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_WE_24

Group Name: Hype Codex

Name	Registration Number	Functionality
YASAS R.M (Group leader)	**IT19133850**	Role Based Authentication, Innovation and Posts Management
Bandara M.H.M.N.D.T	**IT19152288**	Workshop management and Conference Management
Gunawardana K.H.R	**IT19215884**	Research paper management
Dhanasekara D.M.S.M	**IT19056258**	Reviewers management

Repository Link: https://github.com/manuka99/Conference-Management-Tool

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:

Table of Contents

1 Introduction

The functionalities of Conference management tool are developed as REST API by using Node JS and Express JS. Node JS is the most suitable technology to develop a fast, scalable and small size (less complex) server-side application.

Mongo DB is a no SQL / document based database which stores unstructured data. Mongo DB is faster, scalable and can support large amount of data in comparison to the SQL server. Therefore, Mongo DB is used for the database tier.

Express JS is used as the framework to develop this application. Requests sent to the server will be handled by the respective endpoint (controller). Any operation related to database will be handled through the data access layer (DAO). Schemas created using mongoose library are used to perform mongo database operations. Once the request is processed without any error, **JSON** response will be sent back using the utility created. To handle errors, an error handling middleware is created and it will handle the error and send an error response back.

Environment constants will be saved in environment configuration file. Donenv library is used to load these environment variables to the application.

Other Dependencies used in Server

Passport JS – Used as an authentication middleware.

Jsonwebtoken – To generate signed Json web tokens.

Bcrypt – To encrypt user sensitive data such as password.

Nodemailer – To send email from Node JS.

Donenv – To load environment variables in Node Js

Mongoose – Schema based solution to model application data using mongo DB in Node JS

CORS - Express middleware that can be used to enable **CORS** with various option in Node JS.

Device detector Js – To detect the user device details through useragent in request body.

Twilio – To send SMS through twilio cloud messaging service in Node Js.

Enum – To save read-only values with JavaScript (Not Typescript).

express-fileupload – To upload files easily to the database.

express-validator – To validate request data.

Lodash – Some functions are used when handling data.

React is an open-source front-end JavaScript library for building user interfaces or UI components. React is more simple and flexible compared to other competitors. Therefore, React is used to develop user interfaces of the application (MERN stack).

Material UI library is used to style the web client. Asynchronous HTTP requests are made to the REST endpoints using the Api function which is defined as a utility. Api function uses Axios library to make http request. Additionally, interceptors are used in the Api function to handle errors sent back from the rest end points.

To improve performance and avoid loading all the components / pages initially, lazy loading is used. For lazing loading @loadable/component Library is used. Routing to pages is done using react router v6. To improve user experience, Sweet alert library is used to respond for request responses.

Other Dependencies used in Client

Material UI – To develop frontend user interfaces in React

Redux – To manage state globally in React Js (for authentication).

React Router V6 – For routing in React Js.

Axios – To make requests (XMLHttpRequests) from React to the Server.

Sweetalert2 – To display beautiful alerts.

payhere-js-sdk – To process payments with payhere payment gateway.

Moment – To display date time values in human readable format.

react-doc-viewer – To view uploaded files in browser.

loadable/component – Lazy load react components.

react-topbar-progress-indicator – To display loading progress until pages are loaded.

2 Functionalities of the system (Member wise)

2.1 IT19133850 – Role Based Authentication and Innovation Management

2.1.1 Description

Conference management tool is a web application, which will be used by many users with different user roles. In order to secure data and information among user groups, a secure authentication mechanism must be used. The backend / server of the entire system will be developed using Node JS. Therefore, an authentication system based on JSON Web Token will be used. JSON Web Token are one of the secure practice or authentication method used to identify authorized users of the system. Guest users have to verify their credentials (Email and password) once and in return they will get a unique token which is allowed to access for 45 minutes. The token will have user's basic details and role details.

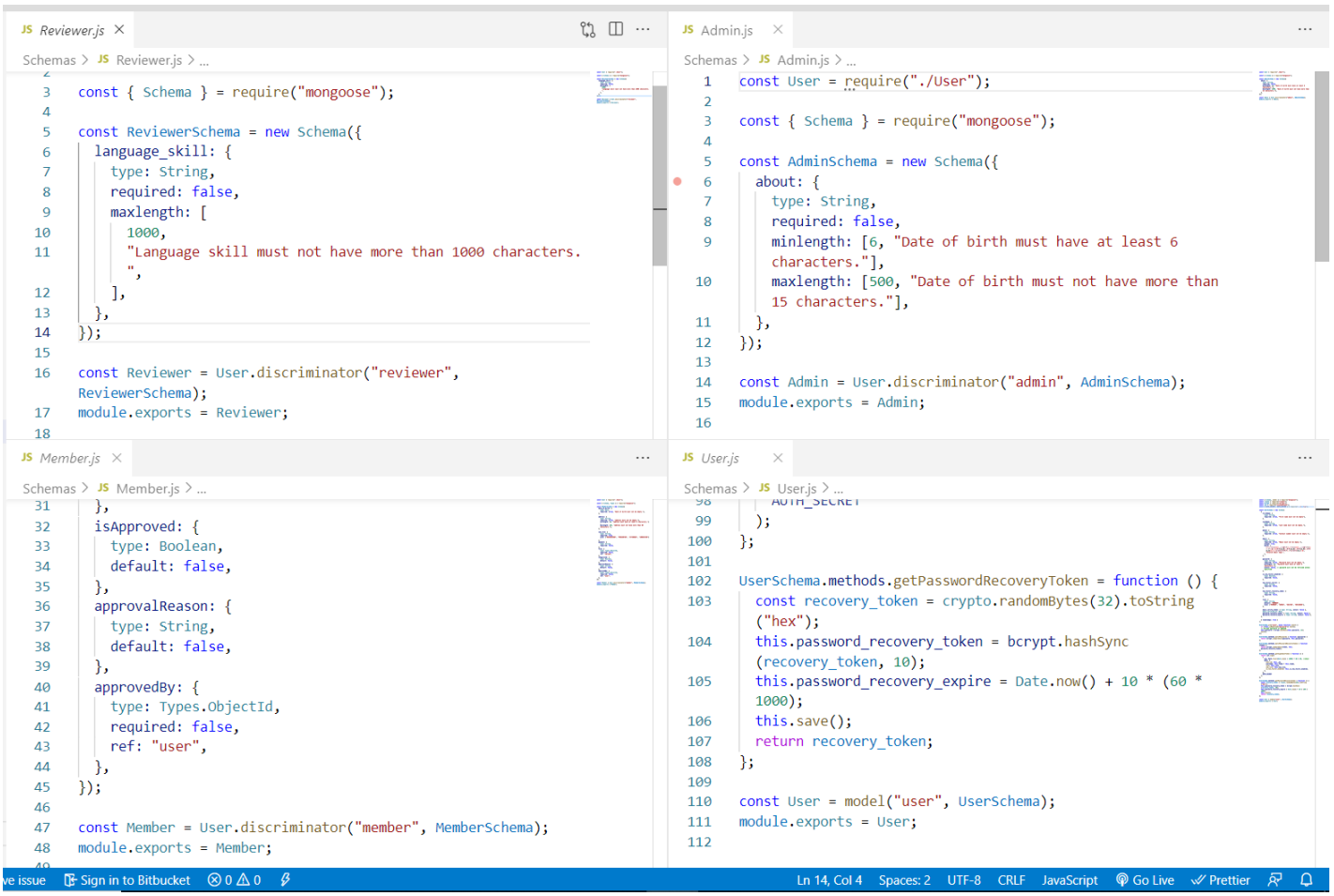
When registering a user and resetting a password, password is encrypted (using bcrypt) and stored in the database. Even when querying we can get only the highest version of the password unless specially quarried for the real password.

User can request to reset the password then a hashed reset token will be generated and an email will be sent to the user's email address along with the reset URL. Once user visit the reset URL, server checks the validity of the reset token and allows changing the password.

There is no specific method to logout the user from application when using JWT. In order to perform logout functionality, a middleware will be created to verify authorization token of each request, if the request was verified (authorized) when the token will be saved or updated in the database. If the user logout of the application, then the current token which is saved in the database will be retrieved and it's property "isValid" is set to false and updated. To avoid user authenticating using the same token, a middleware is created to check the database if the property "isValid" of the token set to false or not. If it is set to false then the user will not be authenticated with the system. If the token is not present in database or if the property "isValid" is set to true then the jwt token will be verified. If the token was verified then the user object will be retrieved from the database and appended to the request (request.user).

Some functions must only be accessed by authorized users, therefore to verify the request is authenticated or not a middleware is created to check if the request has a user object (request.user). In order to perform role-based authentication a similar middleware is created in which it will accept an array of roles. In this middleware it will check if the user object in the request object include those roles, boolean response is returned.

Based on the requirements of the system, there are few user roles such as admin, editor, reviewer and member. Through the guest registration, a user will be given a role of member. Other user roles can be assigned to a user only by an administrator. Since users need to maintain property (data) based on their role, inheritance was used.



Reviewer.js

```
const { Schema } = require("mongoose");

const ReviewerSchema = new Schema({
  language_skill: {
    type: String,
    required: false,
    maxlength: [
      1000,
      "Language skill must not have more than 1000 characters.
      ",
    ],
  },
});

const Reviewer = User.discriminator("reviewer",
ReviewerSchema);
module.exports = Reviewer;
```

Admin.js

```
const User = require("./User");

const { Schema } = require("mongoose");

const AdminSchema = new Schema({
  about: {
    type: String,
    required: false,
    minlength: [6, "Date of birth must have at least 6
    characters."],
    maxlength: [500, "Date of birth must not have more than
    15 characters."],
  },
});

const Admin = User.discriminator("admin", AdminSchema);
module.exports = Admin;
```

Member.js

```
  },
  isApproved: {
    type: Boolean,
    default: false,
  },
  approvalReason: {
    type: String,
    default: false,
  },
  approvedBy: {
    type: Types.ObjectId,
    required: false,
    ref: "user",
  },
});

const Member = User.discriminator("member", MemberSchema);
module.exports = Member;
```

User.js

```
  );
};

UserSchema.methods.getPasswordRecoveryToken = function () {
  const recovery_token = crypto.randomBytes(32).toString
  ("hex");
  this.password_recovery_token = bcrypt.hashSync
  (recovery_token, 10);
  this.password_recovery_expire = Date.now() + 10 * (60 *
  1000);
  this.save();
  return recovery_token;
};

const User = model("user", UserSchema);
module.exports = User;
```

Through the guest registration, user will be registered as a researcher, workshop presenter, innovator, and attendee. After the completion of the registration, user will have given a role of member and a sub role of the type user selected in the form (researcher, workshop presenter or attendee). At the registration, all users must include their basic details such as full name, email, contact number and birth of date.

Additionally, there are certain data that user must submit based on the sub role user selects. To register as an innovator, user must submit a proposal containing all the necessary details about the innovation in pdf format. Additionally, Innovators are required to pay an amount of five hundred rupees to register for the conference. Attendees are required to pay LKR 500 to register. Both researchers and presenter must submit their proposal befour registering to the system. User can complete the payment through credit /debit cards, EZ cash or by Dialog Genie. Payhere payment

gateway will be implemented to support the above requirement. All users are required to be approved by the admin.

If a user was approved by the admin then the user can present in the conference.

To validate request data validators in express validators are used with custom functions.

All the data will be in one file and based on the actions validations can be picked and appended.

Example 1: Validating a user registration as a member.

These are the rules that are common to any user registration. (URegistrationRules)

UserRules.js

Validation > UserRules.js > ...

```
1   const { Validation } = require(".");
2   const { UserEnum } = require("../Models/UserModel");
3
4   exports.LoginRules = [Validation.email(), Validation.password()];
5
6   // for public registration
7   exports.PRegistration = [Validation.includes("role", UserEnum.MEMBER.value)];
8
9   // for administrators registration (except members)
10  exports.ProtectedRegistration = [
11    Validation.includes(
12      "role",
13      UserEnum.ADMIN.value,
14      UserEnum.EDITOR.value,
15      UserEnum.REVIEWER.value
16    ),
17  ];
18
19  // for any user reistration
20  exports.URegistrationRules = [
21    Validation.text("firstName", 4, 20),
22    Validation.text("lastName", 4, 20),
23    Validation.phone(),
24    Validation.unique_user_email(),
25    Validation.password(),
26    Validation.confirm_password(),
27    Validation.includes(
28      "role",
29      UserEnum.ADMIN.value,
30      UserEnum.EDITOR.value,
31      UserEnum.REVIEWER.value,
32      UserEnum.MEMBER.value
33    ),
34  ];
35
36  exports.UserProfileUpdateRules = (req) => {
37    const { firstName, lastName, phone, email } = req.body;
38    var rules = [];
39
```

Rules related to validating member. In line no 9, all the rules that must be valid to register a user are added to the member registration rules. Additionally, other properties that must be validated for a member are added. Also, some rules based on the sub role selected are added.

MemberRules.js

Validation > MemberRules.js > ...

```
1   const { Validation } = require(".");
2   const { MemberEnum } = require("../Models/UserModel");
3   const { URegistrationRules, UserProfileUpdateRules } = require("./UserRules");
4
5   // member registration rules
6   exports.MRegistrationRules = (data) => {
7     const { sub_role } = data;
8     const validations = [
9       ...URegistrationRules,
10      Validation.date("date_of_birth"),
11      Validation.text("address", 10, 150),
12      Validation.includes(
13        "sub_role",
14        MemberEnum.RESEARCHER.value,
15        MemberEnum.PRESENTER.value,
16        MemberEnum.ATTENDEE.value,
17        MemberEnum.INNOVATOR.value
18      ),
19    ];
20
21    if (sub_role) {
22      switch (sub_role) {
23        case MemberEnum.INNOVATOR.value:
24          validations.push(Validation.file());
25          validations.push(Validation.text("payment"));
26          break;
27
28        default:
29          break;
30      }
31    }
32
33    return validations;
34  };
35
36  exports.MemberProfileUpdateRules = (req) => {
37    const { date_of_birth, address, sub_role } = req.body;
38    const { files } = req;
39
40    // add user profile rules
```

issue Sign in to Bitbucket ⊗ 0 ⚠ 0

When a user registration request arrives, the process will be handed over to the endpoint based on the role.

In the member registration endpoint the request validation will be done and if any error, a custom error of validation will be thrown.

```
/* Validations */
const ValidateMemberRegistration = async (req) => {
  await Promise.all(
    MRegistrationRules(req.body).map((validation) => validation.run(req))
  );
  ValidateRequest(req);
};

const ValidateMemberProfileUpdate = async (req) => {
  await Promise.all(
    MemberProfileUpdateRules(req).map((validation) => validation.run(req))
  );
  ValidateRequest(req);
};

// to register member
exports.MemberRegistration = async (req, res, next) => {
  try {
    // validations
    await ValidateMemberRegistration(req);

    const memberData = lodash.pick(req.body, [
      "firstName",
      "lastName",
      "phone",
      "email",
      "date_of_birth",
      "address",
      "payment",
      "password",
      "sub_role",
    ]);

    // register member
    var user = await MemberDao.createNewMember(memberData);

    // upload file if present
    if (req.files && req.files.file) {
      const upload = await UploadDau.UploadFile(
        req.files.file,
        "innovations",
        user._id
      );
      // update user
      user = await MemberDao.updateMember(user._id, { file: upload._id });
    }

    sendSuccess(res, { user, token: user.getSignedJwtToken() });
  } catch (error) {
    next(error);
  }
};
```

Validate request in line no 18 will throw a custom error if validation fails.

UserEndpoint.js | ValidateRequest.js

Middlewares > ValidateRequest.js > ...

```
1   const { validationResult } = require("express-validator");
2   const ValidationError = require("../Common/ValidationError");
3   const _ = require("lodash");
4
5   exports.ValidateRequest = (req, res, next) => {
6     const errors = validationResult(req);
7     if (!errors.isEmpty())
8       throw new ValidationError(
9         "There are some items that require your attention",
10        errors.array()
11      );
12    if (next) next();
13  };
14
```

Thrown error will be handled by the express app middleware.

HandleError.js

Middlewares > HandleError.js > ...

```
1   const mongoose = require("mongoose");
2   const { sendError, FormatValidationError } = require("../Common/util");
3   const ValidationError = require("../Common/ValidationError");
4   const ApplicationError = require("../Common/ApplicationError");
5   const AccessForbiddenError = require("../Common/AccessForbiddenError");
6
7   exports.HandleError = (err, req, res, next) => {
8     console.error(`Error Handler: ${err.message}`);
9     if (err instanceof mongoose.Error) sendError(res, err, 422);
10    else if (err instanceof ValidationError)
11      sendError(res, FormatValidationError(err), 422);
12    else if (err instanceof AccessForbiddenError)
13      sendError(res, { msg: err.message, err }, 403);
14    else if (err instanceof ApplicationError)
15      sendError(res, { msg: err.message, err }, 400);
16    else sendError(res, { msg: err.message, err }, 500);
17  };
18
```

If the error is a validation error then the formatted data will be send. The reason to format data is because express-validator errors are not organized based on parameter therefore it is difficult to display in the client.

util.js

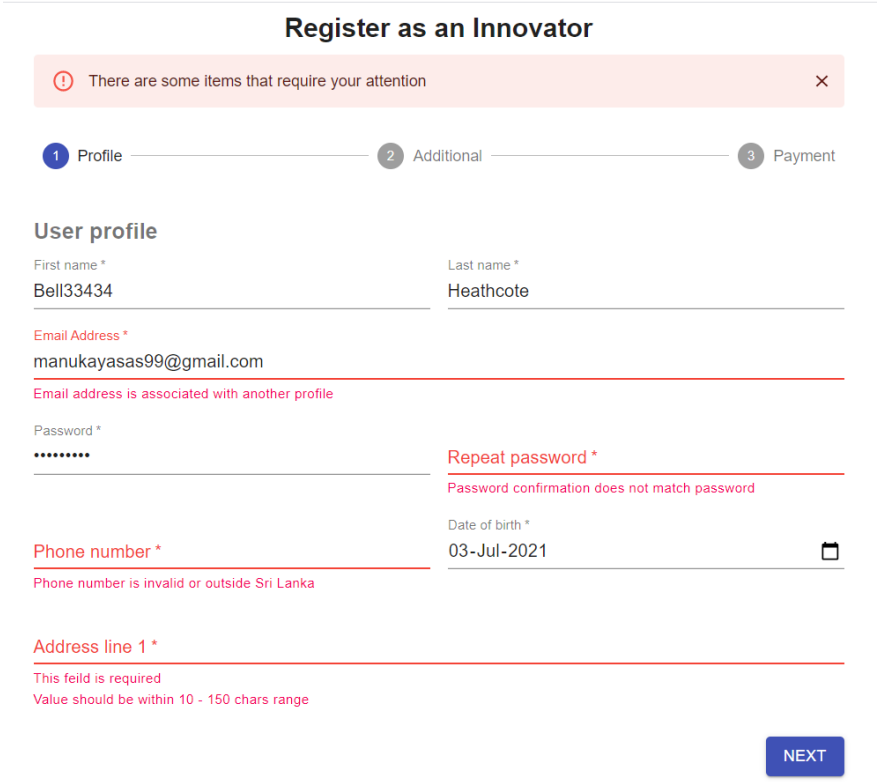
Common > util.js > <unknown>

```
  res.status(200).json({ success: true, data });
};

const sendError = (res, data, errorCode = 400) => {
  res.status(errorCode).json({ success: false, data });
};

// format validations based on params and message
const FormatValidationError = (err) => {
  try {
    if (!err) return err;
    else {
      const { message, data } = err;
      var params = {};
      if (data && Array.isArray(data) && data.length > 0) {
        for (let index = 0; index < data.length; index++) {
          const error = data[index];
          if (error.param in params) params[error.param].push(error.msg);
          else params[error.param] = new Array(error.msg);
        }
        return { msg: message, params };
      }
      return { msg: message };
    }
  } catch (error) {
    Consola.error("FormatValidationError: ", err);
    return err;
  }
};
```

After formatting the data using this function we will get a array of validation errors bases on the attribute. That can be easily displayed as below.



