Bursary Management System

University of Jaffna

Developer Documentation

Copyright's Statement

Component

Bursary Management System.

Component Copyright Notice

License Text

Executive summary

// objective of the project
Goal or outcome to be achieved

Bursary is a fund provided by the university for the students who are not eligible for Mahapola and other financial aids. The main aim of the "Bursary Management System" is to efficiently sort and manage the bursary applications which are submitted by university students and filter the eligible students for this scheme. This system will analyse the eligibility criteria of each applicant and publish the name list of selected students. In addition, this system will be able to provide necessary information and summaries about the bursary payments.

Customer

The end user of this system will be university bursary applicants and Welfare Branch, University of Jaffna.

Problem statement

As we know, a huge number of students are chosen to attend University of Jaffna annually. Among them, most of the students want to apply for the bursary financial aid. Until now every student who wishes to apply for the bursary has to fill the bursary application and must handover it to the welfare branch. After that the welfare branch staff have to check all the applications for verifying that all required documents are provided and they have to filter students one by one who are selected for the bursary funds.

Since the number of applicants is high, it is hard to manage manually with accuracy and efficiency.

It was time consuming and needed more manpower. Therefore as computer science students of the University of Jaffna, for our Team Software Project our team has decided to find some solution for these issues . Finally we decided to develop a system as a solution. This system is named as “**Bursary Management System”.**

Solution

Our team has developed a system to manage these issues and fulfill our client’s requirements. This system can be used by students to submit the filled applications. And also the staff of the welfare branch can use this system to access the students’ data and filter the eligible students. According to the details which are provided by students, the system will filter the eligible students name list as the outcome. There are so many criterias we used to filter students such as the annual salary of the parents, no of siblings who are still studying and so on. By using this system students and the welfare branch will be able to save their time and also get accurate outcomes.

Revision History

Our problem was addressed in the current procedure for bursary scholarship in the welfare branch of our university. A system was developed for bursary application and selecting students for the bursary fund. Their outcomes were finding the suitable students for giving bursary funds. Our main problem was whether the selected students will maintain the highly attendance in their academic activities or not.

Our Team

These are the team members who have developed this Bursary Management System.

Z.M Ardil	2017/CSC/045	btzardil@gmail.com
Vetharshana Thangarajah	2017/CSC/034	vvvttt479@gmail.com
Divya Varatharajan	2017/CSC/006	divyav.cs.006@gmail.com
A.I Lakmal	2017/CSC/016	isurulakmal858@gmail.com
P.A.W.G.R Perera	2017/CSC/027	rameshperera25@gmail.com

Supervisor : Dr. A. Ramanan

Mentors : Mrs. Janani
Ms. Mathangi

Github : https://github.com/zmardil/bursary-management-system.git

Software License

Point of Contacts for issues and support

- In case of any issues regarding our system you can contact our team members via email.
- Also you can use stack overflow for solutions.
- This documentation contains step by step guidance for all the processes which need to run our system.

