

1.Button Collection

1.Approve Research

2.Approve workshop

IT19159072

1.HomePage

1.Navbar

2.ButtonCollection

1.Signin

1.Home(Registered user view)

1.For Authors

2.Workshops

3.For Authors

1.Button Collection

1.Add Research papers

2.ReserchPapers

1.Pay

2.Download

4.Workshops

1.Button Collection

1.Add Workshops

2.Workshops

1.Download

IT19129518

1.HomePage

1.Navbar

2.ButtonCollection

1.Signin

1.Home(Registered user view)

1. Conferences

3.Conferences

1.Button Collection

1.Add Conferences

2.View Conferences

1.pay

3.Edit Conference

IT19140308

1.HomePage

1.Navbar

2.ButtonCollection

1.Signin

1.Home(Registered user view)

1.For Admin

3.For Admin

1.Button Collection

1.Approve Conference

Usage of Rest APIs (member wise)

Rest API is developed using React.it runs on port 8065.Rest API has admin, editor, reviewer ,user , Research papers, workshops, conference details , approve details, payment CRUD operations and send email notifications as well .Rest API has access to mongoose database and web client could access mongoose through this Rest API. This is research paper route. we use that way to implements all REST API. we give access permission like that way in route

IT19123196

End point	Description
http://localhost:8065/api/signup	User registration
http://localhost:8065/api/signin	User Login
http://localhost:8065/api/signout	User Sign out
http://localhost:8065/api/approvep/	View all research papers in Reviewer Dashboard
http://localhost:8065/api/paper/approveper/:_id	Approve research paper
http://localhost:8065/api/paper/rejectpaper/:_id	Reject Research paper
http://localhost:8065/api/approveworkshops	View all Workshop in Reviewer Dashboard
http://localhost:8065/api/approveworkshops/approve/:_id	Approve Workshop
http://localhost:8065/api/approveworkshops/reject/:_id	Reject Workshop

IT19159072

endpoint	description
http://localhost:8065/api/paper/create	Submit papers
http://localhost:8065/api/allpper	Get all papers
http://localhost:8065/api/papers/:paperid	Get paper by id
http://localhost:8065/api/workshop/create	Submit workshop
http://localhost:8065/api/workshops	Get all workshops
http://localhost:8065/api/workshops/:workshopid	Get workshops by id
http://localhost:8065/api/payment/create	Make payment

IT19129518

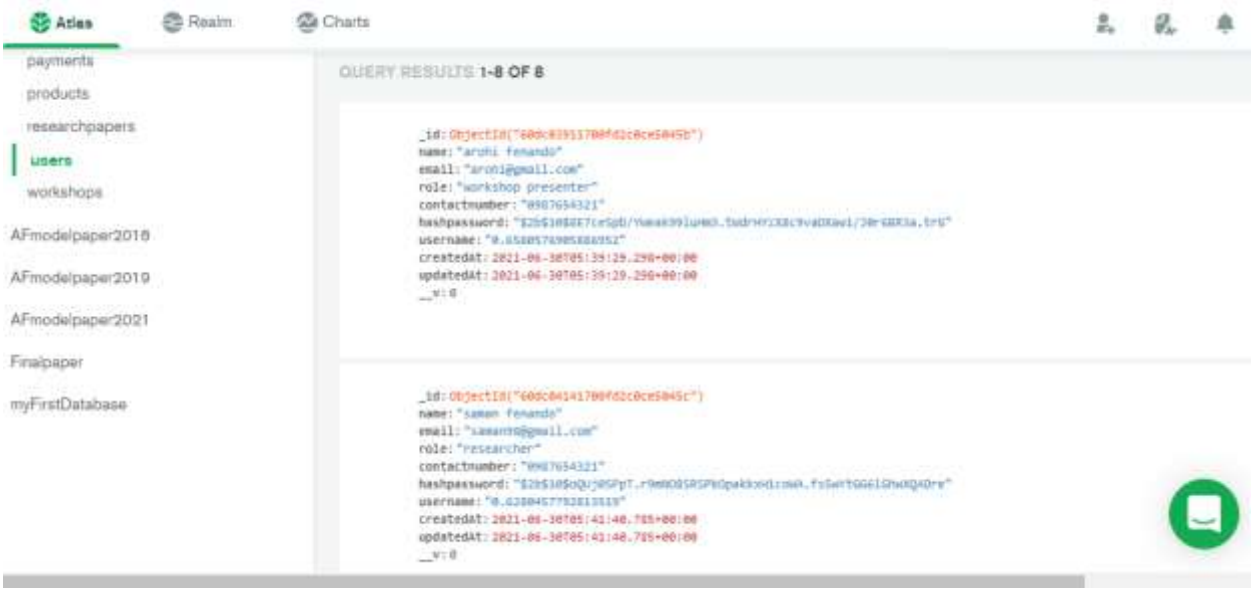
End point	Description
http://localhost:8065/api/conference/addconference	Editor add Conference
http://localhost:8065/api/conference/approve/getconference	View all approve conference
http://localhost:8065/api/conference/getconference	View All conference
http://localhost:8065/api/updateconference/:_id	Update conference
http://localhost:8065/api/payment/conference/create	Make payment