Files will be uploaded to the application uploads directory and additional information will be saved in the database.

FileAccess.js | Upload.js | index.js | UserModel.js

Schemas > Upload.js > UploadSchema > type

```
1   const User = require("./User");
2
3   const { Schema, model } = require("mongoose");
4
5   const UploadSchema = new Schema(
6     {
7       submit_name: {
8         type: Schema.Types.String,
9         required: true,
10      },
11      name: {
12        type: Schema.Types.String,
13        required: true,
14      },
15      user: {
16        type: Schema.Types.String,
17        required: true,
18      },
19      path: {
20        type: Schema.Types.String,
21        required: true,
```

A middleware is used to restrict file access to unauthorized users. In here only users who are the original creators and users with role admin and reviewer are able to access.

To perform role based authentication a middleware is created to validate the user role.

JS RoleAuth.js

Middlewares > JS RoleAuth.js > ...

```
1   const { sendError } = require("../Common/util");
2
3   exports.RoleAuth = (roles) => (req, res, next) => {
4     var hasRole = roles.includes(req.user.role);
5     if (!hasRole) {
6       sendError(
7         res,
8         {
9           msg: "You are not authorized or permitted for this content!",
10        },
11        403
12      );
13      return false;
14    } else if (hasRole && next) next();
15    else return hasRole;
16  };
17
```

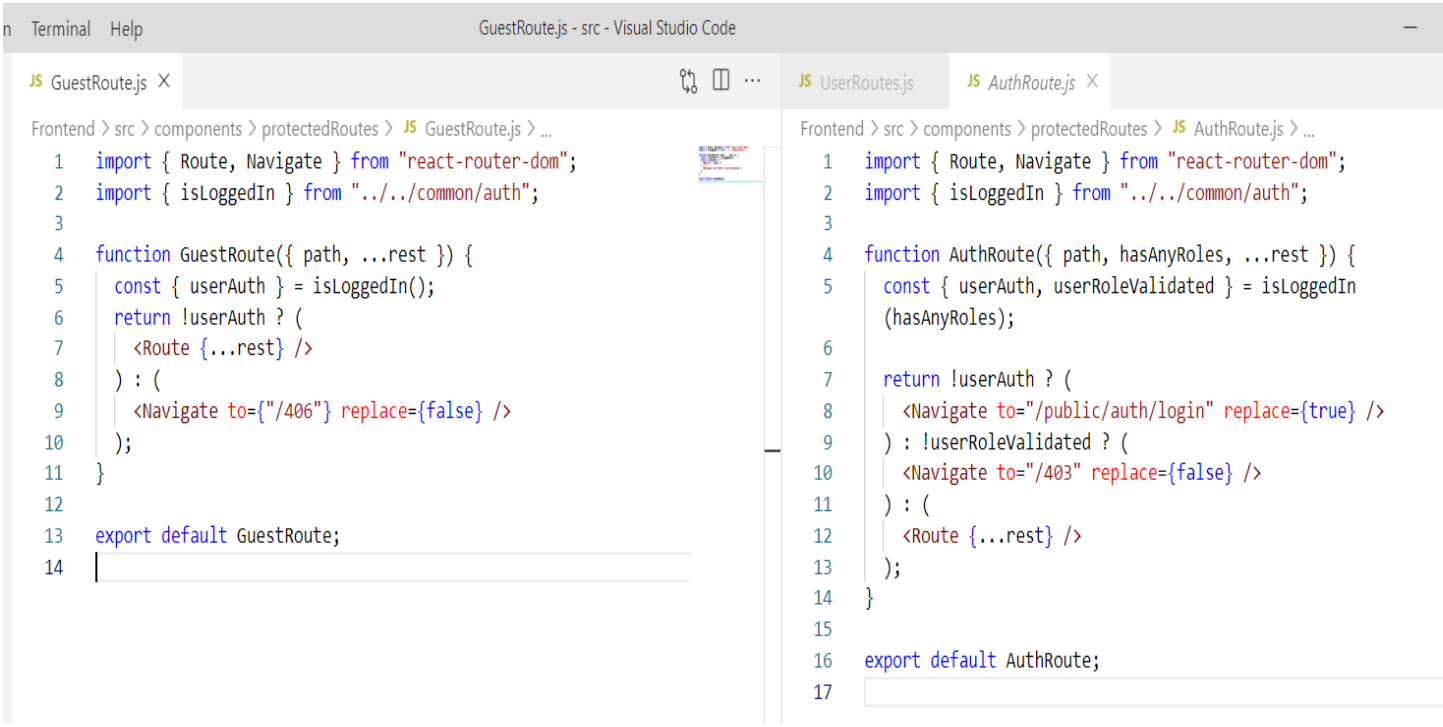
This middleware can be applied to the endpoint or the routes , below is the way how it can be applied in the endpoint,

```
125
126   // Roles of Admin
127   exports.DeleteMember = (req, res, next) => {
128     // validate roles
129     var validator = RoleAuth([UserEnum.ADMIN.value]);
130     if (!validator(req, res)) return -1;
131     // delete
132     const { id } = req.params;
133     MemberDao.deleteMember(id)
134       .then((user) => sendSuccess(res, { user }))
135       .catch(next);
136   };
137
```

It can also be applied to routes in this way

Here to access any request request related to api/reviewer , user must be authenticated and must have a user role of Admin or Reviewer.

Routes must be protected in react app as well therefore redux is used to save the global state of the app users and custom routes are created to protect routes.



Guest routes are used to validate user is not authenticated. Routes such as login register can be protected by using this custom routes.

Auth routes are used to validate user is authenticated and valid role is present. Routes such as administrator panel can be protected by using this custom route.

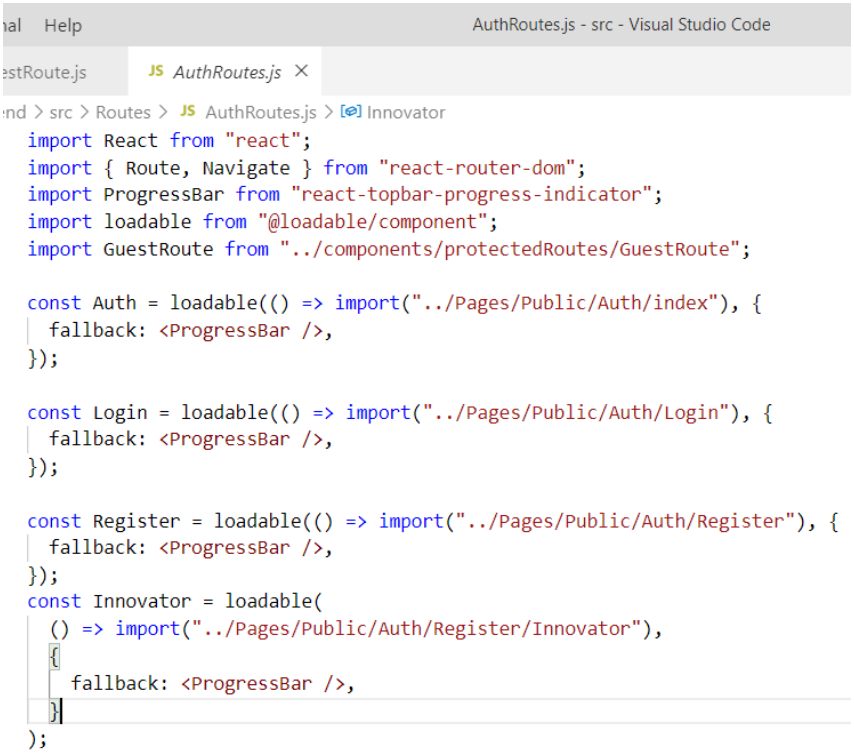
These can be applied as below.

```
export default function UserRoutes() {
  return (
    <AuthRoute hasAnyRoles={ReviewerRoles} path="/users">
      <AuthRoute path="/roles/:role_name" element={<ViewUsers />} />
      <AuthRoute path="/:id" element={<ViewUser />} />
    </AuthRoute>
  );
}
```

```
function AuthRoutes() {
  return (
    <Route path="/auth" element={<Auth />}>
      <GuestRoute path="/" element={<Navigate to="/public/auth/login" />} />
      <GuestRoute path="/login" element={<Login />} />
      <GuestRoute path="/register">
        <GuestRoute path="/" element={<Register />} />
        <GuestRoute path="/innovator" element={<Innovator />} />
        <GuestRoute path="/researcher" element={<Researcher />} />
        <GuestRoute path="/presenter" element={<Presenter />} />
        <GuestRoute path="/attendee" element={<Attendee />} />
      </GuestRoute>
      <GuestRoute path="/recover-password" element={<RecoverPassword />} />
      <GuestRoute
        path="/reset-password/:token/:email/:username"
        element={<ResetPassword />}
      />
    </Route>
  );
}

export default AuthRoutes;
```

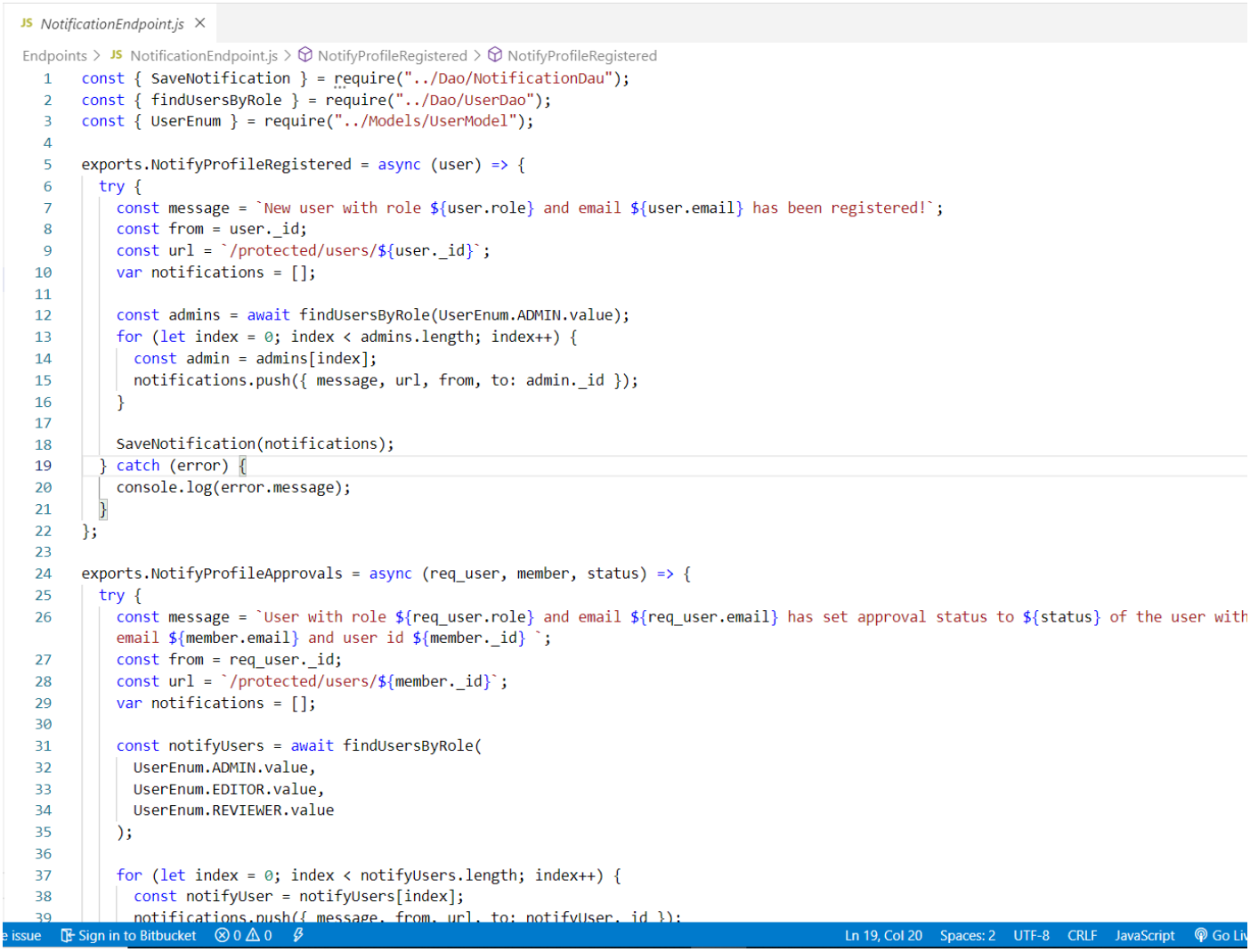
Also to lazy load components library loadable is used.



Additionally users with role Editor can publish posts through the application but they must be approved by the admin before it get's published in the public wall.

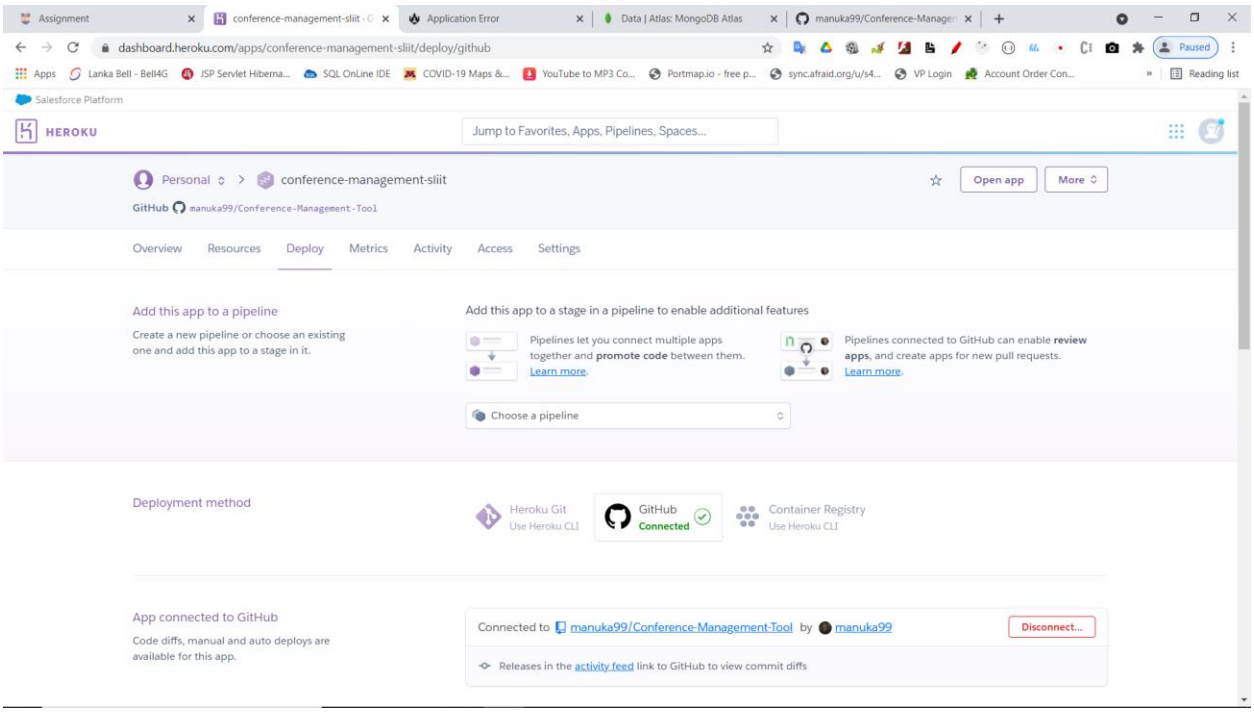
Twilio cloud communications are integrated to send SMS messages to users to reset their password. Also node mailer is used to send email messages.

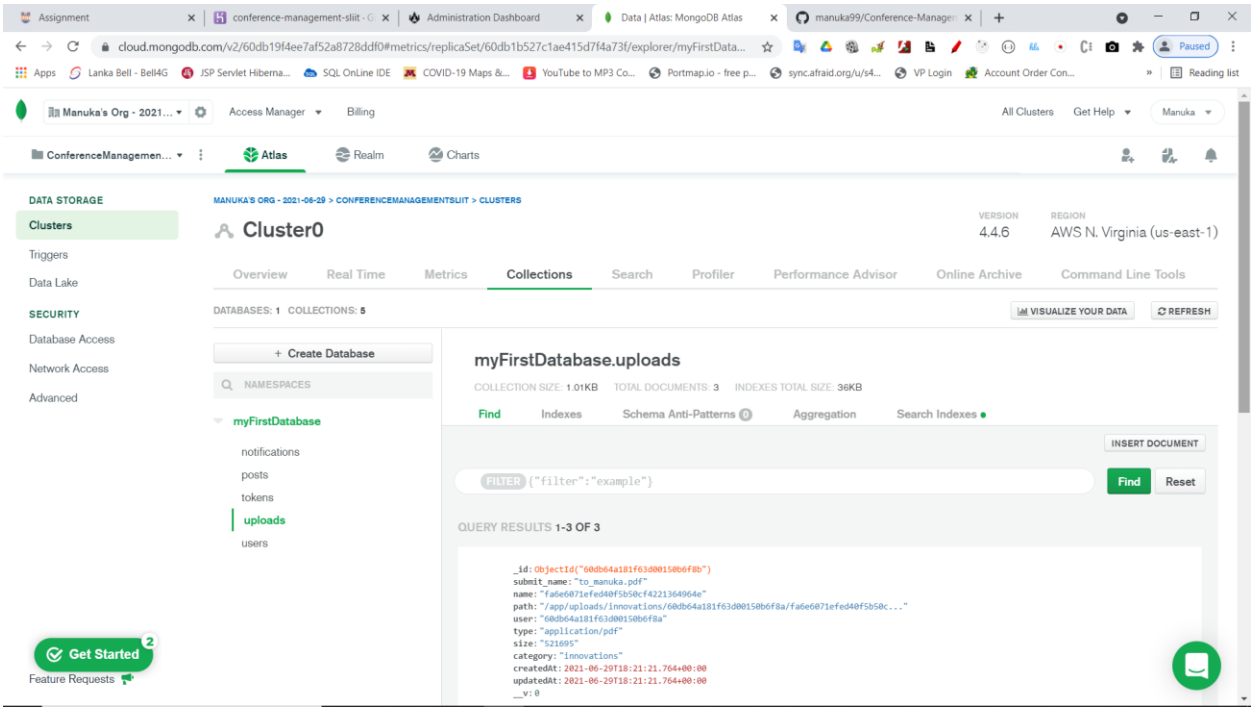
For every registration user will be sent an welcome email. In addition, for all these activities Administrators will receive in app notifications (user profile activation, registration, submit of new post). Below is the app notification service.