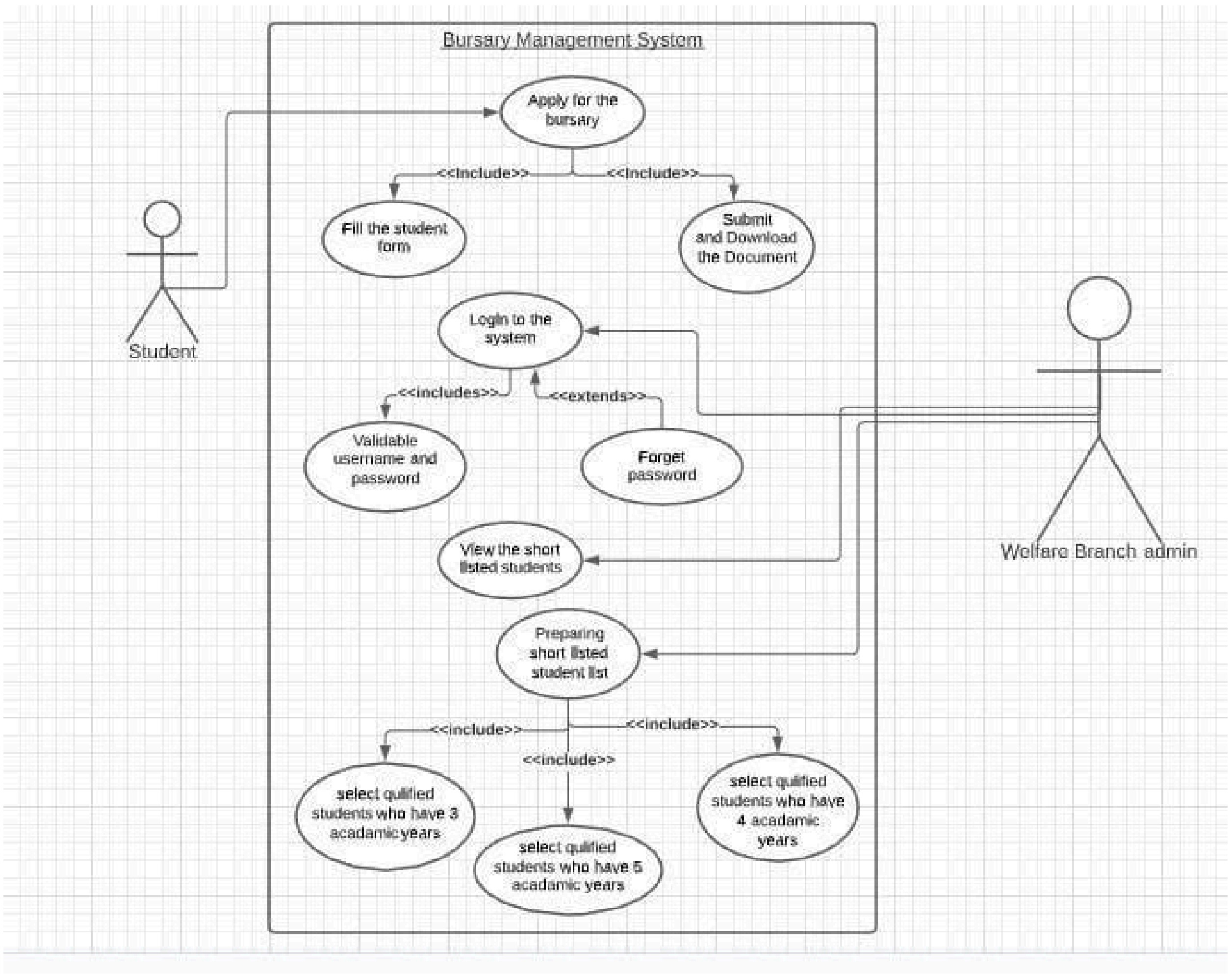
System Overview

In this platform designing students apply applications and clients maintain the fund's application. Student can be filling the application without any wrong and it is easy fill, student need to add their email and, then receive the mail for applying, after filling the application receive the all filling application with GS and GS certify application, then they can take the certify from GS and DS then they need to handover it for university,

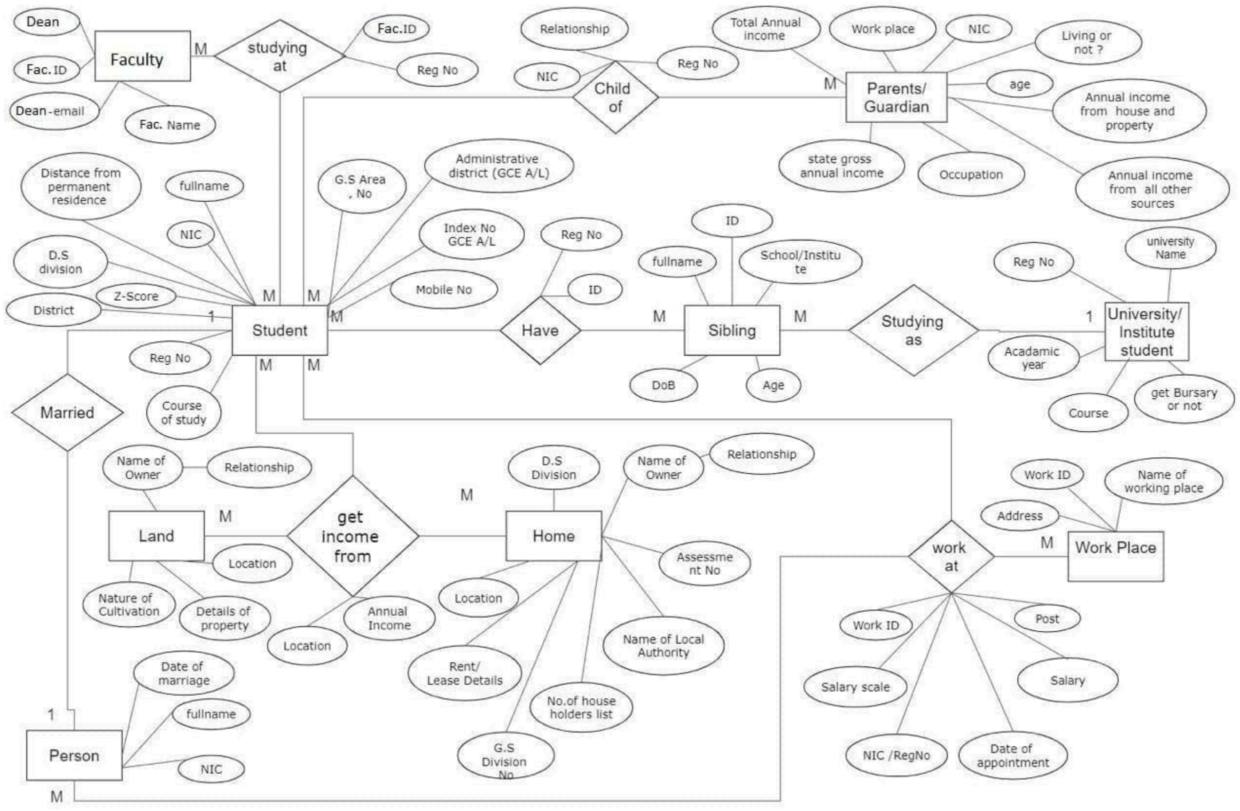
Then consider the client application they can see all of the data students have filled, they can add, delete, update, and choose who is the selection for the fund, they decide how amount how much should fund, and previous fund, when it needs to pay, last pay, how much have paid, then they need to pass that detail for the dean and he accept to pay the amount, a client can maintain the very easy this system with best.it is easy to save time for every user.This is system is designed for students to apply Bursary Fund to the university