IT19140308

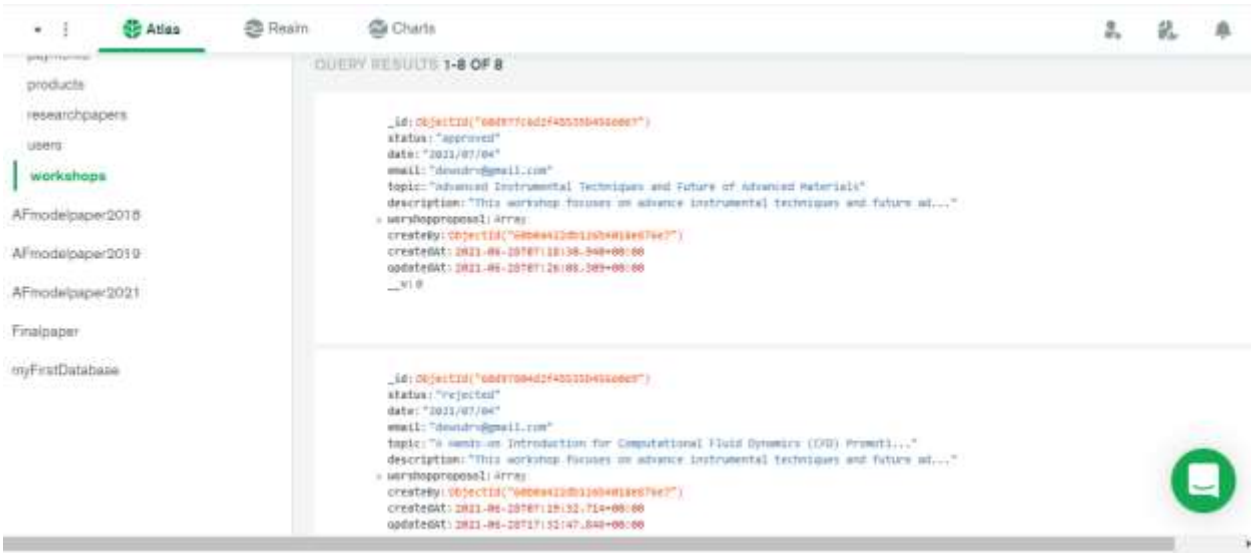
endpoint	description
http://localhost:8065/api/conference/getapproveconference	View all conference
http://localhost:8065/api/conference/approveconference/_:id	Admin approve conference
http://localhost:8065/api/conference/rejectconference/_:id	Admin reject conference

Screen shots of Mongo DB Collections (member wise)