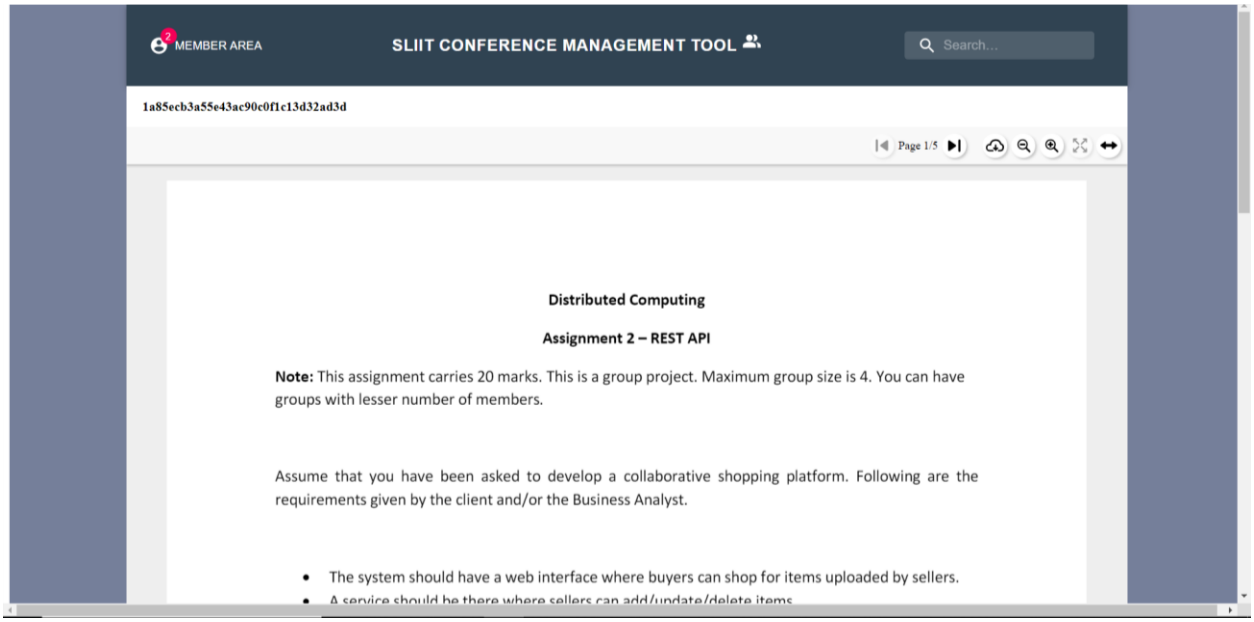
All the functionalities I have implemented have been deployed at heroku cloud. Atlas mongo database is used. Below is the url of the website.

http://conference-management-sliit.herokuapp.com/





View Files in web browser

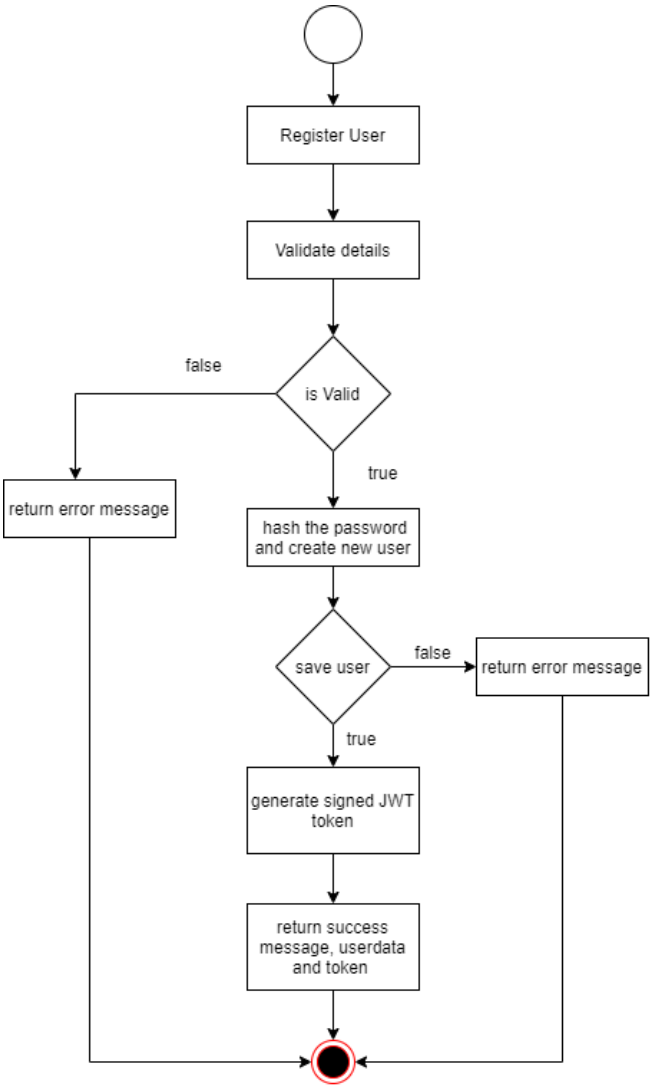


2.1.2 Rest API

Process of registering a new user

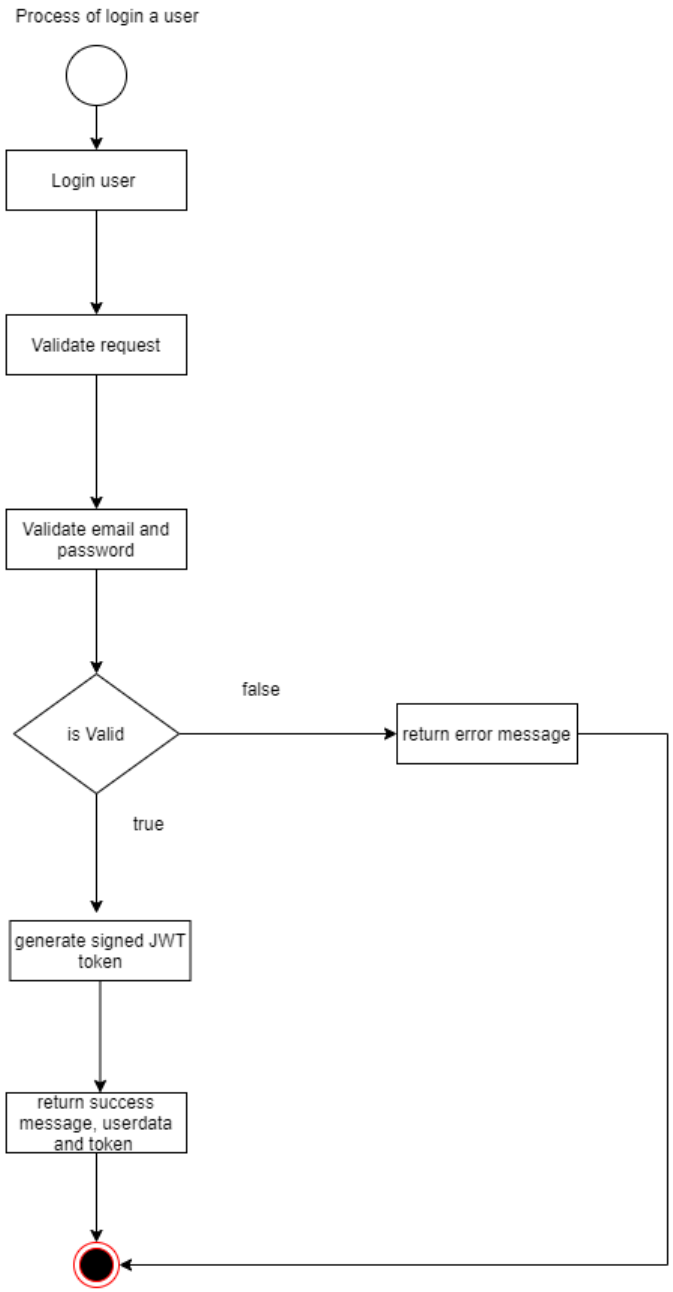
A POST request with the details (first name, last name, phone, email, password, repeat password and user type) will be sent to the authentication service. The email should be unique, and the password and repeat password must match. Certain validations are checked based on the user type selected. An innovator must include a proposal containing all the necessary details about the innovation and must be in pdf. In addition, they are required to pay an amount of five hundred rupees to register for the conference. The request validation process was explained above.

If the request is valid, a new user is created by adding new JSON document to the user collection in mongoDB. After successful registration a Json web token for authentication will be created (auth_token) and will be return along with the user data. It will be used for the authentication of the user when login to the system. The password is converted to a hash code for better privacy using bcrypt and stored. When querying, password is not passing unless specially asked for it.



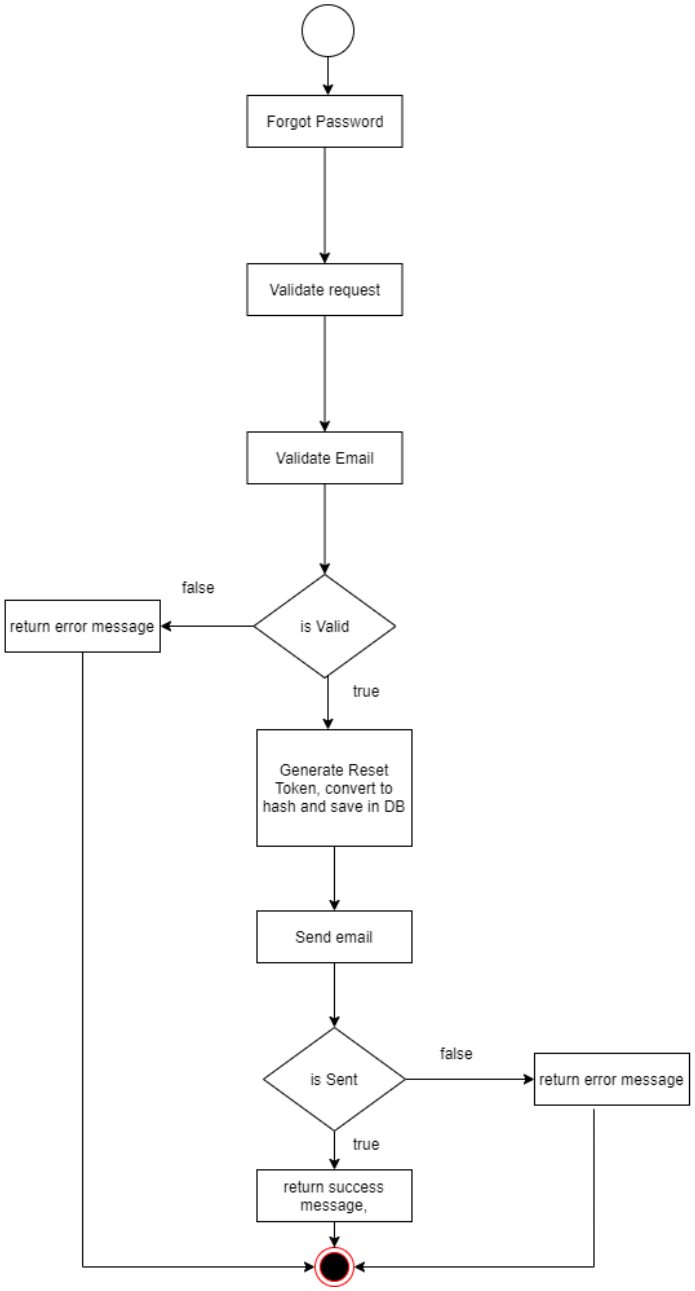
Process of login

A POST request with email and password will be sent to the authentication service. Once request is received the system check the availability of the user by searching for the email and match the passwords. If the passwords match, a sign in token will be generated by adding all details in user object except password and returned. If the user is not found, an error response with status code of 400 will be returned.



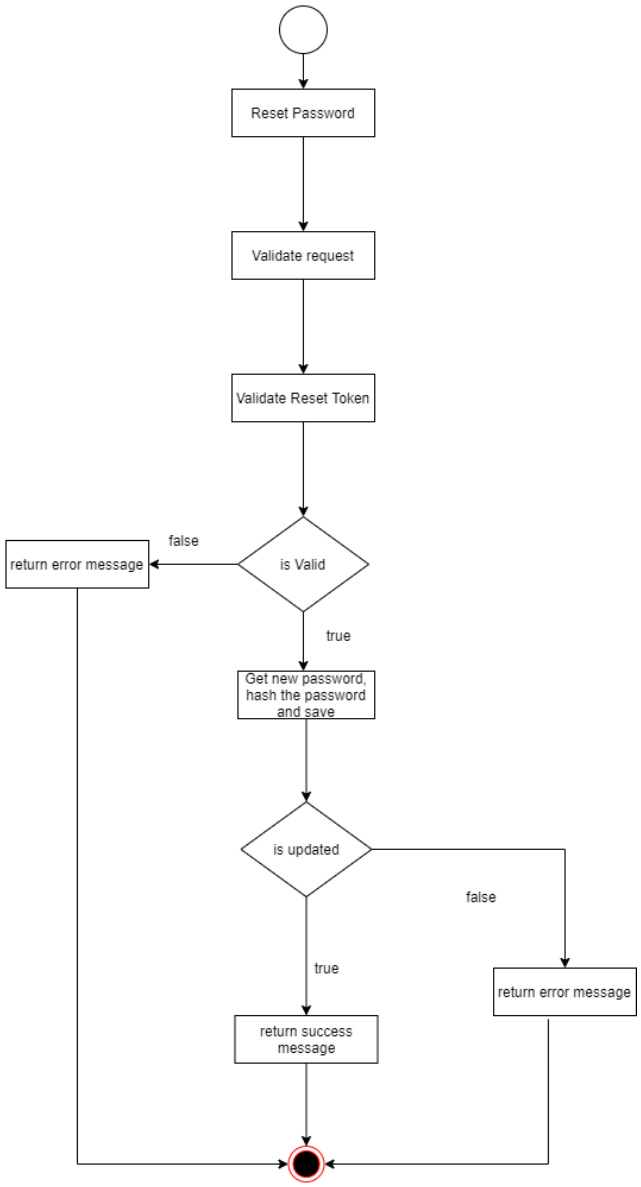
Process of password recovery

A POST request with the email will be sent to the authentication service. Verify the email provided exists in database. If the email exists a reset token is created which is expired in 10mins, and the hashed version of the token will be stored in the database. An email with the http message embedded with the reset url, email and the reset token will be sent to the provided email address. Nodemailer and sendGrid is used for this purpose. Once the email sending procedure is successfully done, success message will be generated. Otherwise, an error response with status code of 400 will be returned.



Process of resetting user password

A PATCH request with a reset token, email, new password and repeat new password will be sent to the authentication service. User object will be fetched from the database from the email provided. Bycrypt is used to compare the token in the URL parameters with the hashed reset token in the user object retrieved. If the validation is success, new password will be matched with the repeat password. If passwords match, new password will be hashed and the user data will be updated. After a successful password reset, a bearer token for authentication is also generated and returned. If some error occurred during the process, an error response with status code of 400 will be returned.



Process of logging out the user

A POST request will be sent to the authentication service. Token object is retrieved from the database based on the token in the authorization header in the request. If the token exists its "isValid" property is set to false and updated else a new token object is created and saved setting it's "isValid" property to false.

Process of view all innovations

A GET request will be sent to the innovation service. An authenticated user with role admin and editor can view all innovations. If successfully validated, all the innovations submitted by the users in the DB will be fetched. In case if an error occurred, an error response with status code of 400 will be returned.

Process of view one innovation

A GET request with the innovation id will be sent to the innovation service. An authenticated user with role admin and editor can view innovations. If successfully validated, the innovation id will be validated and if there is no error, the innovation corresponding to the id will be fetched. In case if an error occurred, an error response with status code of 400 will be returned.

Process of updating an innovation

A Patch request with the innovation id and updated details will be sent to the innovation service. An authenticated user with role admin and editor can update innovations. If the request was successfully validated, innovation details will be updated and saved in the DB. If there is error in the process, an error response with status code of 400 will be returned.

Process of deleting an innovation

A DELETE request with the innovation id will be sent to the innovation service. An authenticated user with role admin can delete innovations. After the authentication, the request will be validated. Then the availability of the innovation object is checked and if available the innovation will be deleted. If there is error in the process error message will be returned.

Process of view all posts

A GET request will be sent to the post service. Any user can view all the approved posts. To view unapproved posts the user must be an administrator or the author of the post. All the posts in the DB will be fetched. In case if an error occurred, an error response with status code of 400 will be returned.

Process of view post

A GET request with the post id will be sent to the post service. An authenticated user with role admin and author can view the post if it was unapproved else any user can view it. If successfully validated, the post id will be validated and if there is no error, the post corresponding to the id will be fetched. In case if an error occurred, an error response with status code of 400 will be returned.

Process of approving a post

A Patch request with the post id and approval status will be sent to the post service. An authenticated user with role admin can approve posts. If the request was successfully validated, post details will be updated and saved in the DB. If there is error in the process, an error response with status code of 400 will be returned.

Process of deleting a post

A DELETE request with the post id will be sent to the post service. An authenticated user with role admin or its author can delete a post. After the authentication, the request will be validated. Then the availability of the post object is checked and if available the post will be deleted. If there is error in the process error message will be returned.

2.1.3 Test cases

Test Case 01		
Author	User	
Summary	Register to the system as a researcher.	
Precondition	User must have a valid internet connection. User must have a valid email address.	
Step No.	**Step Action**	**Expected output**
1	User visit to the homepage.	Load the landing page.
2	User click on register button.	Navigate to user registration form.
3	User enter first name.	Text field should accept input data.
4	User enter last name.	Text field should accept input data.
5	User enter email.	Text field should accept input data.
6	User enter phone number.	Text field should accept input data.
7	User enter date of birth	Text field should accept input data.
8	User enter address	Text field should accept input data.
9	User enter password.	Text field should accept input data.
10	User enter same password in repeat password text field.	Text field should accept input data.
11	User select user type as researcher.	User type researcher is selected. Research paper upload field is visible.
12	User select research file,	File field should accept selected file.
13	User clicks on “Register” button.	Application alerts, “Your registration details were successfully saved” and navigates to the home page.