Diagrams

- ### User Diagram


Bursary Management System
Apply for the bursary
<<include>>
<<include>>
Fill the student form
Submit and Download the Document
Student
Login to the system
<<includes>>
<<extends>>
Validiable username and password
Forget password
Welfare Branch admin
View the short listed students
Preparing short listed student list
<<include>>
<<include>>
<<include>>
select qulified students who have 3 acadamic years
select qulified students who have 5 acadamic years
select qulified students who have 4 acadamic years

- ### ER Diagram



System Installation

In order to install bursary-management-system there are few prerequisites, the steps are mentioned below,

The installation guide provided below is for Ubuntu LTS (x86_64 architecture), for other systems please visit the relevant links.

MongoDB installation & setup

MongoDB comprises two editions, Community & Enterprise Edition. The installation steps shown below are for the community edition. For more details and installation guide for the Enterprise edition please refer the link Install MongoDB — MongoDB Manual

1. Import the public key used by the package management system.

```
wget -qO - https://www.mongodb.org/static/pgp/server-4.4.asc | sudo
apt-key add -
```

2. Create a list file for MongoDB.

```
echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu
focal/mongodb-org/4.4 multiverse" | sudo tee
/etc/apt/sources.list.d/mongodb-org-4.4.list
```

3. Reload local package database.

```
sudo apt-get update
```

4. Install the MongoDB packages.

```
sudo apt-get install -y mongodb-org
```

   To specify other available versions of mongoDB refer to the stated link above.

5. Create a `bursary-management-system` database.

   To create the database use `mongo` command in the terminal, and paste the following command,

```
use bursary-management-system
```

For installation guide using .tgz tarball for Ubuntu and other platforms visit the official installation documentation provided above. For Windows systems simply use the `.msi` file (installer) provided here MongoDB Community Download | MongoDB, and make sure to create a data directory for the database and specify the path to the installer.

NodeJS installation & setup

Although NodeJS has a pre-built installer for every platform, the steps given below use nvm; a popular way to install NodeJS.

for pre-built Installers - Download | Node.js (nodejs.org)
for installation guide using nvm - nvm-sh/nvm: Node Version Manager - POSIX-compliant bash script to manage multiple active node.js versions (github.com)

6. Install `nvm` using bash.

```
wget -qO-
https://raw.githubusercontent.com/nvm-sh/nvm/v0.38.0/install.sh | bash
```

7. Install NodeJS using `nvm`

```
nvm install node
```

Application Installation & configuration

8. Download the application using `git`.

   ```
   git clone https://github.com/zmardil/bursary-management-system.git
   ```

9. Change directory to bursary-management-system.

   ```
   cd bursary-management-system
   ```

10. Duplicate or rename the `.env.example` file and configure the application's environment variables

    ```
    mv .env.example .env && vi .env
    ```

    Change `MONGODB_URI` to `mongodb://localhost:27017/bursary-management-system`, and other variables as preferred.

11. Install dependencies using `npm` (NodeJS package manager)

    ```
    npm install; npm install --prefix client
    ```

12. Start the application using `npm`

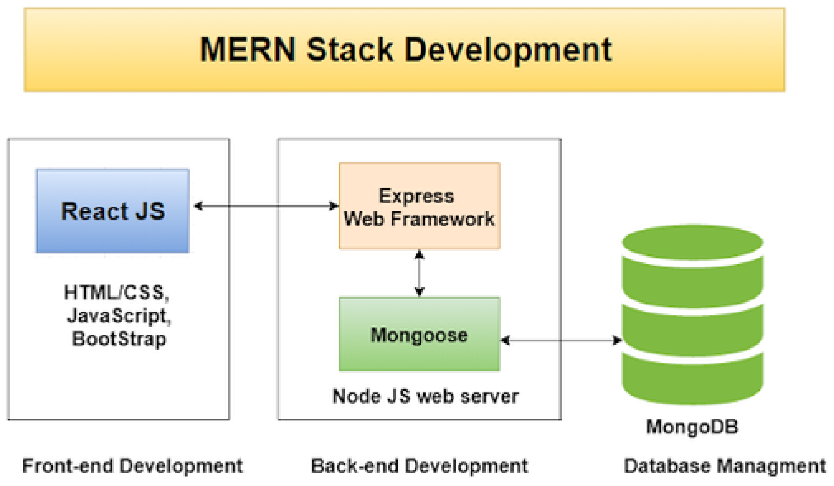
    ```
    npm start
    ```

To view the application open any browser and goto `localhost:[port]`. Here the port is as specified in the `.env` file.