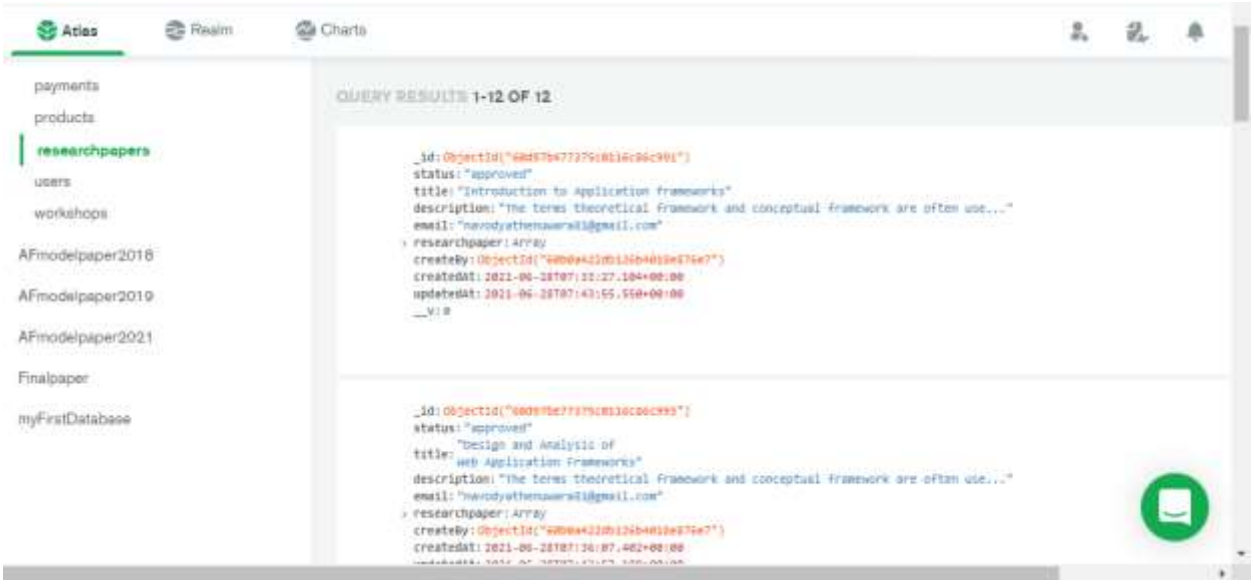
IT19123196

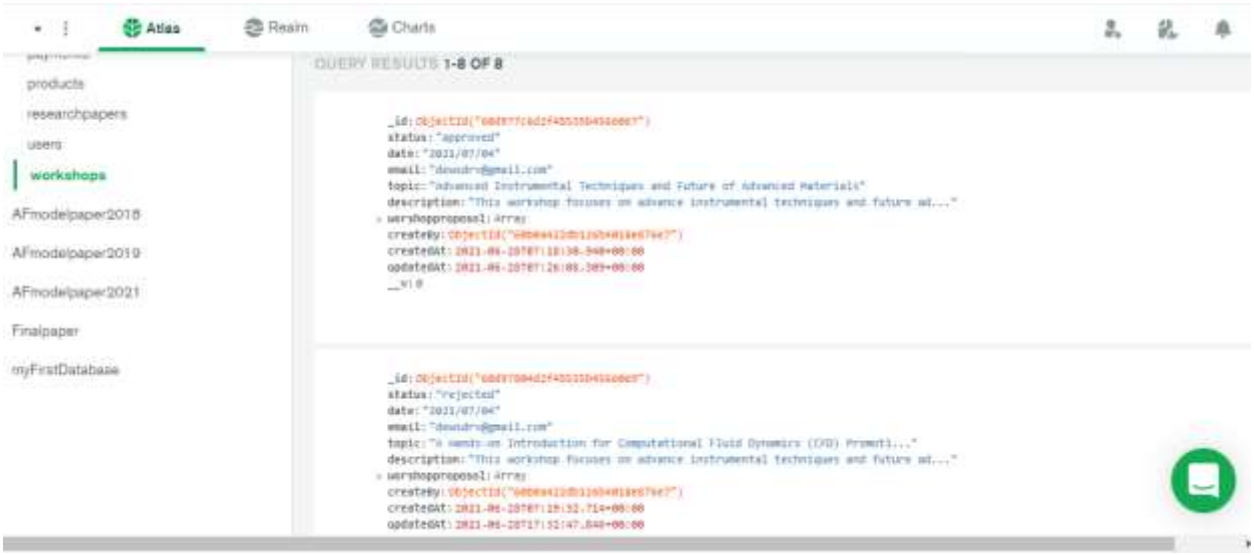

Atlas
Realm
Charts
payments
products
researchpapers
users
workshops
AFmodelpaper2018
AFmodelpaper2019
AFmodelpaper2021
Finalpaper
myFirstDatabase
QUERY RESULTS 1-8 OF 8
_id: ObjectId("60dc6351170bfd2c0ce5045b")
name: "arohi fenando"
email: "arohi@gmail.com"
role: "workshop presenter"
contactnumber: "0987654321"
username: "0.6580576985886852"
createdAt: 2021-06-30T05:39:29.296+00:00
updatedAt: 2021-06-30T05:39:29.296+00:00
__v: 0
_id: ObjectId("60dc64141700fd2c0ce5045c")
name: "saman fenando"
email: "samanfn@gmail.com"
role: "researcher"
contactnumber: "0987654321"
username: "0.6238457792815519"
createdAt: 2021-06-30T05:41:40.785+00:00
updatedAt: 2021-06-30T05:41:40.785+00:00
__v: 0

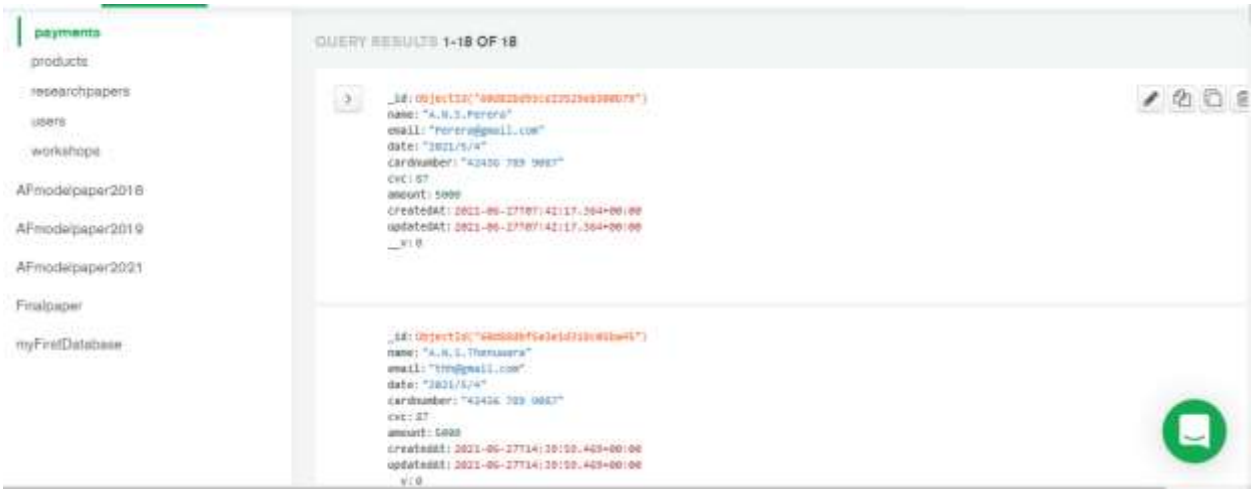

Atlas
Realm
Charts
payments
products
researchpapers
users
workshops
AFmodelpaper2018
AFmodelpaper2019
AFmodelpaper2021
Finalpaper
myFirstDatabase
QUERY RESULTS 1-12 OF 12
_id: ObjectId("60d87be77275c8116c86c901")
status: "approved"
title: "Introduction to Application Frameworks"
description: "The terms theoretical framework and conceptual framework are often use..."
email: "navodyathenuwara1@gmail.com"
researchpaper: Array
createBy: ObjectId("60b0e421db126b4010e676e7")
createdAt: 2021-06-28T07:33:27.104+00:00
updatedAt: 2021-06-28T07:43:55.550+00:00
__v: 0
_id: ObjectId("60d97be77275c8116c86c993")
status: "approved"
title: "Design and Analysis of
Web Application Frameworks"
description: "The terms theoretical framework and conceptual framework are often use..."
email: "navodyathenuwara1@gmail.com"
researchpaper: Array
createBy: ObjectId("60b0e421db126b4010e676e7")
createdAt: 2021-06-28T07:36:07.462+00:00