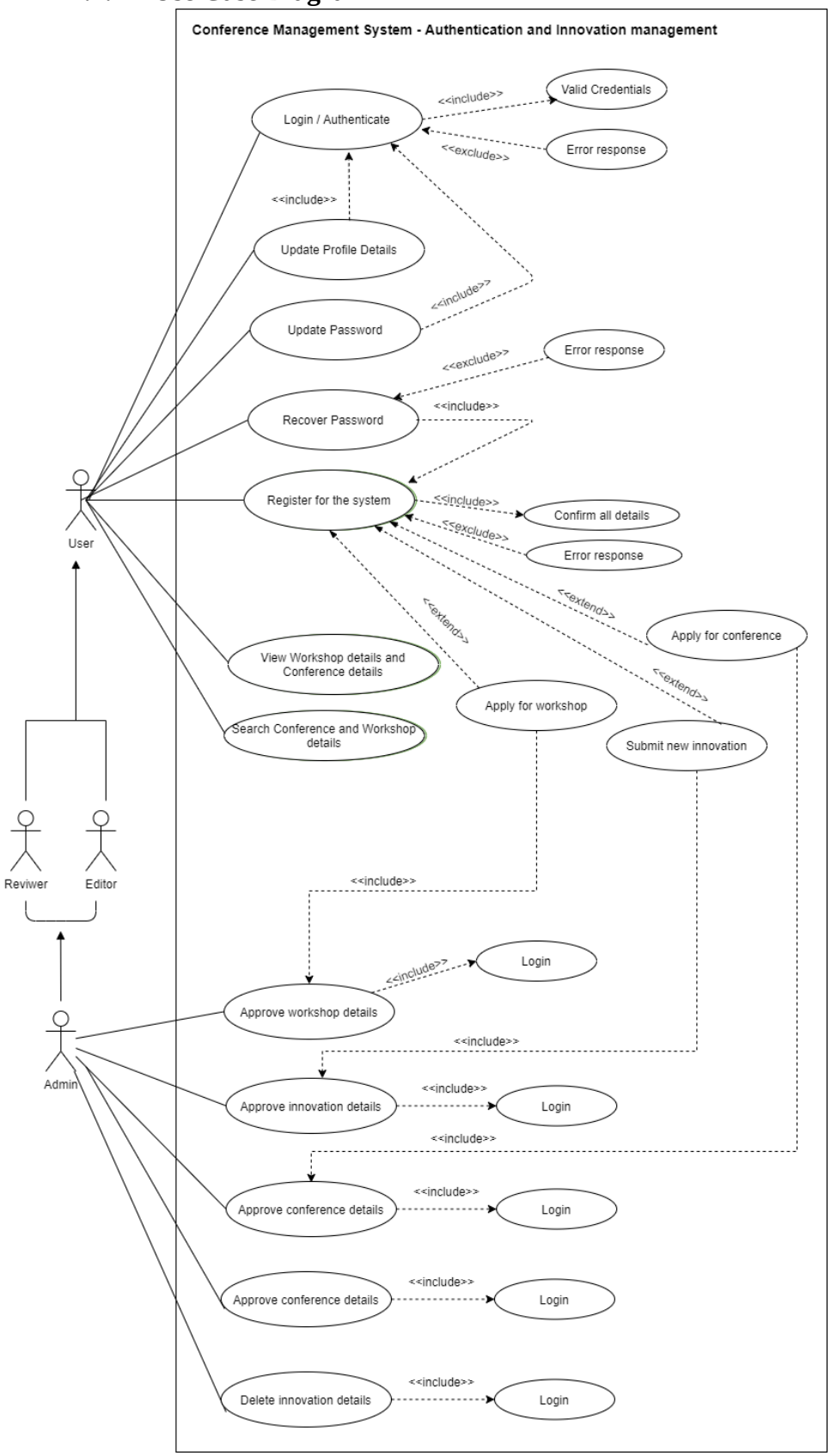
Test Case 02		
Author	User	
Summary	Register to the system as an attendee.	
Precondition	User must have a valid internet connection. User must have a valid email address. User must have a valid credit card with a balance of at least Rs. 500.00.	
Step No.	**Step Action**	**Expected output**
1	User visit to the homepage.	Load the landing page.
2	User click on register button.	Navigate to user registration form.
3	User enter first name.	Text field should accept input data.
4	User enter last name.	Text field should accept input data.
5	User enter email.	Text field should accept input data.
6	User enter phone number.	Text field should accept input data.
7	User enter date of birth	Text field should accept input data.
8	User enter address	Text field should accept input data.
9	User enter password.	Text field should accept input data.
10	User enter same password in repeat password text field.	Text field should accept input data.
11	User select user type as attendee.	Payment form is displayed.
12	User enter credit card number.	Text field should accept input data.
13	User enter CVV number.	Text field should accept input data.
14	User enter expiration date.	Text field should accept input data.
15	User clicks on “Register” button.	Application alerts, “Your registration details were successfully saved” and navigates to the home page.

Test Case 03		
Author	User	
Summary	Login to the system.	
Precondition	User must have a valid internet connection. User must have a valid registered account.	
Step No.	**Step Action**	**Expected output**
1	User visit to the homepage.	Load the landing page.
2	User click on “login” button.	Navigate the user login page.
3	User enter username.	Text field should accept input data.
4	User enter password.	Text field should accept input data.
5	User clicks on “Login” button.	Application alerts, “You are successfully login” and navigates to the home page.

Test Case 04		
Author	User	
Summary	Update user’s contact details.	
Precondition	User must have valid internet connection. User must be an authenticated user.	
Step No.	**Step Action**	**Expected output**
1	Visit home page.	Load the landing page.
2	User click on “Manage profile” button.	Navigate to the update profile page.
3	User edit first name.	Text field should accept input data.
4	User edit last name.	Text field should accept input data.
5	User edit phone number.	Text field should accept input data.
6	User edit address.	Text field should accept input data.
7	User clicks “Update contact details” button.	Application alerts, “You update details were successfully saved”.

Test Case 05		
Author	User	
Summary	Recover Password	
Precondition	User must have valid internet connection. User must have a valid registered account.	
Step No.	**Step Action**	**Expected output**
1	Visit home page.	Load the landing page.
	User click on “Recover Password” button.	Navigate to the recover password page.
2	User enter email address.	Text field should accept input data.
3	User click on “Recover” button.	Application alerts, “Reset URL has been sent to your email”.

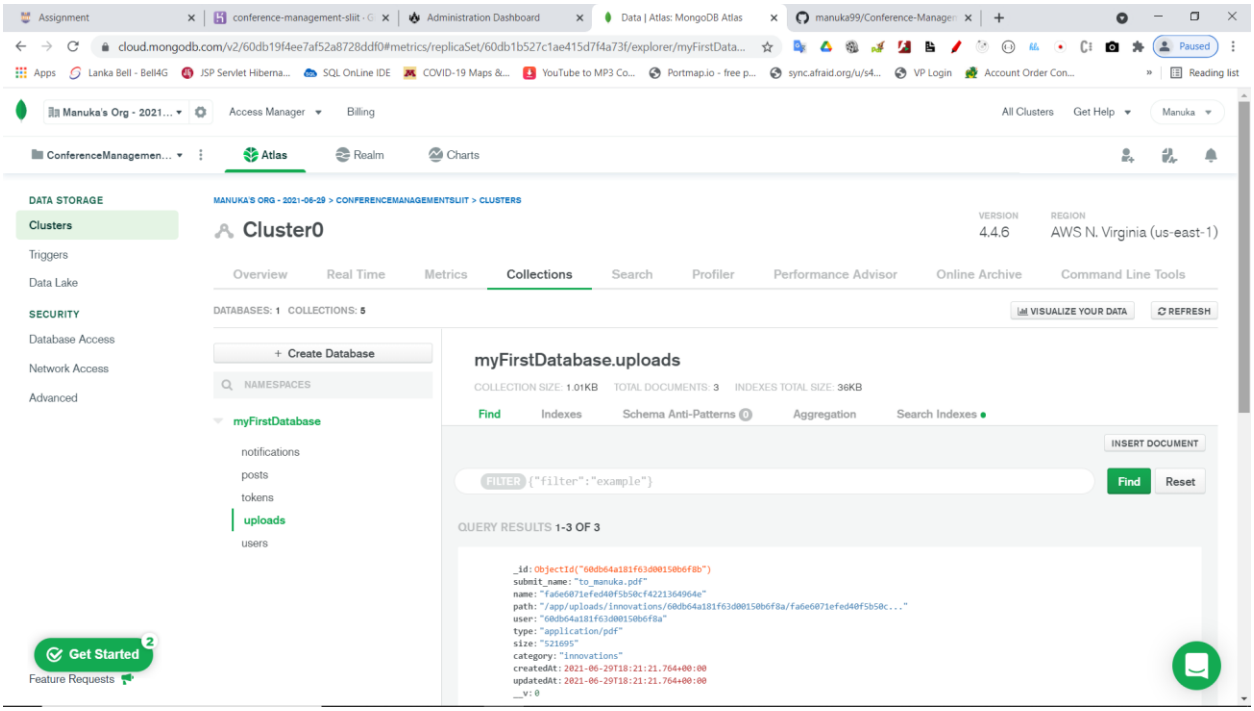
2.1.4 Use Case Diagram



2.1.5 React Component Diagram

1. APP
    1. Homepage
        1. Header
        2. Footer
        3. Navbar
        3. Body
            1. Landing Page
                1. Hero Component
                2. Workshops Component
                3. Conferences Component
            2. Login Page
            3. Register Page
                1. Contact Form component
                2. Workshop Form component
                3. Research Form component
                4. Innovation Form component
                5. Payment Component
            3. Recover Password Page
            4. Reset Password Page
                1. Reset Password Form
            4. Manage Profile Page
                1. Contact Details Form
                2. Change Password Form
                3. Change Email Form
            5. My Innovation Page
            6. My Workshop Page
            7. My Research Page
            7. My Attendance Page
    2. Admin Dashboard
        1. Admin Header
        2. Footer
        3. Navbar
        4. Body
            1. Admin Homepage
                1. New Users component
                2. New Workshops Component
                3. New Conferences Component
                4. New Innovation Component
                5. New Attendees Component
            2. Manage Innovation Page
                1. Innovation Form component
                2. View Innovation Component
                3. View All Innovations Component

2.1.6 Mongo database document screenshot



User

```
_id: ObjectId("60db64a181f63d00150b6f8a")
isApproved: false
approvalReason: "false"
role: "ADMIN"
__t: "admin"
firstName: "Bell43434"
lastName: "Heathcote"
phone: "0721146092"
email: "manukayasas99@gmail.com"
date_of_birth: 2021-07-01T00:00:00.000+00:00
address: "2485 Franecki Trafficway Apt. 023"
payment: "sdsdsd"
password: "$2b$12$cDkh00gQV1NVIEc0Kd2QuemwzJLzQjm1RDnn6aEKwgaqild7Q3TYC"
createdAt: 2021-06-29T18:21:21.356+00:00
updatedAt: 2021-06-29T18:21:21.798+00:00
__v: 0
file: ObjectId("60db64a181f63d00150b6f8b")
```

```
_id: ObjectId("60db67f372a4dd0015c954ef")
isApproved: false
```

Token


```
_id: ObjectId("60db681c72a4dd0015c954f1")
user_id: ObjectId("60db67f372a4dd0015c954ef")
token: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2MjQ5OTUzMzIsImRhdGEiO..."
ip_address: "113.59.214.13"
user_agent: "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, l..."
deviceType: "desktop"
deviceInfo: "Microsoft Windows"
osInfo: "Windows 10.0"
browser: "Chrome"
version: "91.0.4472.124"
payload: "Microsoft Windows"
last_activity: 1624992773
isValid: true
createdAt: 2021-06-29T18:36:12.732+00:00
updatedAt: 2021-06-29T18:52:53.332+00:00
__v: 0
```


Uploads


```
_id: ObjectId("60db64a181f63d00150b6f8b")
submit_name: "to_manuka.pdf"
name: "fa6e6071efed40f5b50cf4221364964e"
path: "/app/uploads/innovations/60db64a181f63d00150b6f8a/fa6e6071efed40f5b50c..."
user: "60db64a181f63d00150b6f8a"
type: "application/pdf"
size: "521695"
category: "innovations"
createdAt: 2021-06-29T18:21:21.764+00:00
updatedAt: 2021-06-29T18:21:21.764+00:00
__v: 0
```

Notification

```
_id: ObjectId("60dc677bdd5d5f204c90131f")
isSeen: false
url: "/protected/users/60dc6667f3c7c13648f3c03f"
message: "User with role ADMIN and email manukayasas99@gmail.com has set approva..."
from: ObjectId("60da06f9f86c9c3488bb45fc")
to: ObjectId("60da06f9f86c9c3488bb45fc")
__v: 0
```

```
_id: ObjectId("60dc677bdd5d5f204c901320")
isSeen: false
url: "/protected/users/60dc6667f3c7c13648f3c03f"
message: "User with role ADMIN and email manukayasas99@gmail.com has set approva..."
from: ObjectId("60da06f9f86c9c3488bb45fc")
to: ObjectId("60da0abef86c9c3488bb4603")
__v: 0
```

```
_id: ObjectId("60dc677bdd5d5f204c901321")
```

Posts

```
_id: ObjectId("60dc6d860c5c2a41d83e1cfe")
title: "sdsdssasasasasasasasasasasasasasasasad"
description: "asaasasas asaasasasasasasasasasasasasasa"
user: ObjectId("60da06f9f86c9c3488bb45fc")
__v: 0
isApproved: true
```

```
_id: ObjectId("60dc6e628c387b42e87ac1b1")
isApproved: false
title: "sdsdssasasasasaq      "
description: "asaasasas asaasasasasasasasasasasasasasasa"
user: ObjectId("60da06f9f86c9c3488bb45fc")
__v: 0
```

2.2 IT19152288 - Workshop management and Conference Management

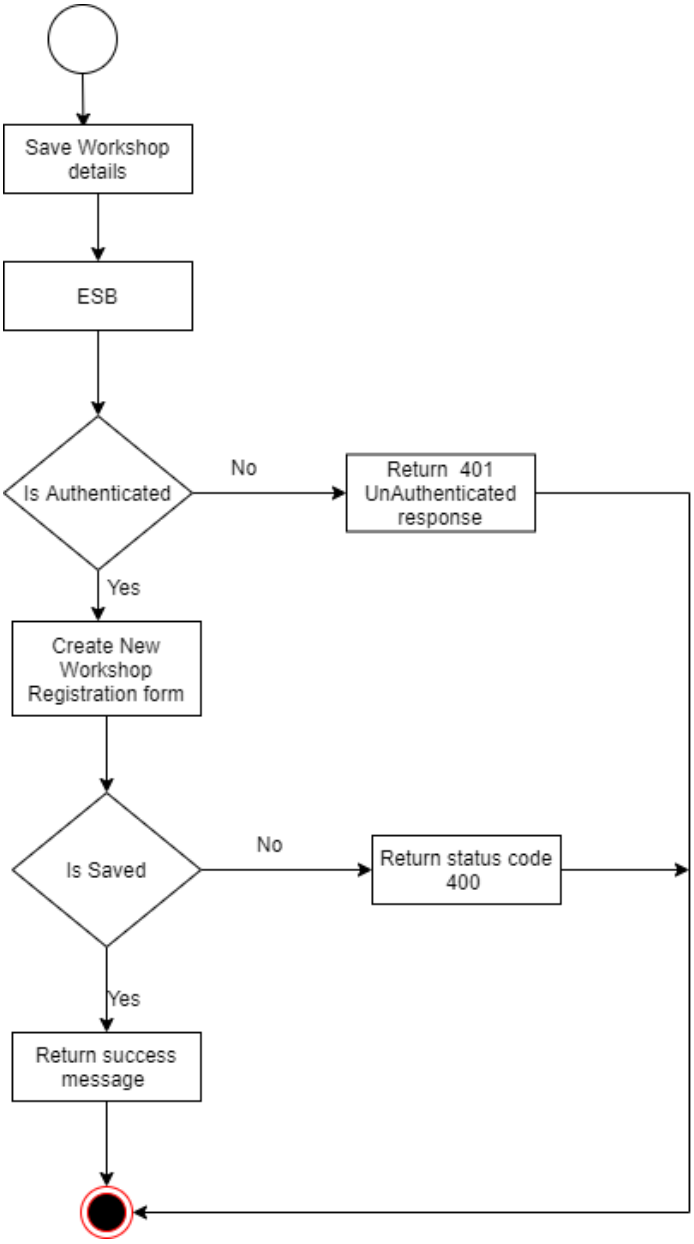
2.2.1 Description

Users can view conference/workshop details without login to the system. But only registered user can request for conduct workshop or conference. For registration, user can add their details to the given form and submit it. For registration for conduct workshop also user need to submit given form.

2.2.2 Rest API

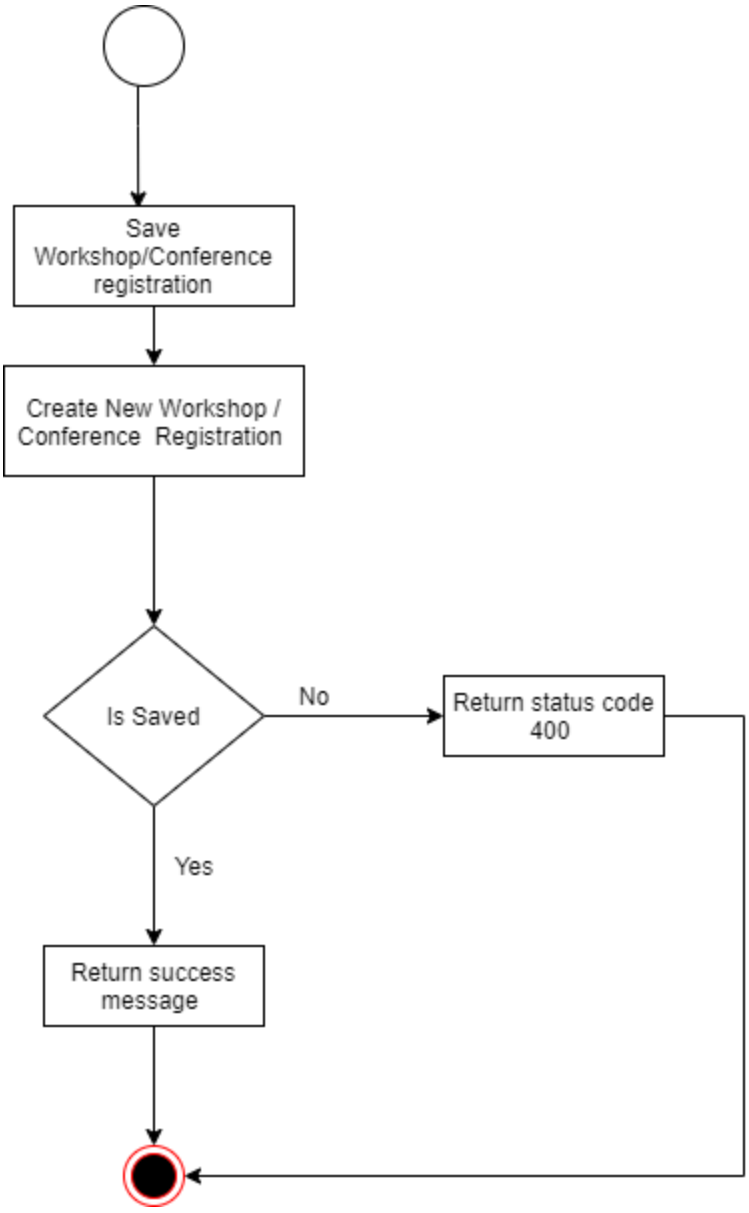
Submit registration form for conduct workshop/conference

A POST request with the relevant details will be sent. User will be authenticated to check whether the user id is valid. If there is an error occurred during authentication process the error response will be sent back. If the authenticated process success the provided data in the request body will be validated. If there is no error occurred during validation, a new workshop will be created and saved. If error occurred during saving the error response is returned. Else successful message will be sent back.