Technologies and Tools

1. MERN stack

A standard high level structure for a MERN stack application and the components



- **Front-end development:**

The front end component responses to end user interactions and the communication with the server. In the structure displayed above, it uses React Js as its main framework which handles the user interactions and renders the information received from the server.

  - ☐ **JavaScript (React) -** https://reactjs.org/docs/getting-started.html
  - ☐ **Node js -** https://nodejs.org/en/docs/

- **Back-end development:**

The back end component responses to front end requests and database management. Express web framework is an open-source and lightweight web framework for Node Js. And Mongoose is the main database for MERN development as other databases like SQL are not compatible with Node Js.

  - ☐ **Node js -** https://nodejs.org/en/docs/
  - ☐ **Express.js-** http://expressjs.com/en/api.html

- **Database management:**

MongoDB is the native driver for interacting with a mongodb instance. In this example, it is used as the shell for Mongoose.

  - ☐ **MongoDB-** https://docs.mongodb.com/

2. POSTMAN

Postman is a great tool when trying to dissect RESTful APIs made by others or test ones you have made yourself. It offers a sleek user interface with which to make HTML requests, without the hassle of writing a bunch of code just to test an API's functionality

Website : https://www.postman.com/downloads/

3. **MongoDB Compass**
   MongoDB Compass is a GUI to explore, analyze, and interact with the content stored in a MongoDB database without knowing or using queries.

   MongoDB Compass is an open-source tool.
   Website : https://www.mongodb.com/try/download/compass
4. **Visual studio code**
   Visual Studio Code combines the simplicity of a source code editor with powerful developer tooling, like IntelliSense code completion and debugging.
   Website : https://code.visualstudio.com/download

5. **Github**
   GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere.
   Website : https://github.com

6. **Packages**
   - **PdfMake**
     Website : https://pdfmake.github.io/docs/0.1/
   - **NodeMailer**
     Website : https://nodemailer.com/about/
   - **Jsonwebtoken**
   - **bcrypt**
   - **Nodemon**
   - **Mongoose**
   - **Moment**
   - **Dotenv**
   - **Express**
   - **Cors**
   - **React dependencies**

     Website : https://www.npmjs.com/

7. **Material-UI for UI design**

Main Components and Their classes and GUI

Database Schema

A database schema is an abstract design that represents the storage of our data in a database.

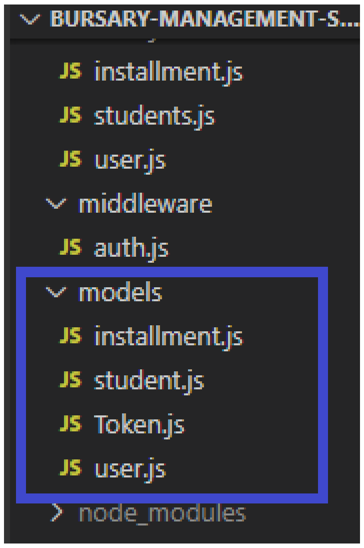
In the MERN stack, normally , we use no sql database which is MongoDB .

- **Why MongoDB?**

  If our application stores any data (user profiles, content, comments, uploads, events, etc.), then we're going to want a database that's just as easy to work with as React, Express, and Node. That's where MongoDB comes in.

Here , JSON documents created in your React.js front end can be sent to the Express.js server, where they can be processed and (assuming they're valid) stored directly in MongoDB for later retrieval. Again, if you're building in the cloud, you'll want to look at Atlas.

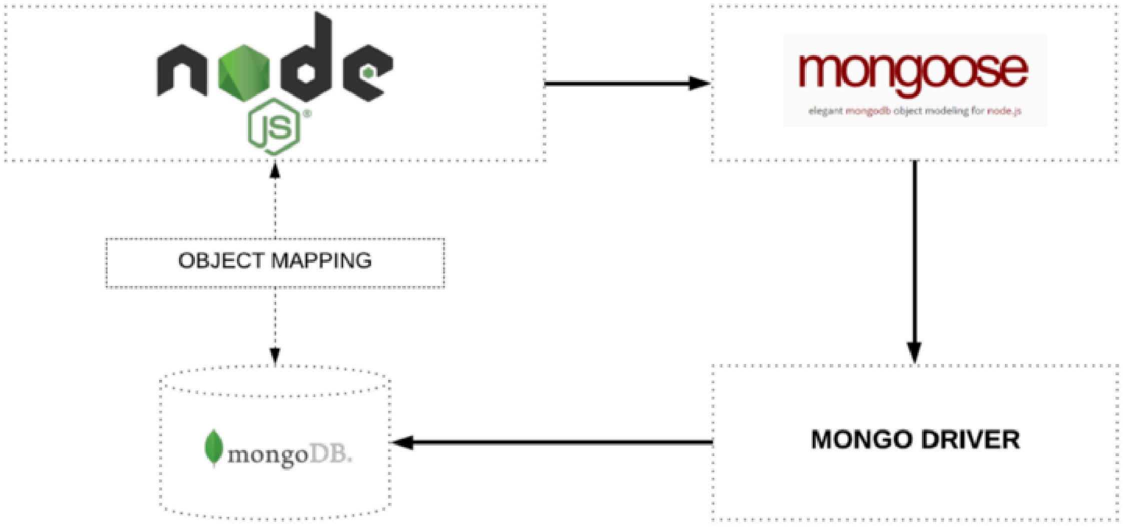
Now, when looking at models in the code - Here, we have 4 models.They are,