Atlas
Realm
Charts
products
researchpapers
users
workshops
AFmodelpaper2018
AFmodelpaper2019
AFmodelpaper2021
Finalpaper
myFirstDatabase
QUERY RESULTS 1-8 OF 8
_id: ObjectId("60d977c6d2f455350456b0d7")
status: "approved"
date: "2021/07/04"
email: "devadrn@gmail.com"
topic: "Advanced Instrumental Techniques and Future of Advanced Materials"
description: "This workshop focuses on advance instrumental techniques and future ad..."
workshopproposal: Array
createBy: ObjectId("60b0e421db126b4010e676e7")
createdAt: 2021-06-28T07:18:30.940+00:00
updatedAt: 2021-06-28T07:26:08.389+00:00
__v: 0
_id: ObjectId("60d97804d2f455350456b0d8")
status: "rejected"
date: "2021/07/04"
email: "devadrn@gmail.com"
topic: "A hands-on Introduction for Computational Fluid Dynamics (CFD) Prometi..."
description: "This workshop focuses on advance instrumental techniques and future ad..."
workshopproposal: Array
createBy: ObjectId("60b0e421db126b4010e676e7")
createdAt: 2021-06-28T07:19:32.714+00:00
updatedAt: 2021-06-28T17:33:47.840+00:00