Submit registration form for participate workshop/conference

A POST request with the relevant details will be sent. The provided data in the request body will be validated. If there is no error occurred during validation, a new workshop will be created and saved. If error occurred during saving the error response is returned. Else successful message will be sent back.



2.2.3 Test cases

Test Case 01		
Author	User	
Summary	User register for Conference / Workshop details	
Precondition	User need to be authenticated.	
Step No.	**Step Action**	**Expected output**
1	User visit to the homepage.	Navigate to the homepage.
2	User click on “Workshop” button.	Navigate to the workshop page.
3	Click on “Register for workshop” button.	Navigate to the registration form.
4	User fill email.	Text field should accept input data.
5	User fill Name.	Details were filled in the text field.
6	User fill Affiliation	Details were filled in the text field.
7	User fill mobile number.	Text field should accept input data.
8	User fill prefeed workshop.	User select the preferred workshop.
9	User fill statement of interest.	Details were filled in the text field.
10	User click on “Register” button.	Application alerts, “Your registration details were successfully saved” and navigates to the home page

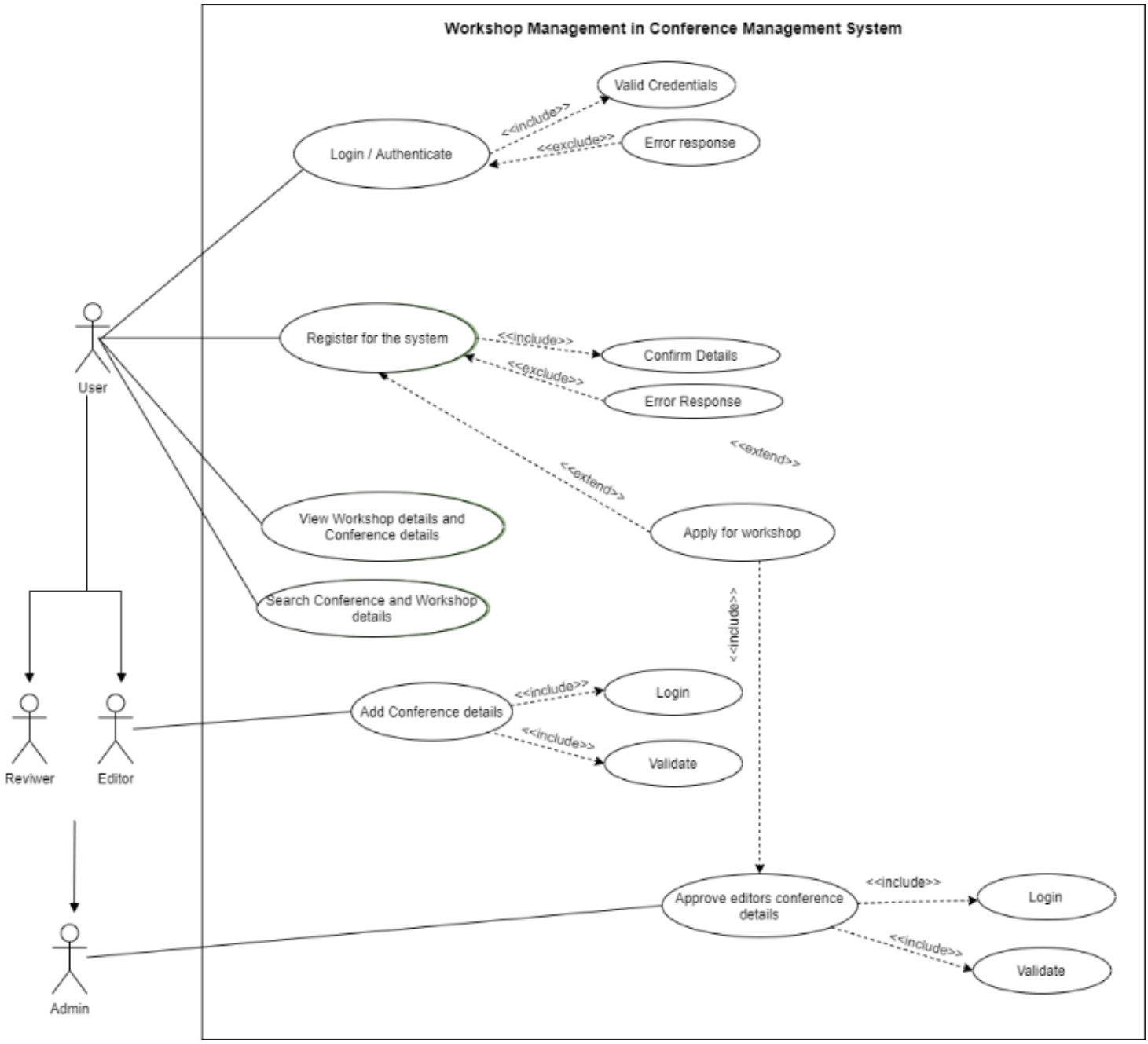
Test Case 02		
Author	User	
Summary	User register for conduct Conference / Workshop.	
Precondition	User need to be authenticated.	
Step No.	**Step Action**	**Expected output**
1	User visit to the homepage.	Navigate to the homepage.
2	User click on “Workshop” button.	Navigate to the workshop page.
3	Click on “Register for conduct workshop” button.	Navigate to the registration form.
4	User fill title of the workshop.	Details were filled in the text field.
5	User fill organizers name.	Details were filled in the text field.
6	User fill organizers Affiliation	Details were filled in the text field.
7	User fill organizers email.	Text field should accept input data.
8	User fill Scope and topic of workshop.	Details were filled in the text field.
9	User fill potential participant.	Details were filled in the text field.
10	User fill planned duration.	Details were filled in the text field.
11	User fill preferred day of workshop.	Details were filled in the text field.
12	User fill name of the referred papers.	Details were filled in the text field.
13	User click on “Register” button.	Application alerts, “Your registration details were successfully saved” and navigates to the home page

Test Case 03		
Author	Editor	
Summary	Add new workshop /conference to the system.	
Precondition	User need to be authenticated.	
Step No.	**Step Action**	**Expected output**
1	User visit to the homepage.	Navigate to the homepage.
2	Click on admin dashboard.	Navigate to the admin dashboard.
3	User click on “Workshop” button.	Navigate to the workshop page.
4	Click on “Add new workshop” button.	Navigate to the add workshop page.
5	User fill title of the workshop.	Details were filled in the text field.
6	User fill organizers name.	Details were filled in the text field.
7	User fill organizers Affiliation	Details were filled in the text field.
8	User fill Scope and topic of workshop.	Details were filled in the text field.
9	User fill potential participant.	Details were filled in the text field.
10	User fill planned duration.	Details were filled in the text field.
11	User fill preferred day of workshop.	Details were filled in the text field.
12	User click on “Add” button.	Application alerts, “Your details were added successfully”.

Test Case 04		
Author	Editor	
Summary	Edit existing workshop /conference in the system.	
Precondition	User need to be authenticated.	
Step No.	**Step Action**	**Expected output**
1	User visit to the homepage.	Navigate to the homepage.
2	Click on admin dashboard.	Navigate to the admin dashboard.
3	User click on “Workshop” button.	Navigate to the workshop page.
4	Click on “Edit workshop” button.	Navigate to the edit workshop page.
6	User edit organizers name.	Details were filled in the text field.
7	User edit organizers Affiliation	Details were filled in the text field.
9	User edit potential participant.	Details were filled in the text field.
10	User edit planned duration.	Details were filled in the text field.
11	User edit preferred day of workshop.	Details were filled in the text field.
12	User click on “Save” button.	Application alerts, “Your details were saved successfully”.

Test Case 05		
Author	Admin	
Summary	Accept the request for conducting workshop.	
Precondition	User need to be authenticated.	
Step No.	**Step Action**	**Expected output**
1	User visit to the homepage.	Navigate to the homepage.
2	Click on admin dashboard.	Navigate to the admin dashboard.
3	User click on “Workshop” button.	Navigate to the workshop page.
4	User Click on “Request for conduct workshop” button.	Navigate to the request page.
5	User click on “Accept” button.	Application alerts, “Your details were saved successfully”.

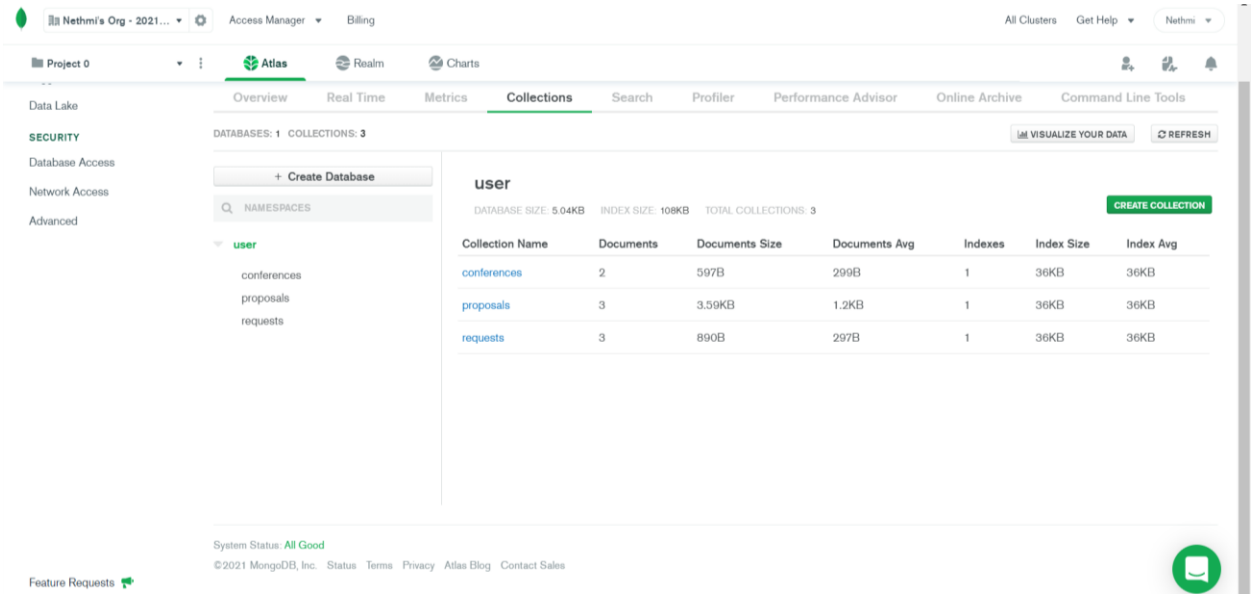
2.2.4 Use Case Diagram

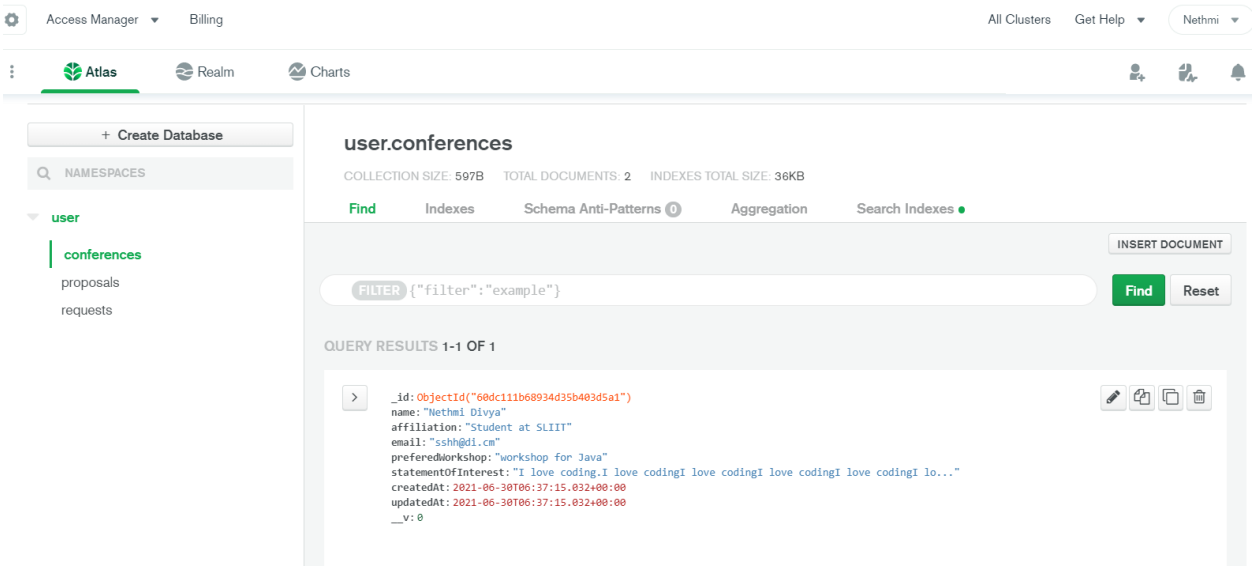


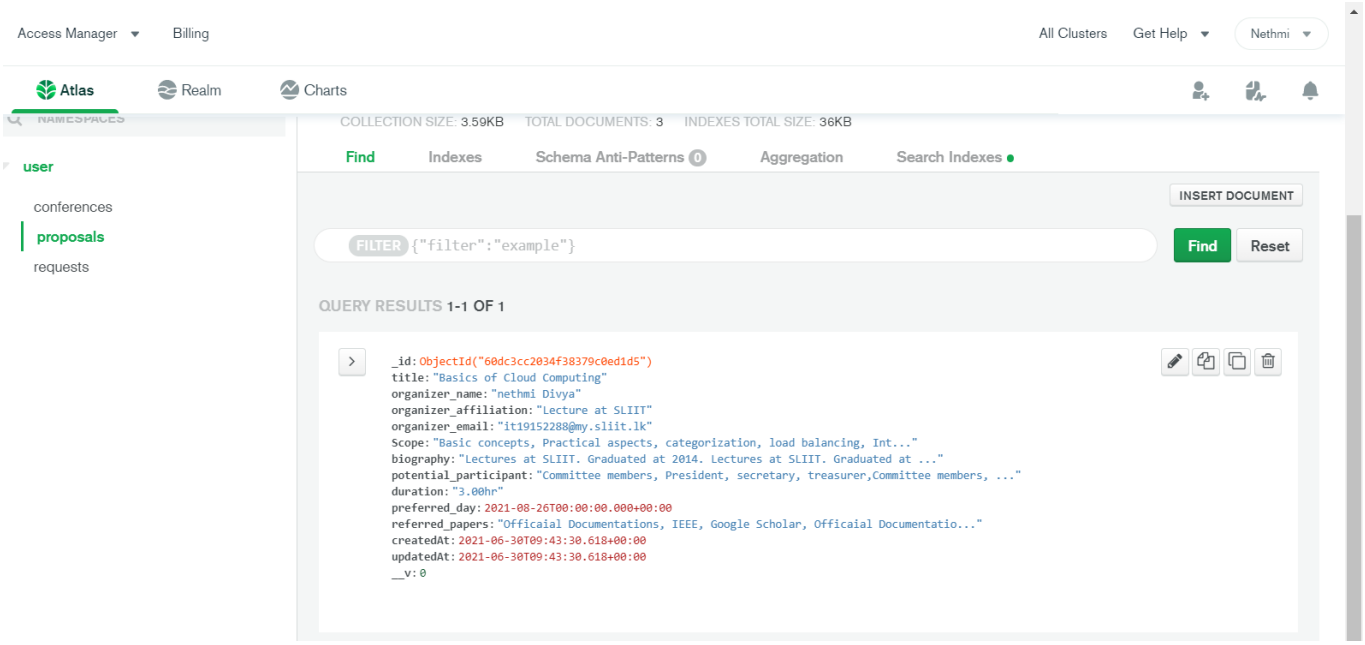
2.2.5 React Component Diagram

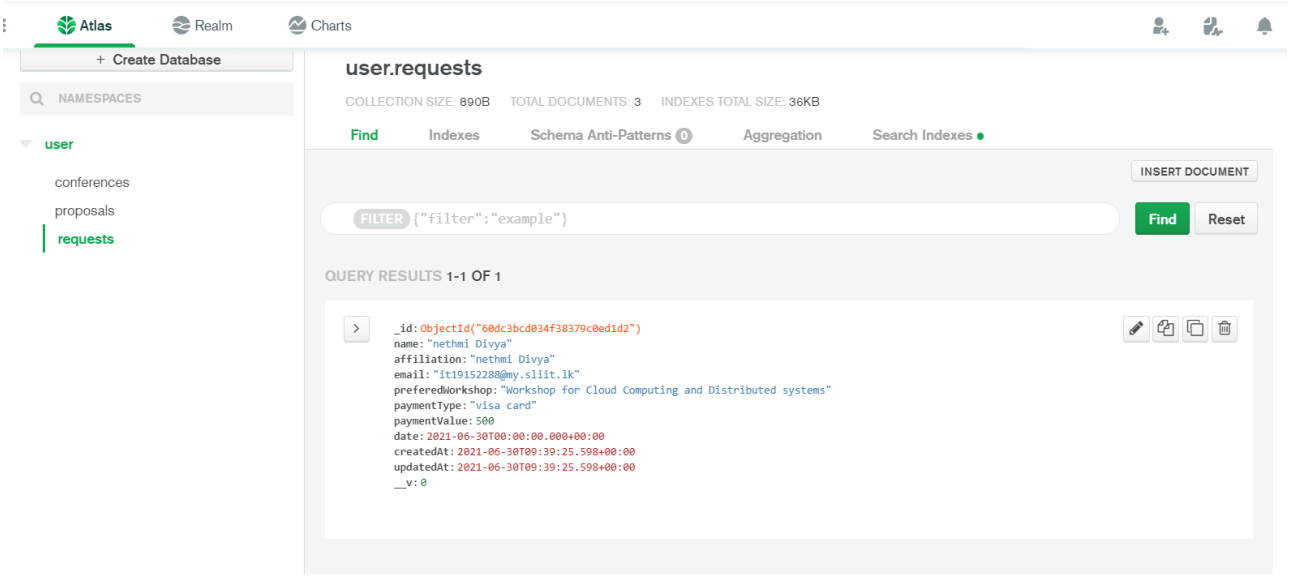
1. App
    - My Workshop Page
        - Search workshop
        - Register for workshop form
        - Register for conduct workshop form
2. Admin Dashboard
    - Workshop component
        - Add new workshop
        - Edit workshop
        - Delete workshop
        - Accept request for conduct workshop

2.2.6 Mongo database document screenshot






Access Manager
Billing
All Clusters
Get Help
Nethmi
Atlas
Realm
Charts
NAMESPACES
user
conferences
proposals
requests
COLLECTION SIZE: 3.59KB
TOTAL DOCUMENTS: 3
INDEXES TOTAL SIZE: 36KB
Find
Indexes
Schema Anti-Patterns 0
Aggregation
Search Indexes
INSERT DOCUMENT
FILTER {"filter":"example"}
Find
Reset
QUERY RESULTS 1-1 OF 1
_id: ObjectId("60dc3cc2034f38379c0ed1d5")
title: "Basics of Cloud Computing"
organizer_name: "nethmi Divya"
organizer_affiliation: "Lecture at SLIIT"
organizer_email: "it19152288@my.sliit.lk"
Scope: "Basic concepts, Practical aspects, categorization, load balancing, Int..."
biography: "Lectures at SLIIT. Graduated at 2014. Lectures at SLIIT. Graduated at ..."
potential_participant: "Committee members, President, secretary, treasurer,Committee members, ..."
duration: "3.00hr"
preferred_day: 2021-08-26T00:00:00.000+00:00
referred_papers: "Officaial Documentations, IEEE, Google Scholar, Officaial Documentatio..."
createdAt: 2021-06-30T09:43:30.618+00:00
updatedAt: 2021-06-30T09:43:30.618+00:00
__v: 0


Atlas
Realm
Charts
+ Create Database
NAMESPACES
user
conferences
proposals
requests
user.requests
COLLECTION SIZE: 890B
TOTAL DOCUMENTS: 3
INDEXES TOTAL SIZE: 36KB
Find
Indexes
Schema Anti-Patterns 0
Aggregation
Search Indexes
INSERT DOCUMENT
FILTER {"filter":"example"}
Find
Reset
QUERY RESULTS 1-1 OF 1
_id: ObjectId("60dc3bcd034f38379c0ed1d2")
name: "nethmi Divya"
affiliation: "nethmi Divya"
email: "it19152288@my.sliit.lk"
preferedWorkshop: "Workshop for Cloud Computing and Distributed systems"
paymentType: "visa card"
paymentValue: 500
date: 2021-06-30T00:00:00.000+00:00
createdAt: 2021-06-30T09:39:25.598+00:00
updatedAt: 2021-06-30T09:39:25.598+00:00
__v: 0

2.3 IT19215884 - Research paper management

2.3.1 Description

Authors can submit, update, delete and see the details of a submitted papers under paper management. Not only that user can download templates from the site. Users can also create their own templates and upload them to the site. Furthermore, they can update and delete those templates.