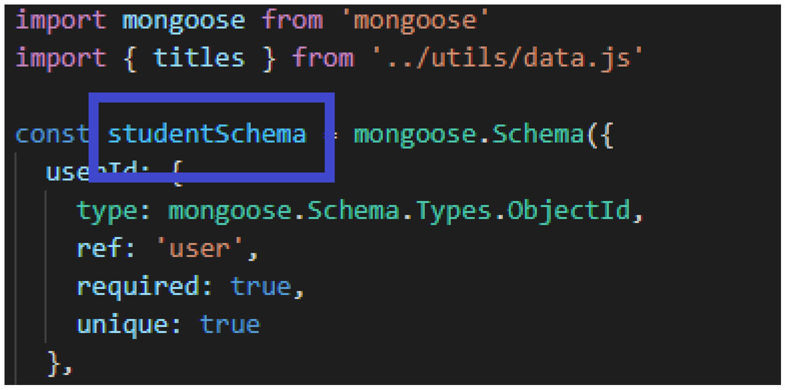
```
BURSARY-MANAGEMENT-S...
    installment.js
    students.js
    user.js
  middleware
    auth.js
  models
    installment.js
    student.js
    Token.js
    user.js
  node_modules
```

 Also we will export it to controllers (will be discussed)   by using mongoose. You may think what is mongoose ,

Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node. js. It manages relationships between data, provides schema validation, and is used to translate between objects in code and the representation of those objects in MongoDB.

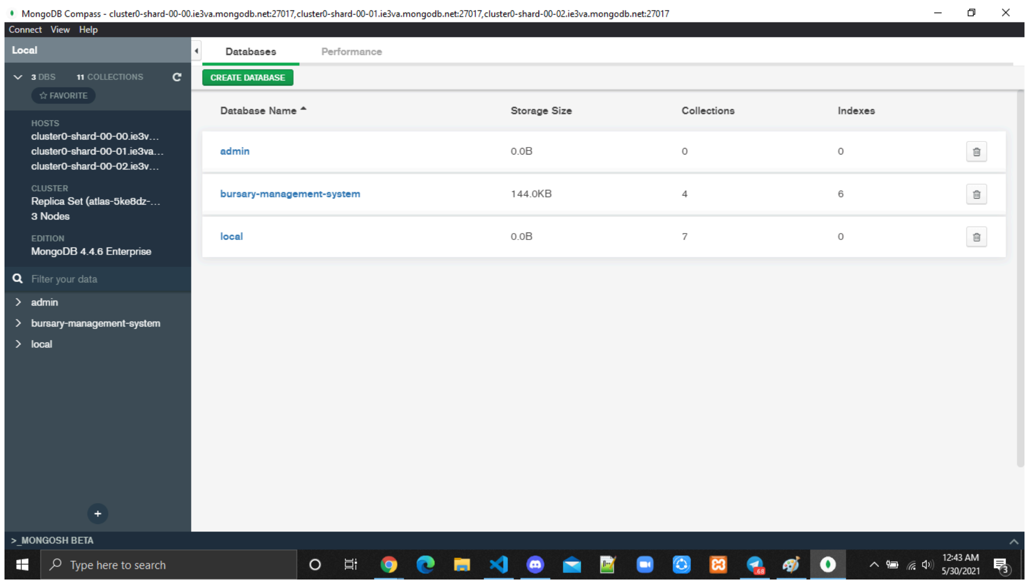


For Example, when we consider the student model, here we created a schema named “student schema” for the student model.



```
import mongoose from 'mongoose'
import { titles } from '../utils/data.js'

const studentSchema = mongoose.Schema({
  userId: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'user',
    required: true,
    unique: true
  },
```

Through this we can understand that each model has its own schema. To check the datas that we entered as an input , we can use , MongoDBCompass -> Paste your connection string and connect ->


MongoDB Compass - cluster0-shard-00-00.ie3va.mongodb.net:27017,cluster0-shard-00-01.ie3va.mongodb.net:27017,cluster0-shard-00-02.ie3va.mongodb.net:27017

Connect View Help

Local

3 DBS 11 COLLECTIONS

HOSTS
cluster0-shard-00-00.ie3v...
cluster0-shard-00-01.ie3va...
cluster0-shard-00-02.ie3v...