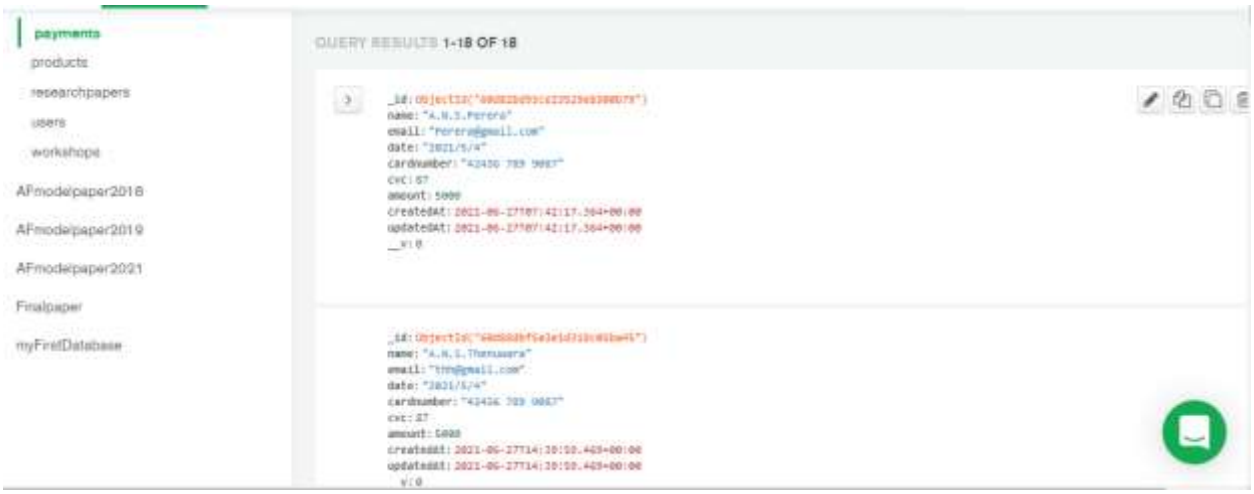
IT19159072

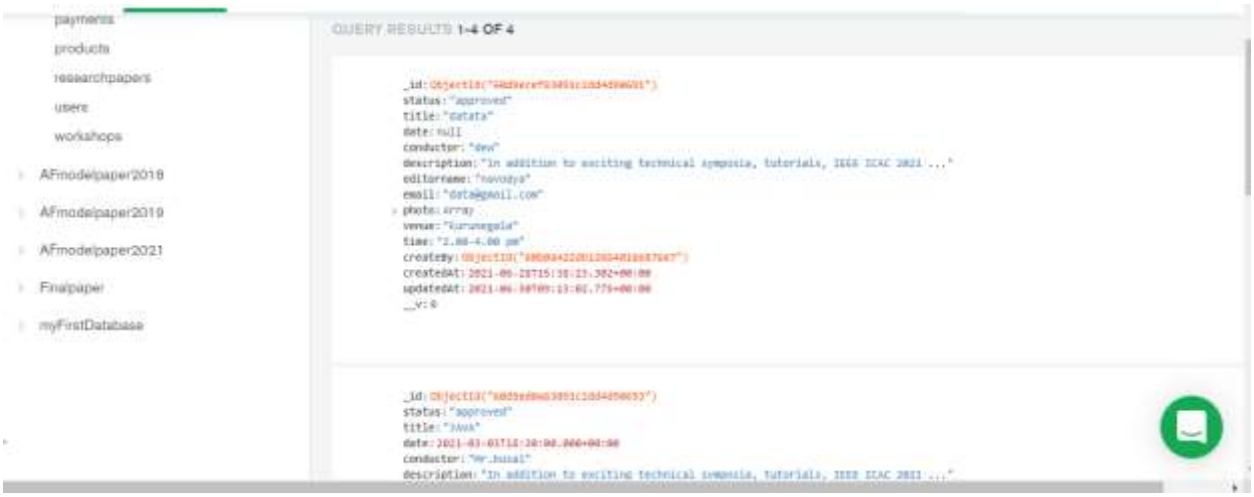

Atlas
Realm
Charts
payments
products
researchpapers
users
workshops
AFmodelpaper2018
AFmodelpaper2019
AFmodelpaper2021
Finalpaper
myFirstDatabase
QUERY RESULTS 1-12 OF 12


Atlas
Realm
Charts
products
researchpapers
users
workshops
AFmodelpaper2018
AFmodelpaper2019
AFmodelpaper2021
Finalpaper
myFirstDatabase
QUERY RESULTS 1-8 OF 8

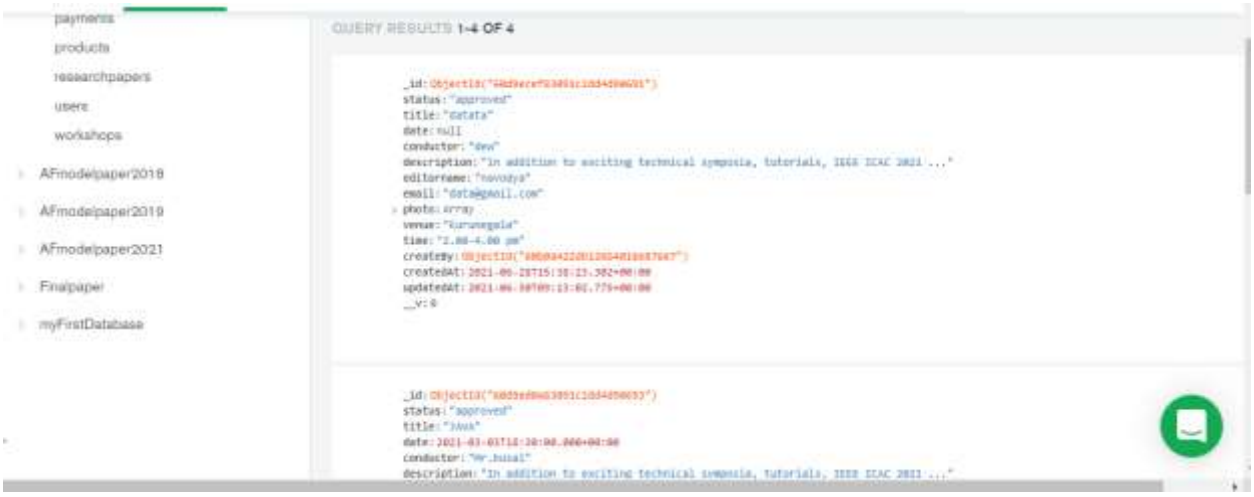

payments
products
researchpapers
users
workshops
AFmodelpaper2018
AFmodelpaper2019
AFmodelpaper2021
Finalpaper
myFirstDatabase
QUERY RESULTS 1-18 OF 18