When submitting a paper, the details of the paper such as title, subject of the paper, names of the authors, type of the paper, number of the pages in the paper and submitting date will be taking as the inputs. Date is not compulsory to fill, if that field is empty, the default value will be set to the current date and time. The content of the paper should be uploaded to the site in pdf format. The templates also can be uploaded to the site and if in need of uploading an updated version, user can delete the previous and upload the new one.

Dependencies

The functionalities of paper management are developed as REST API by using MERN stack. Java Script framework, React.js will be used for the frontend development, express.js and node.js for the server tier and mongoDB for the database tier. When a request is sent , the routes will forward the request to the controller and the controller will get the data from the DB through the model and display that data on a browser as the response.

2.3.2 Rest API

Submit Paper

A POST request with the above-mentioned details will be sent. The user should be an authenticated user, so that user id will be gone through an authentication checkup. If there is an error, an error message will be returned. If the user is an authenticated user, the provided data in the request body will be validated. If there is no error occurred in validation, a new paper will be created and saved in the DB. If an error occurred, the user will be informed by an error message. After the new paper saved in the DB successfully, a success message will be returned. Otherwise, an error message will be returned.

Get all papers

A GET request will be sent through the authentication service. After authentication, the validation check will be taking placed and if it is successfully validated, all the papers submitted by the user in the DB will be fetched. In case if an error occurred, error message will be returned.

Fetch one paper

A GET request with the specified id will be sent to the authentication service. After authentication, the paper id will be validated and if there is no error, the paper corresponding to the id will be fetched. In case if an error occurred, error message will be returned.

Update papers

A PATCH request with the paper id and updated details will be sent to the authentication service. After the authentication, the request will be validated. Then the availability of the paper object is checked and if available, paper will be updated and saved in the DB. If there is error in the process error message will be returned.

Delete papers

A DELETE request with the paper id will be sent to the authentication service. After the authentication, the request will be validated. Then the availability of the paper object is checked and if available the delivery will be deleted. If there is error in the process error message will be returned.

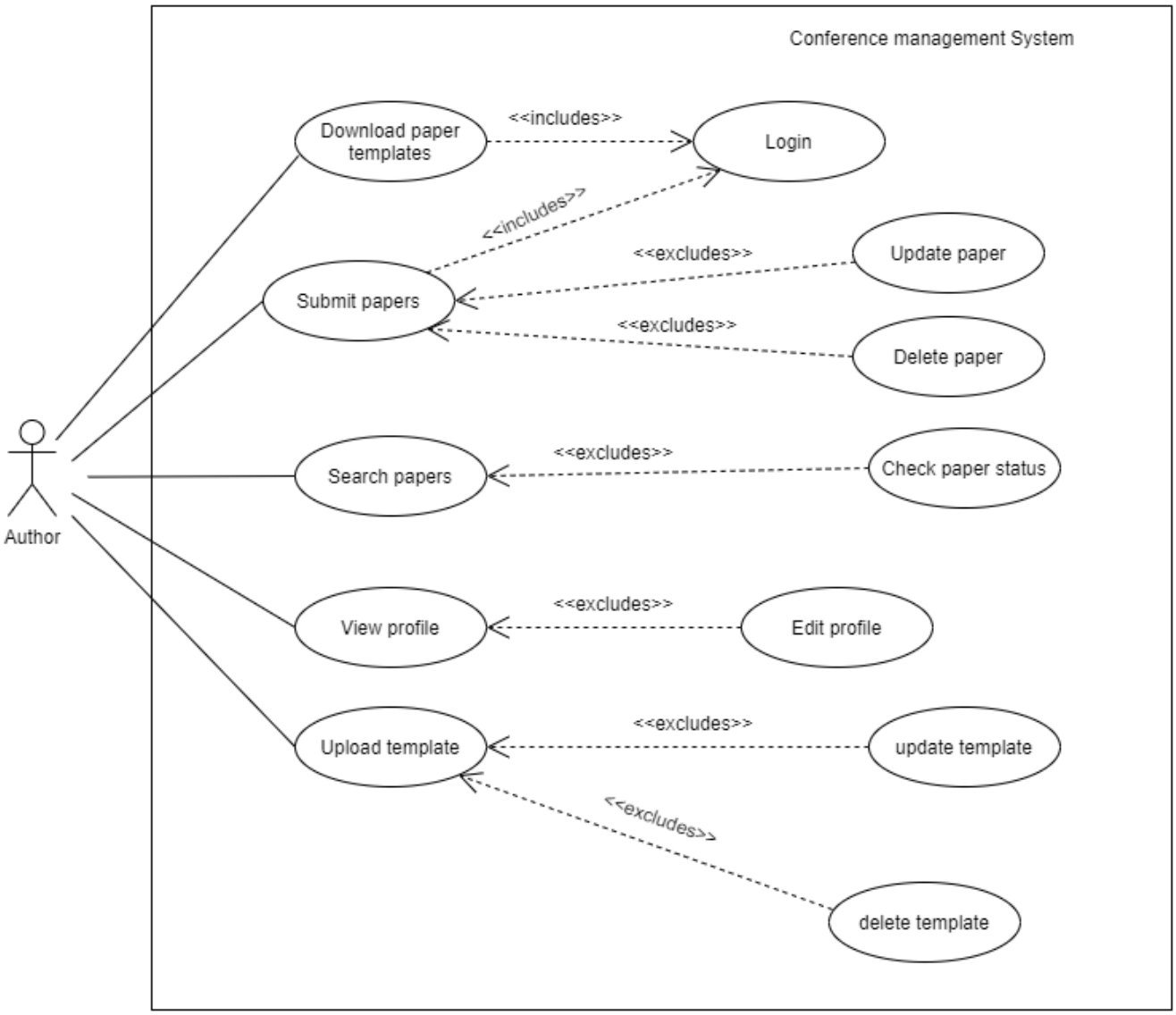
2.3.3 Test cases

1. Summary This test case checks whether the paper templates are downloaded in word format.		
Preconditions The user should be an authenticated user.		
Step number	**Step action**	**Expected result**
1	User clicks on "Explore templates"	User should successfully navigate to templates
2	User selects a template and clicks download button	The templated downloads in word format.

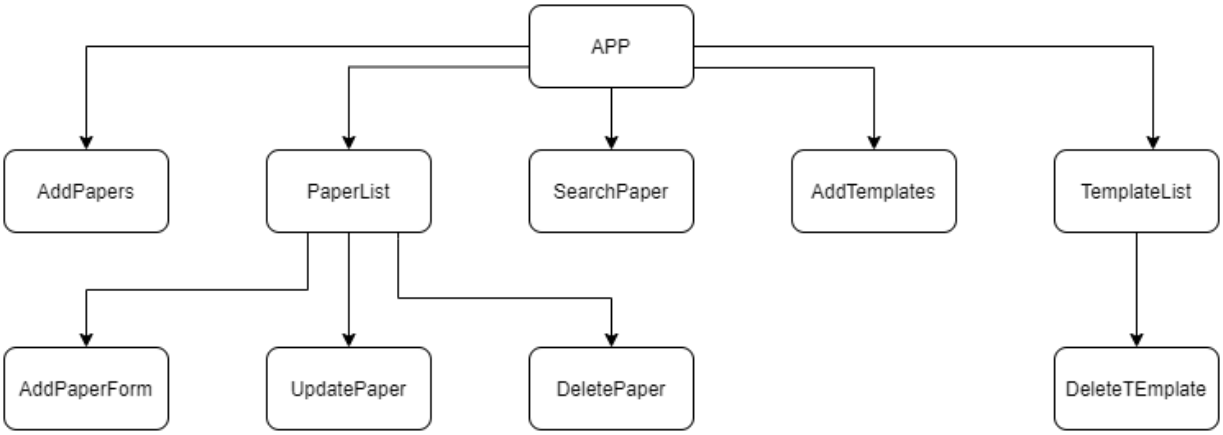
2. Summary This test case checks whether the user fills all the fields of the form when submitting a paper.		
Preconditions The user should be an authenticated user.		
Step number	**Step action**	**Expected result**
1	User clicks on "submit paper" button in the dashboard	User should successfully navigate to submit paper page
2	User enters title of the paper.	The text field should accept the entered data.
3	User enters subject of the paper.	The text field should accept the entered data.
4	User enters authors of the paper.	The text field should accept the entered data.
5	User selects the type of the paper.	The text field should accept the entered data.
6	User enters no: of pages of the paper.	The text field should accept the entered data.
7	User uploads the content of the paper in pdf format.	The text field should accept the entered data.
8	User clicks on "save" button	A paper is submitted successfully, and details saved in the DB.

3. Summary This test case checks whether the number of pages field does not accept letters		
Preconditions The user should be an authenticated user.		
Step number	**Step action**	**Expected result**
1	User clicks on "submit paper" button in the dashboard	User should successfully navigate to submit paper page
2	User enters characters to no: of pages field.	Error message should be prompt.