CLUSTER
Replica Set (atlas-5ke8dz-...
3 Nodes

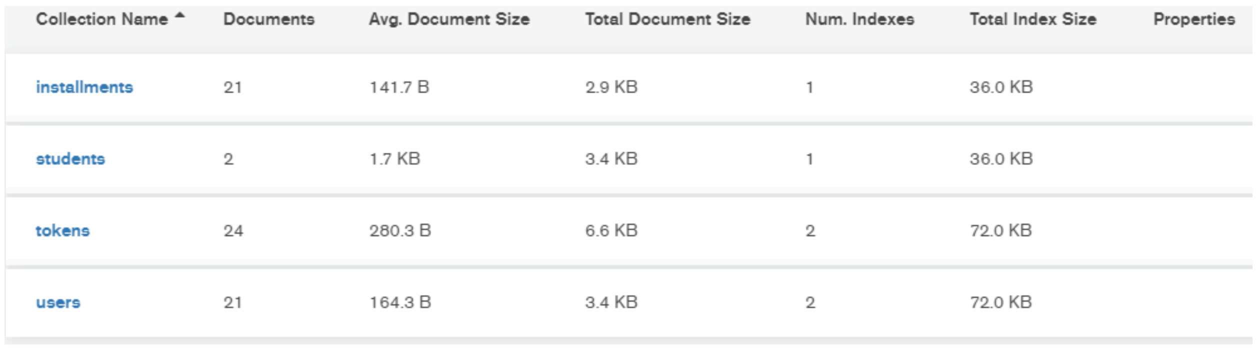
EDITION
MongoDB 4.4.6 Enterprise

- admin
- bursary-management-system
- local

Database Name	Storage Size	Collections	Indexes
admin	0.0B	0	0
bursary-management-system	144.0KB	4	6
local	0.0B	7	0

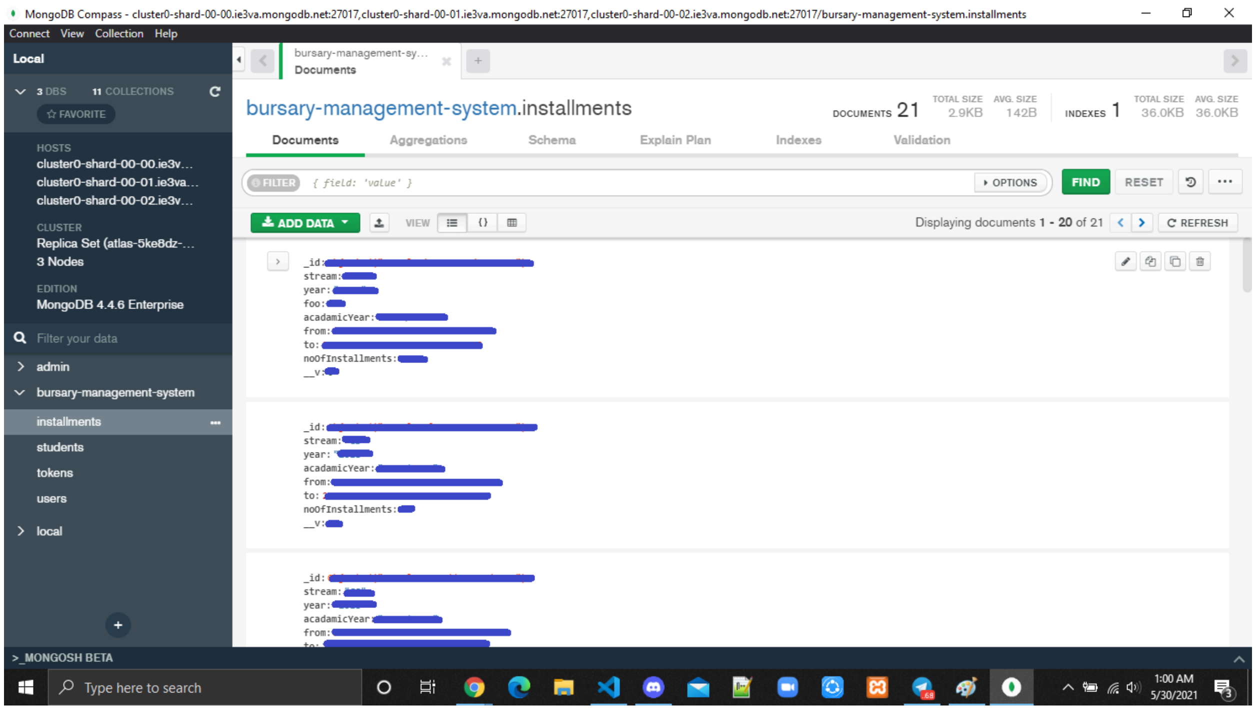
When you connect, You can see a window like this!!

These are the collections you can see in the Database ' Bursary Management System' When you click it!



Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
installments	21	141.7 B	2.9 KB	1	36.0 KB	
students	2	1.7 KB	3.4 KB	1	36.0 KB	
tokens	24	280.3 B	6.6 KB	2	72.0 KB	
users	21	164.3 B	3.4 KB	2	72.0 KB	

Click the collection that you want, to see the collection data. Where you can see like this!!