IT19129518





IT19140308



Test Cases (member wise)

IT19123196

approvepapers.test.js

```
const app = require("../server");
const mongoose = require("mongoose");
const supertest = require("supertest");

describe("Sample Test", () => {
    it("should test that true === true", () => {
      expect(true).toBe(true);
    });
  });



  test("test Approve reseach papers", async () => {
    const result = await supertest(app)
    put('/api/paper/approveper/:_id')
      .send({
        status: "approved"

      });
    expect(result.statusCode).toBe(201);
  });

  test("test Reject reseach papers", async () => {
    const result = await supertest(app)
    put('/api/paper/rejectpaper/:_id')
      .send({
        status: "rejected"

      });
    expect(result.statusCode).toBe(201);
  });

test("test approve reseachpapers ", async () => {
    const result = await supertest(app)
      .get("/api/approvep/")
      .send({});
    expect(result.statusCode).toBe(200);
  });
```

Approveworkshops.test.js

```
const app = require("../server");
const mongoose = require("mongoose");
const supertest = require("supertest");

describe("Sample Test", () => {
    it("should test that true === true", () => {
      expect(true).toBe(true);
    });
  });

  test("test Approve Workshop", async () => {
    const result = await supertest(app)
    put('/api/approveworkshops/approve/:_id')
      .send({
        status: "approved"

      });
    expect(result.statusCode).toBe(201);
  });

  test("test Reject Workshop", async () => {
    const result = await supertest(app)
    put('/api/approveworkshops/reject/:_id')
      .send({
        status: "rejected"

      });
    expect(result.statusCode).toBe(201);
  });

test("test approve Workshop ", async () => {
    const result = await supertest(app)
      .get("/api/approveworkshops")
      .send({});
    expect(result.statusCode).toBe(200);
  });
```

User.test.js

```
const app = require("../server");
const mongoose = require("mongoose");
const supertest = require("supertest");

describe("Sample Test", () => {
    it("should test that true === true", () => {
```

```
      expect(true).toBe(true);
    });
  });

test("test register users without relevent data", async () => {
    const result = await supertest(app)
      .post("/api/signup")
      .send({});
    expect(result.statusCode).toBe(500);
  });

  test("test createPapers controller with relevent data", async () => {
    const result = await supertest(app)
      .post("/api/signup")
      .send({
        name: "saman perera",
        email: "saman98@gmail.com",
        password: "saman98",
        role:"researcher",
        contactnumber:"0711339887"

      });
    expect(result.statusCode).toBe(201);
  });

test("test getPapers controller ", async () => {
    const result = await supertest(app)
      .get("/api/allpper/")
      .send({});
    expect(result.statusCode).toBe(200);
  });
```

IT19159072

Papers.test.js

```
const app = require("../server");
const mongoose = require("mongoose");
const supertest = require("supertest");

describe("Sample Test", () => {
    it("should test that true === true", () => {
      expect(true).toBe(true);
    });
```

```
  });

test("test add papers without relevent data", async () => {
    const result = await supertest(app)
      .post("/api/paper/create")
      .send({});
    expect(result.statusCode).toBe(500);
  });

  test("test createPapers controller with relevent data", async () => {
    const result = await supertest(app)
      .post("/api/paper/create")
      .send({
        title: "Test Test",
        description: "testCat",
        email: "testCat-01",
        status:"testCat-02",
        // researchpaper:"testCat-03"
      });
    expect(result.statusCode).toBe(201);
  });

test("test getPapers controller ", async () => {
    const result = await supertest(app)
      .get("/api/allpper/")
      .send({});
    expect(result.statusCode).toBe(200);
  });
```