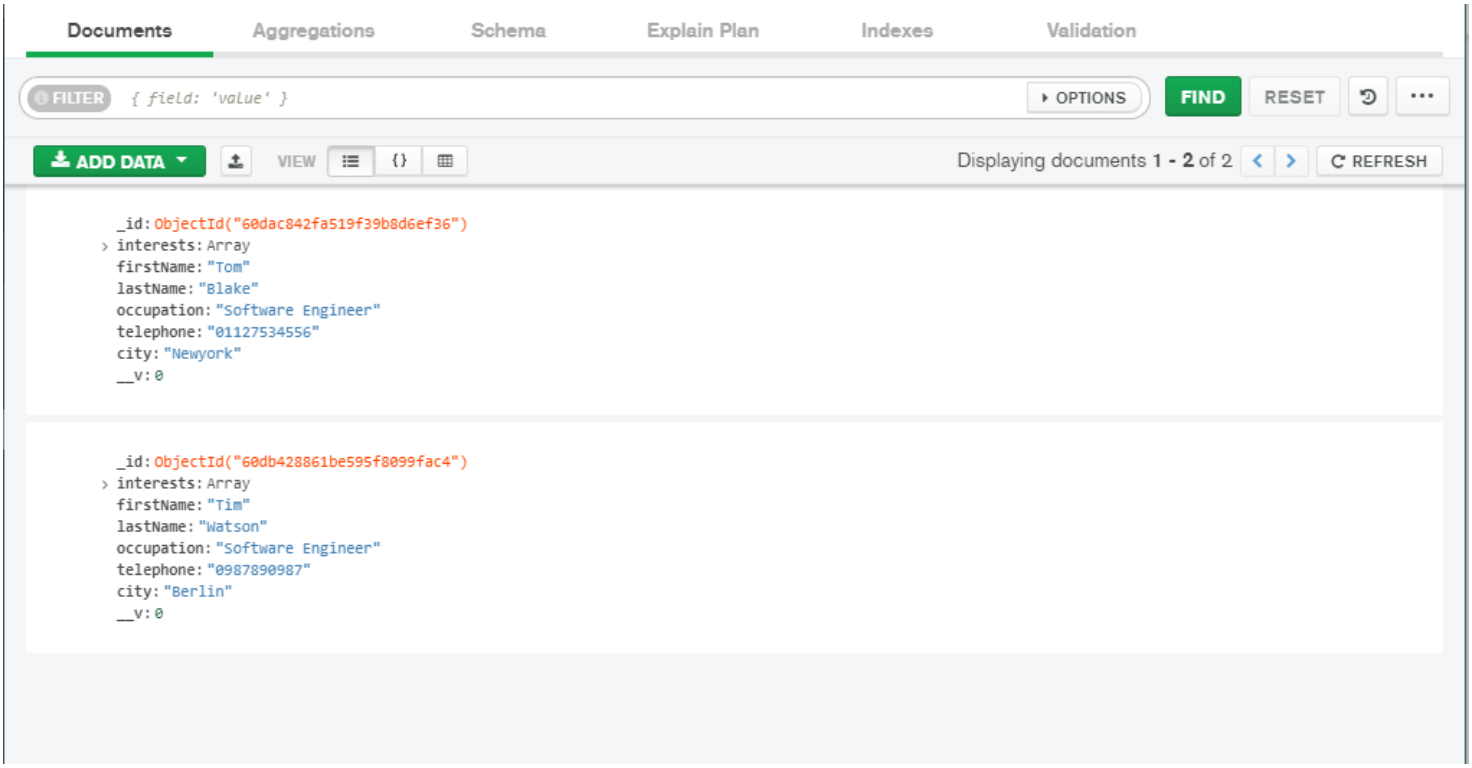
2.3.4 Use Case Diagram

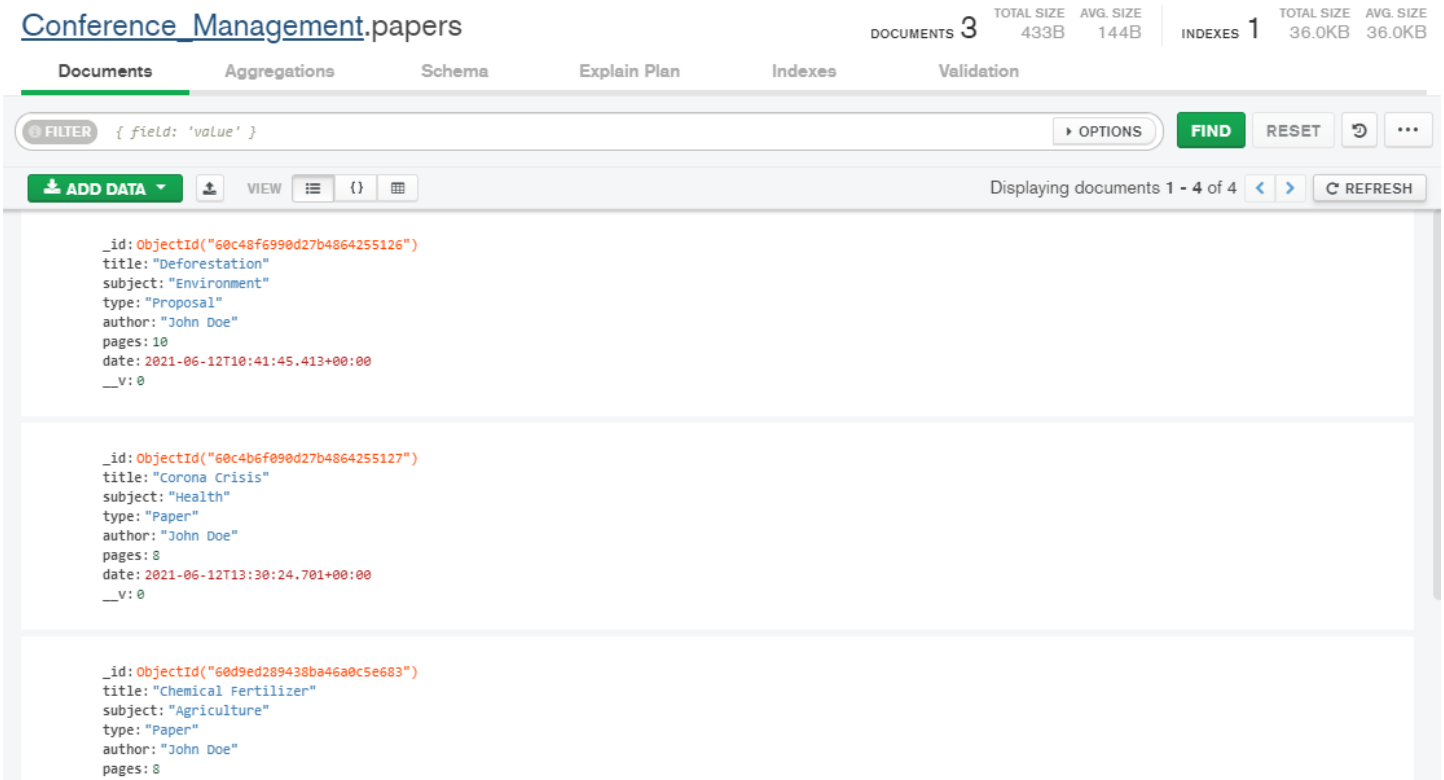


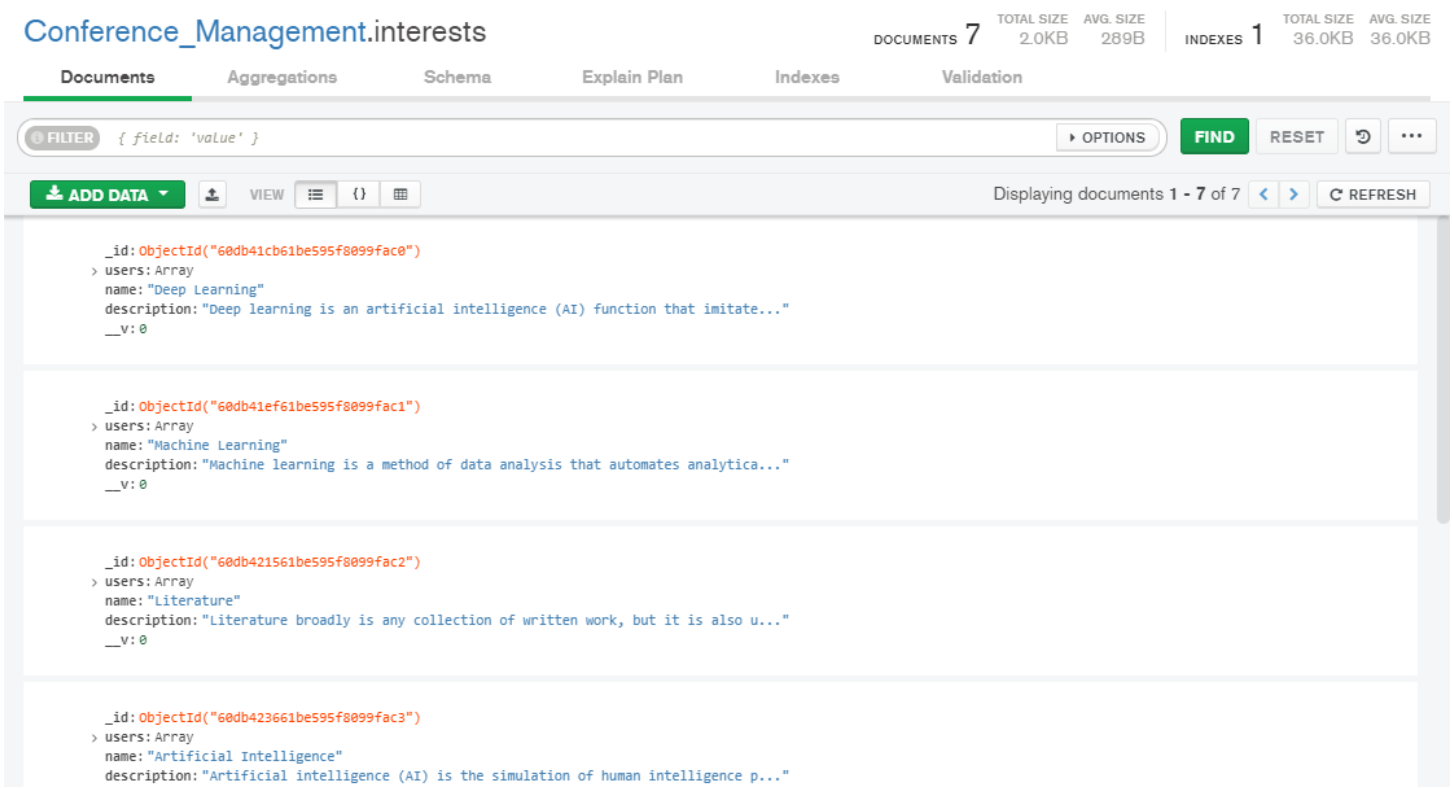
2.3.5 React Component Diagram

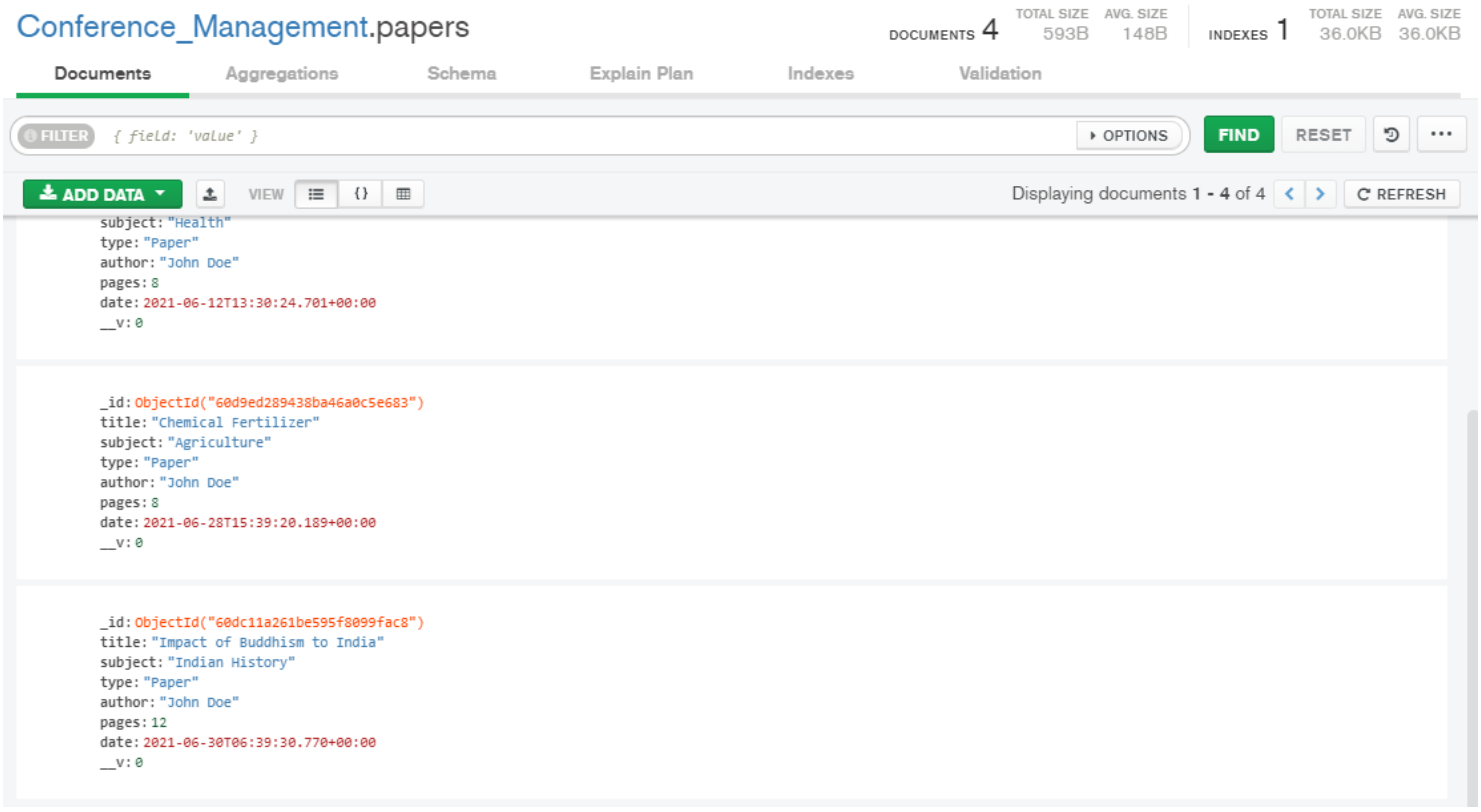


2.3.6 Mongo database document screenshot

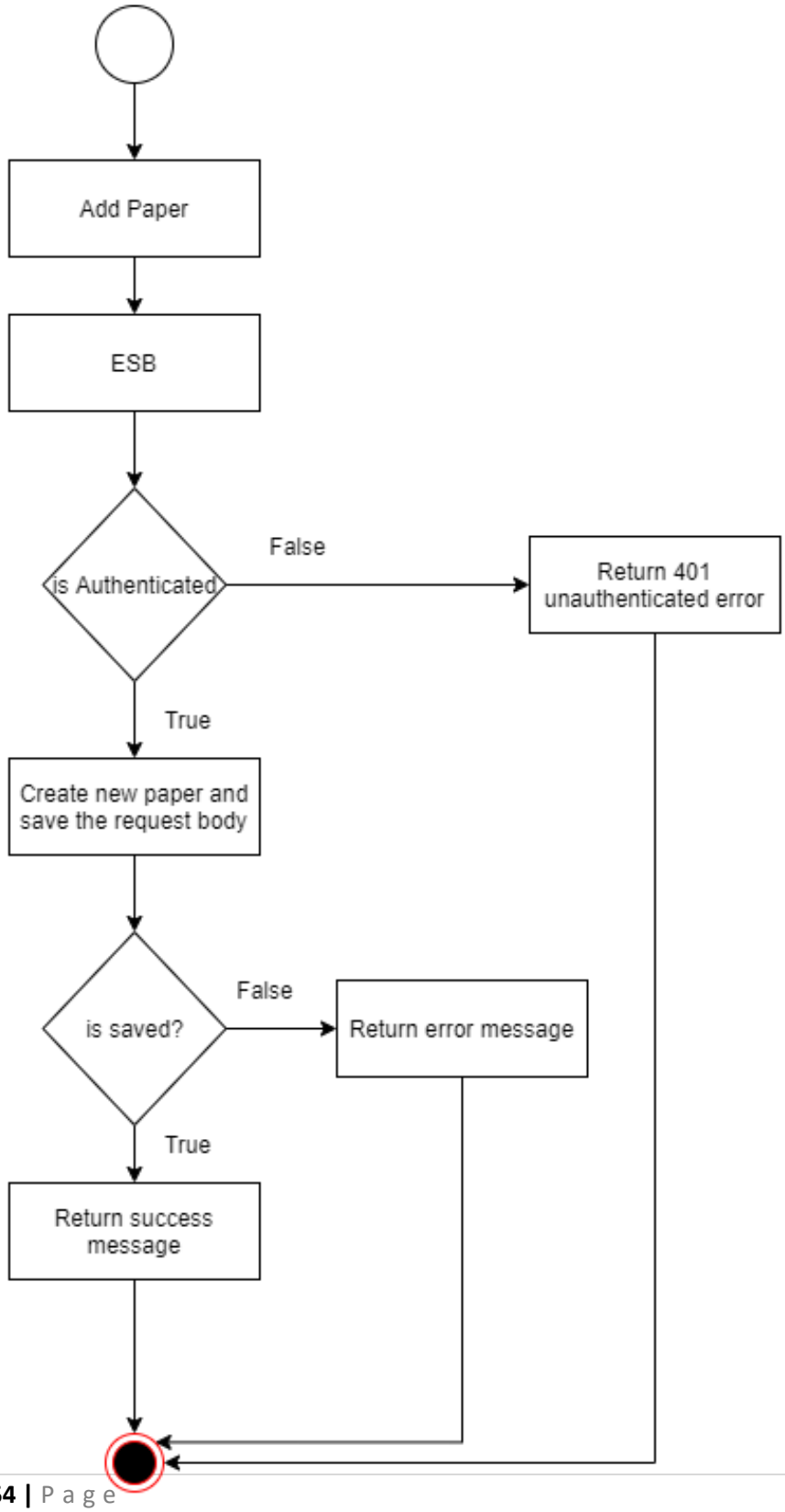


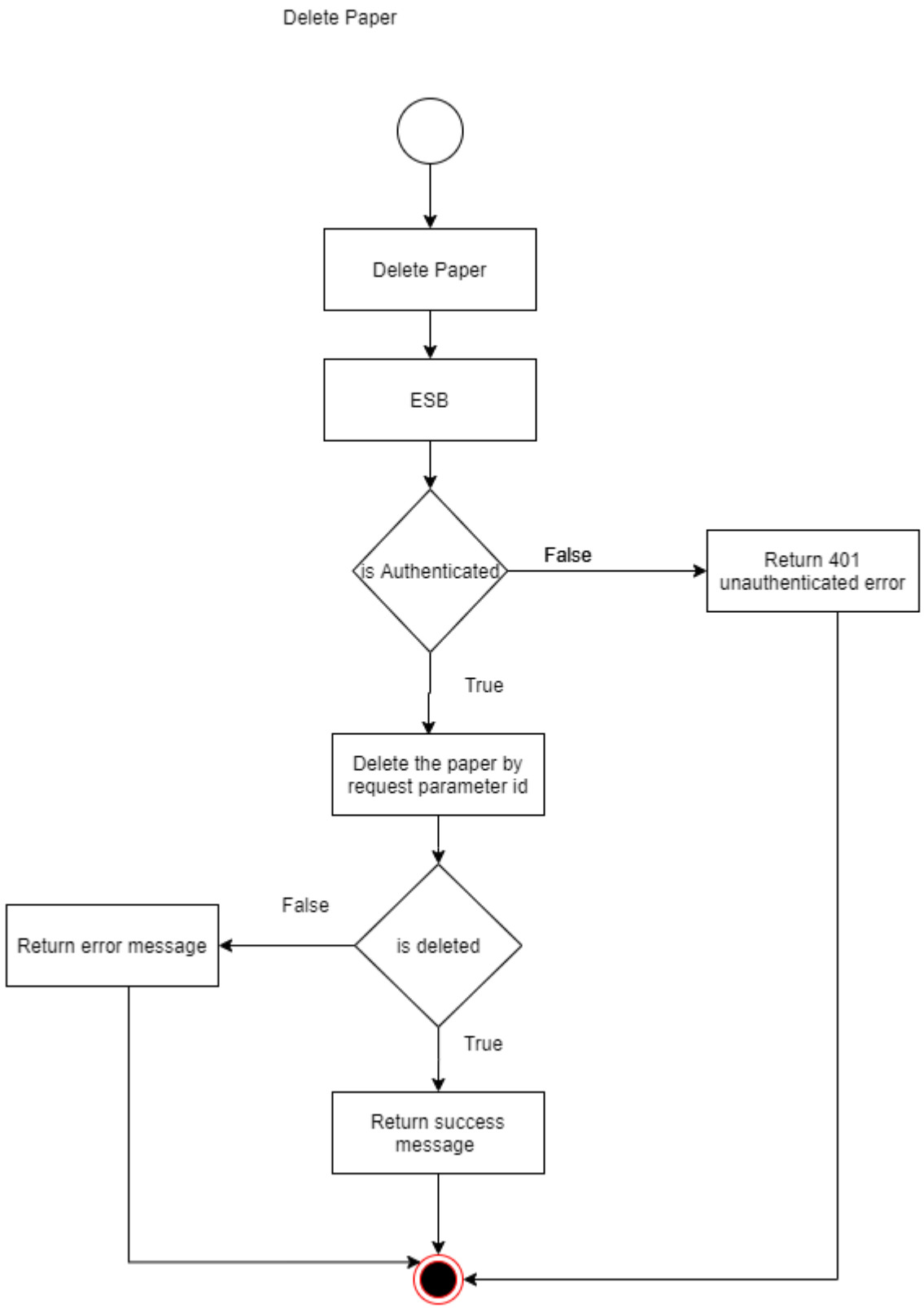


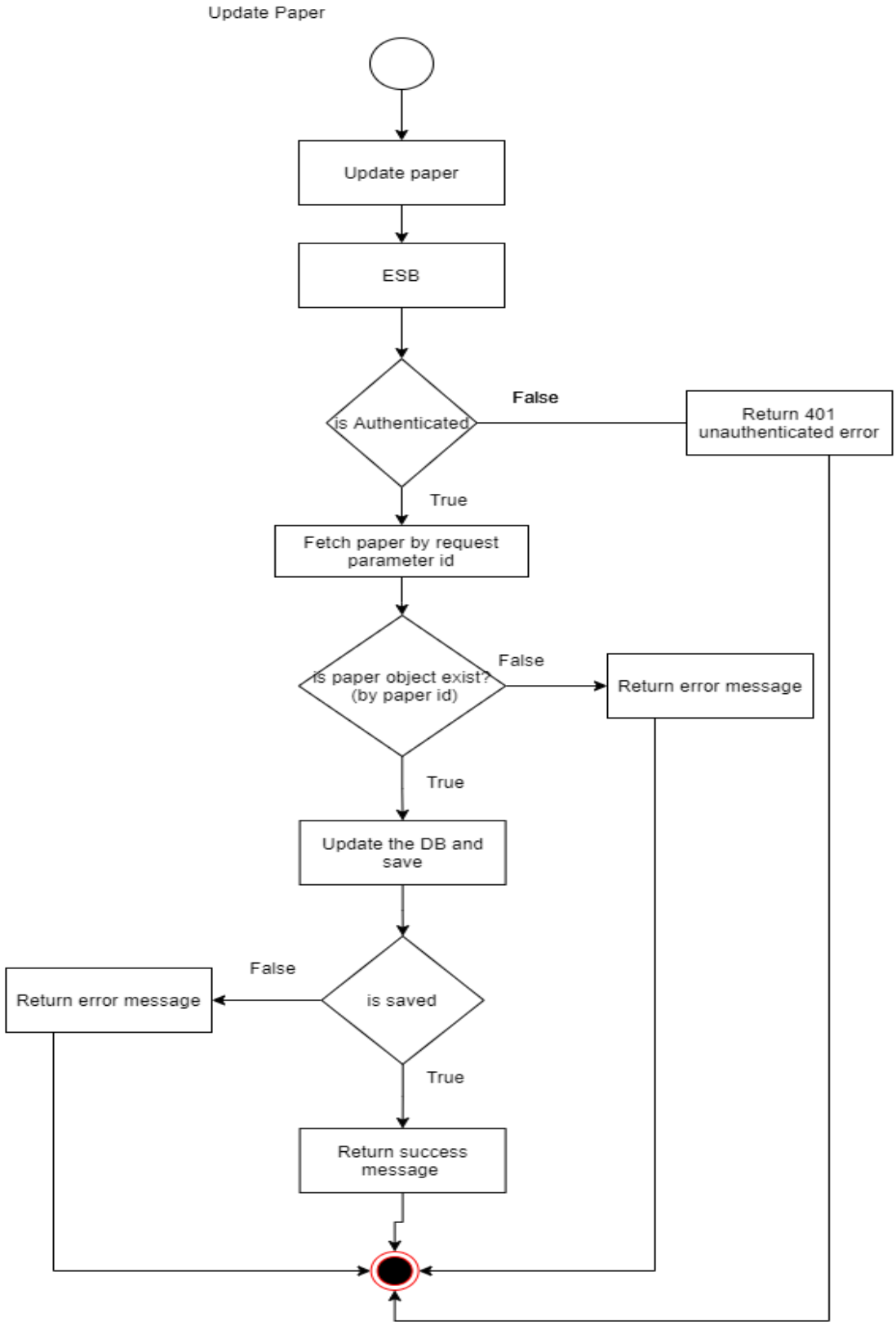

Conference_Management.interests
DOCUMENTS 7 TOTAL SIZE 2.0KB AVG. SIZE 289B
INDEXES 1 TOTAL SIZE 36.0KB AVG. SIZE 36.0KB
Documents Aggregations Schema Explain Plan Indexes Validation
FILTER { field: 'value' } OPTIONS FIND RESET
ADD DATA VIEW Displaying documents 1 - 7 of 7 REFRESH
_id: ObjectId("60db41cb61be595f8099fac0")
users: Array
name: "Deep Learning"
description: "Deep learning is an artificial intelligence (AI) function that imitate..."
__v: 0
_id: ObjectId("60db41ef61be595f8099fac1")
users: Array
name: "Machine Learning"
description: "Machine learning is a method of data analysis that automates analytica..."
__v: 0
_id: ObjectId("60db421561be595f8099fac2")
users: Array
name: "Literature"
description: "Literature broadly is any collection of written work, but it is also u..."
__v: 0
_id: ObjectId("60db423661be595f8099fac3")
users: Array
name: "Artificial Intelligence"
description: "Artificial intelligence (AI) is the simulation of human intelligence p..."