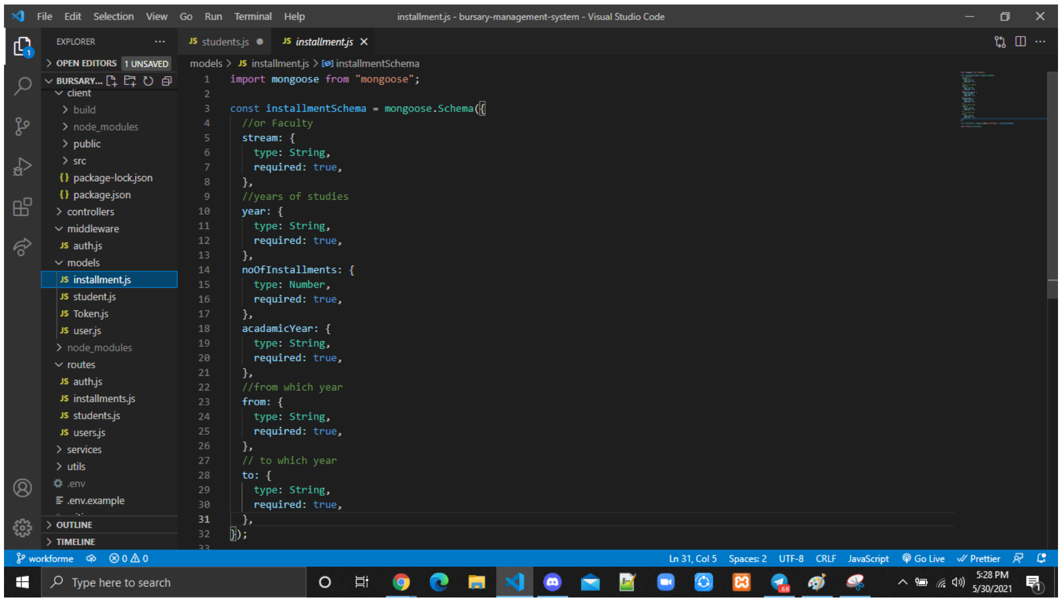


```
_id:[redacted]
stream:[redacted]
year:[redacted]
foo:[redacted]
acadamicYear:[redacted]
from:[redacted]
to:[redacted]
noOfInstallments:[redacted]
__v:[redacted]

_id:[redacted]
stream:[redacted]
year:[redacted]
acadamicYear:[redacted]
from:[redacted]
to:[redacted]
noOfInstallments:[redacted]
__v:[redacted]

_id:[redacted]
stream:[redacted]
year:[redacted]
acadamicYear:[redacted]
from:[redacted]
to:[redacted]
```

Let’s go through the code,

Student model, User model, Token model, Installment model files must import mongoose using it’s path.

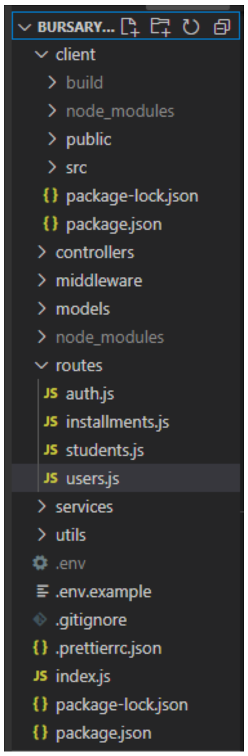
For example, installment model file be like,



```
import mongoose from "mongoose";

const installmentSchema = mongoose.Schema({
  //or Faculty
  stream: {
    type: String,
    required: true,
  },
  //years of studies
  year: {
    type: String,
    required: true,
  },
  noOfInstallments: {
    type: Number,
    required: true,
  },
  acadamicYear: {
    type: String,
    required: true,
  },
  //from which year
  from: {
    type: String,
    required: true,
  },
  // to which year
  to: {
    type: String,
    required: true,
  },
});
```

Main Components

As we said already, we used the MERN stack method for our project.
Here, our Directory Structure,



```
BURSARY...
  client
    build
    node_modules
    public
    src
    package-lock.json
    package.json
  controllers
  middleware
  models
  node_modules
  routes
    auth.js
    installments.js
    students.js
    users.js
  services
  utils
  .env
  .env.example
  .gitignore
  .prettierrc.json
  index.js
  package-lock.json
  package.json
```

Let’s start from the backend development.
First of all, we created models,

- Models

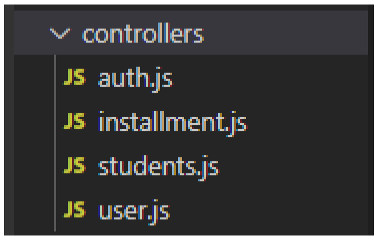
  As we mentioned earlier , you may have an idea about the stored data and the schema. And you know that our system has 4 models . By using them ,controllers were connected to do all of our request handling logic.

  You can see the code that we used for our system in Database schema content.

- controllers
  Instead of defining all of your request handling logic as Closures in route files, you may wish to organize this behavior using Controller classes. Controllers can group related request handling logic into a single class.