Workshops.test.js

```
const app = require("../server");
const mongoose = require("mongoose");
const supertest = require("supertest");

describe("Sample Test", () => {
    it("should test that true === true", () => {
      expect(true).toBe(true);
    });
  });

test("test addworkshops without relevent data", async () => {
    const result = await supertest(app)
      .post("/api/workshop/create")
```

```
      .send({});
    expect(result.statusCode).toBe(500);
  });

  test("test createWorkshops controller with relevent data", async () => {
    const result = await supertest(app)
      .post("/api/workshop/create")
      .send({
        date: "Test Test",
        email: "testCat",
        topic: "testCat-01",

      });
    expect(result.statusCode).toBe(201);
  });

test("test getWorkshops controller ", async () => {
    const result = await supertest(app)
      .get("/workshops")
      .send({});
    expect(result.statusCode).toBe(200);
  });
```

Reseachpaperpayment.test.js

```
const app = require("../server");
const mongoose = require("mongoose");
const supertest = require("supertest");

describe("Sample Test", () => {
    it("should test that true === true", () => {
      expect(true).toBe(true);
    });
  });

test("test add payment without relevent data", async () => {
    const result = await supertest(app)
      .post("api/payment/create")
      .send({});
    expect(result.statusCode).toBe(500);
  });

  test("test addPayment controller with relevent data", async () => {
    const result = await supertest(app)
```

```
      .post("api/payment/create")
      .send({
        name: "Test Test",
        date: "testCat",
        email: "testCat-01",

      });
    expect(result.statusCode).toBe(201);
  });
```

IT19129518

Conference.test.js

```
const app = require("../server");
const mongoose = require("mongoose");
const supertest = require("supertest");

describe("Sample Test", () => {
    it("should test that true === true", () => {
      expect(true).toBe(true);
    });
  });

test("test add conference without relevent data", async () => {
    const result = await supertest(app)
      .post("/api/conference/addconference")
      .send({});
    expect(result.statusCode).toBe(500);
  });

  test("test create conference controller with relevent data", async () => {
    const result = await supertest(app)
      .post("/api/conference/addconference")
      .send({
        title: "Test Test",
        description: "testCat",
        email: "testCat-01",
        status:"testCat-02",
        date:"testCat-03"

      });
```

```
    expect(result.statusCode).toBe(201);
  });

test("test getConference controller ", async () => {
    const result = await supertest(app)
      .get("/api/getconference")
      .send({});
    expect(result.statusCode).toBe(200);
  });
```

Conferencepayment.test.js

```
const app = require("../server");
const mongoose = require("mongoose");
const supertest = require("supertest");

describe("Sample Test", () => {
    it("should test that true === true", () => {
      expect(true).toBe(true);
    });
  });

test("test add payment without relevent data", async () => {
    const result = await supertest(app)
      .post("/payment/conference/create")
      .send({});
    expect(result.statusCode).toBe(500);
  });

  test("test addPayment controller with relevent data", async () => {
    const result = await supertest(app)
      .post("/payment/conference/create")
      .send({
        name: "Test Test",
        date: "testCat",
        email: "testCat-01",

      });
    expect(result.statusCode).toBe(201);
  });
```

IT19140308

approveconference.test.js

```
const app = require("../server");
const mongoose = require("mongoose");
const supertest = require("supertest");

describe("Sample Test", () => {
    it("should test that true === true", () => {
      expect(true).toBe(true);
    });
  });

  test("test Approve Conference", async () => {
    const result = await supertest(app)
    put('/api/approveconference/:_id')
      .send({
        status: "approved"

      });
    expect(result.statusCode).toBe(201);
  });

  test("test Reject Conference", async () => {
    const result = await supertest(app)
    put('/api/rejectconference/:_id')
      .send({
        status: "rejected"

      });
    expect(result.statusCode).toBe(201);
  });

test("test approve Conference ", async () => {
    const result = await supertest(app)
      .get("/api/getapproveconference")
      .send({});
    expect(result.statusCode).toBe(200);
  });
```