Conference_Management.papers
DOCUMENTS 4 TOTAL SIZE 593B AVG. SIZE 148B
INDEXES 1 TOTAL SIZE 36.0KB AVG. SIZE 36.0KB
Documents Aggregations Schema Explain Plan Indexes Validation
FILTER { field: 'value' } OPTIONS FIND RESET
ADD DATA VIEW Displaying documents 1 - 4 of 4 REFRESH
subject: "Health"
type: "Paper"
author: "John Doe"
pages: 8
date: 2021-06-12T13:30:24.701+00:00
__v: 0
_id: ObjectId("60d9ed289438ba46a0c5e683")
title: "Chemical Fertilizer"
subject: "Agriculture"
type: "Paper"
author: "John Doe"
pages: 8
date: 2021-06-28T15:39:20.189+00:00
__v: 0
_id: ObjectId("60dc11a261be595f8099fac8")
title: "Impact of Buddhism to India"
subject: "Indian History"
type: "Paper"
author: "John Doe"
pages: 12
date: 2021-06-30T06:39:30.770+00:00
__v: 0

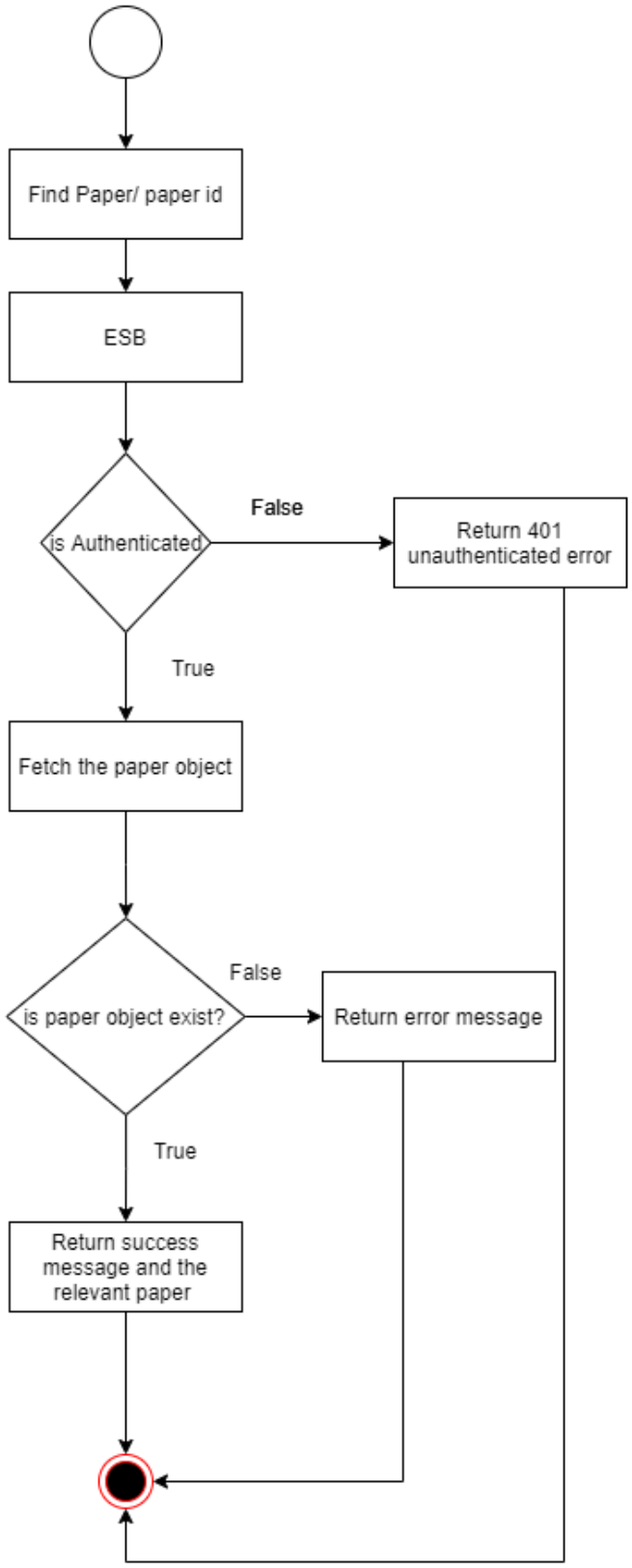
Add Paper


Add Paper
ESB
is Authenticated
False
Return 401 unauthenticated error
True
Create new paper and save the request body
is saved?
False
Return error message
True
Return success message

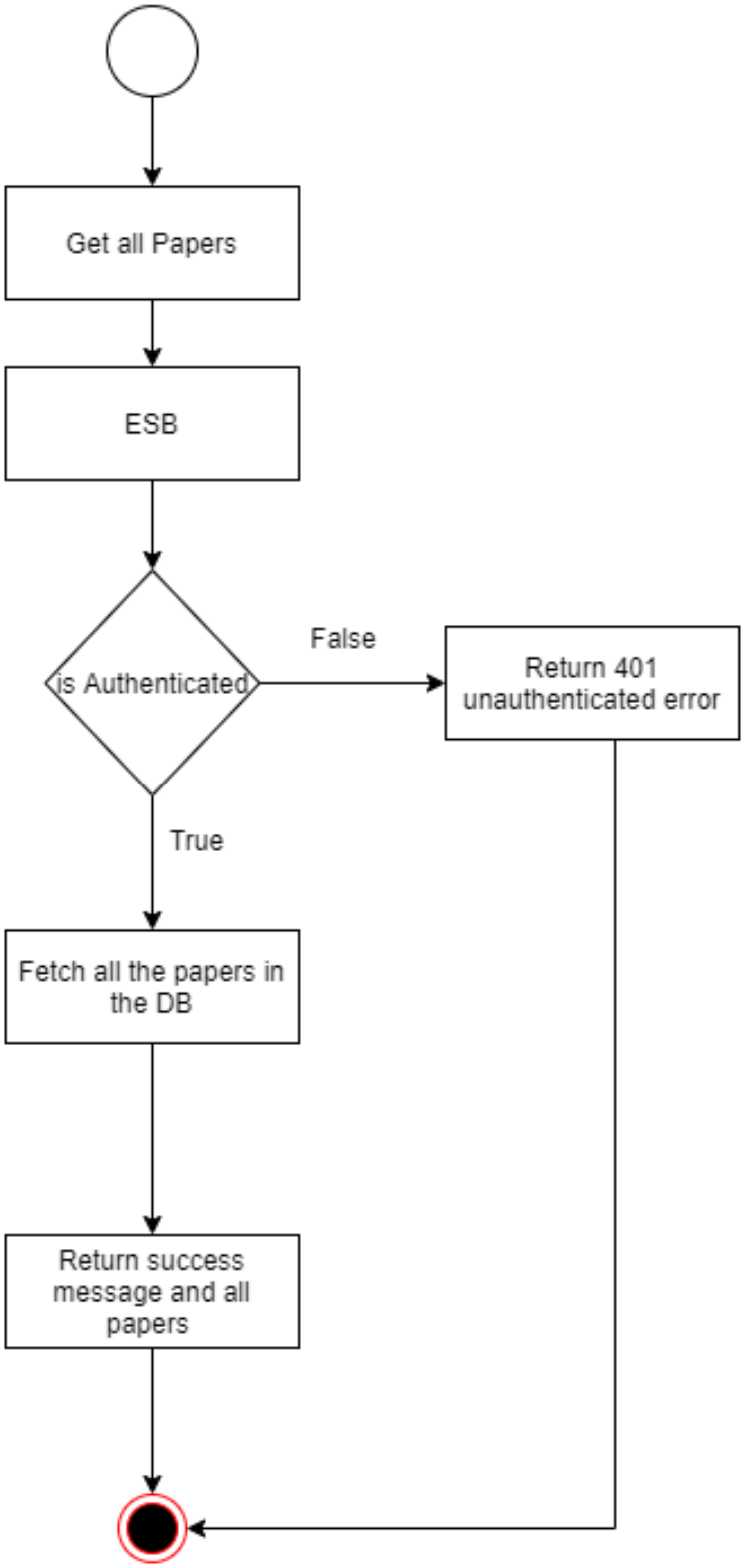

Delete Paper
Delete Paper
ESB
is Authenticated
False
Return 401 unauthenticated error
True
Delete the paper by request parameter id
is deleted
False
Return error message
True
Return success message


Update Paper
Update paper
ESB
is Authenticated
False
Return 401 unauthenticated error
True
Fetch paper by request parameter id
is paper object exist? (by paper id)
False
Return error message
True
Update the DB and save
is saved
False
Return error message
True
Return success message

Get one paper



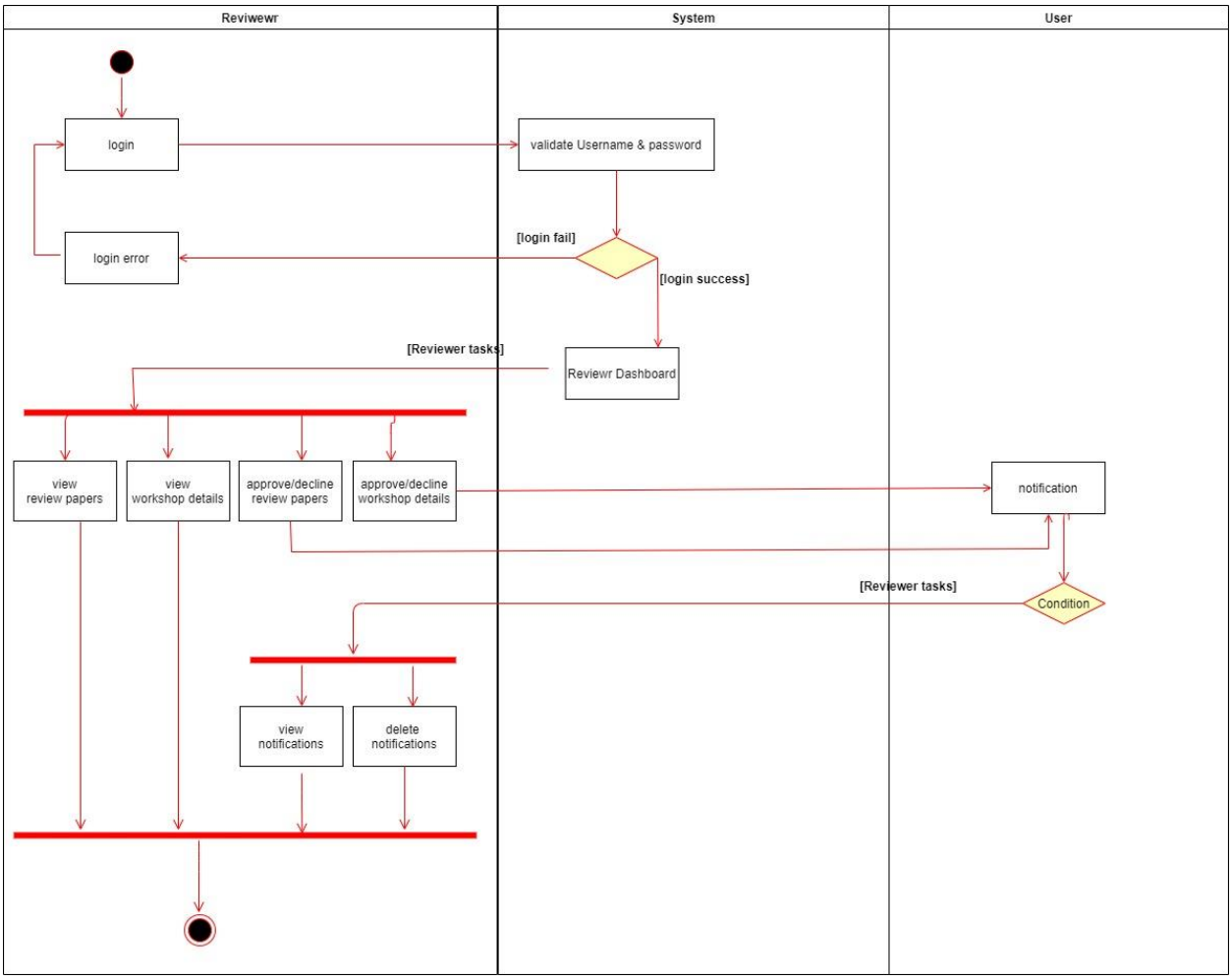
Get all papers



2.4 IT19056258 - Reviewers management

2.4.1 Description

Review can access the system by successfully login. After login reviewer able to view all workshop details and research paper details as list and again reviewer can see all the details and research papers in detail. Reviewers have authority to approve and decline the research papers and workshop details. After approving or decline reviewer will send the notification to user. Finally, reviewer can view list of notification as a list and can delete notifications from the database



Activity diagram for Reviewer function

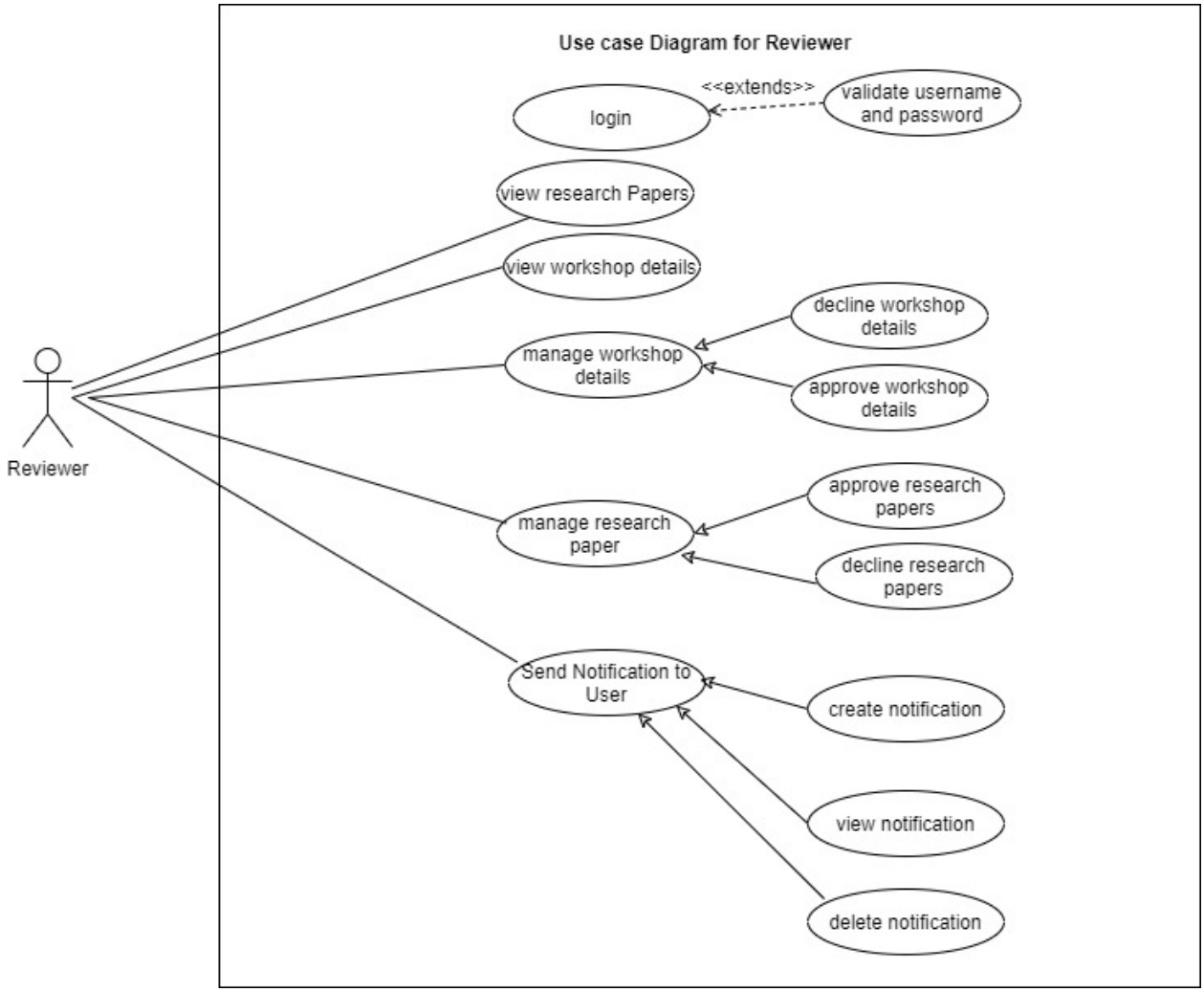
2.4.2 Rest API

Reviewer visit to web page then the component is mount to the DOM. The componentDidMount or useEffect method is called sending off the HTTP request . HTTP request can be as GET, POST, PUT or DELETE When reviewer click buttons it will send a request to router and from router it will send to controller. Then from controller it will call to paticular model and take the necessary data and will response to reviewer request by displaying data.

2.4.3 Test cases

Test case ID	Test Scenario	Test steps	Test Data	Expected Results	Actual Results	Pass/Fail
1	Check reviewer login with valid Data	1. Go to Site 2. Enter username 3. Enter password 4. Click submit	Userid= reviewer Password = reviewer123	reviewer should login to the dashboard	As expected	Pass
2	Verify field by entering valid email address of user to send notifications	1. Go to site 2. Move to send notification form 3. Type email address of user 4. Click send button	Email = nayomi@gmail.com	Notification should send successfully	As expected	Pass
3	Verify the field of status in approve or decline filled	1. Go to the site 2. Move to Approve or decline form 3. Select approve or decline to status field 4. Click submit	Status = Approved	Display Successful saved message	As Expected,	Pass

2.4.4 Use Case Diagram

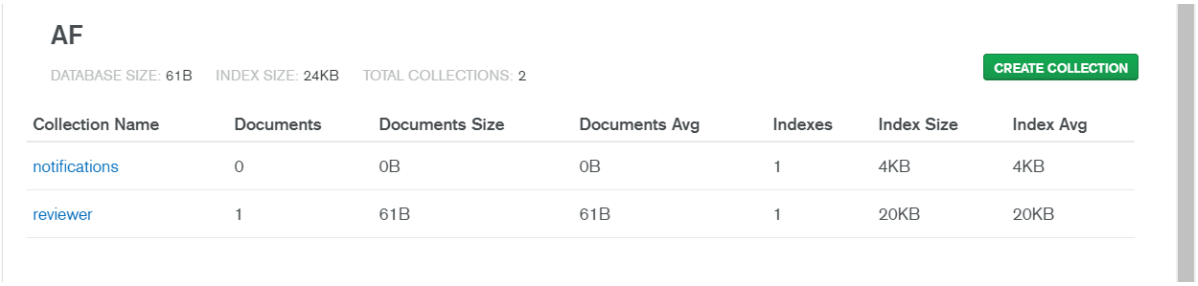


2.4.5 React Component Diagram

reviewerModel
- viewResearchPaper form
- viewworkshopdetails form
- Approve/Decline form
- sendNotification form
  - addNotifications
  - view listOfNotifications
  - deleteNotifications

2.4.6 Mongo database document screenshot


```
_id: ObjectId("60b1d987e64bf5a3b521e159")
status: "approved"
user_email: "sureni@gmail.com"
message: "this is approved"
created_date: 2021-05-28T18:30:00.000+00:00
document_id: ObjectId("60b1d987e64bf5a3b521e159")
```

AF

DATABASE SIZE: 61B INDEX SIZE: 24KB TOTAL COLLECTIONS: 2

CREATE COLLECTION

Collection Name	Documents	Documents Size	Documents Avg	Indexes	Index Size	Index Avg
notifications	0	0B	0B	1	4KB	4KB
reviewer	1	61B	61B	1	20KB	20KB