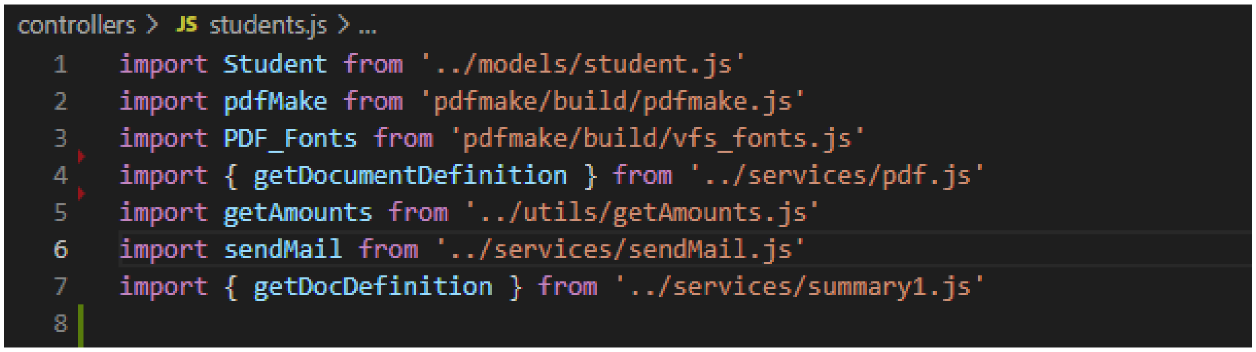
Controllers are stored in the app/Http/Controllers directory.

We used 4 type of controllers, They are,



```
controllers
  auth.js
  installment.js
  students.js
  user.js
```

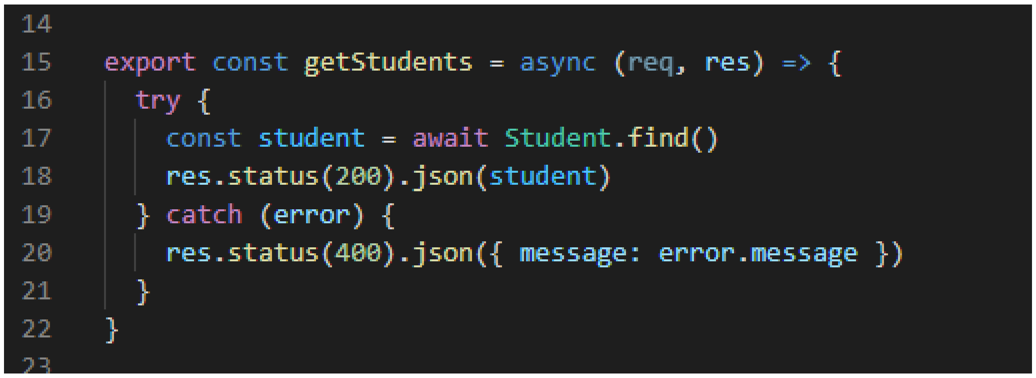
Let's consider the **student controller** part, where we import student model , pdfmake components,functions (**getAmounts** , **sendMail**) that we created and created pdf document formats using their paths.



controllers > students.js > ...

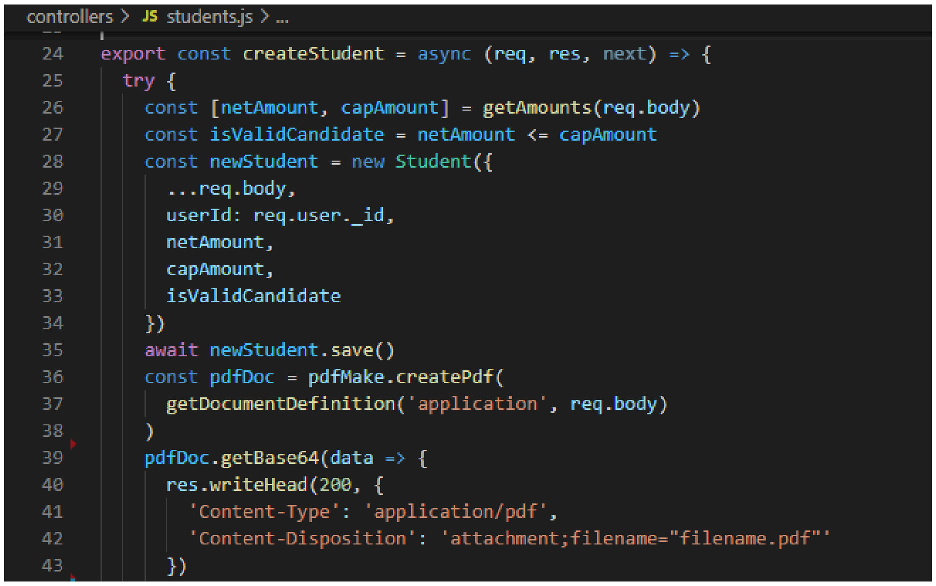
```
import Student from '../models/student.js'
import pdfMake from 'pdfmake/build/pdfmake.js'
import PDF_Fonts from 'pdfmake/build/vfs_fonts.js'
import { getDocumentDefinition } from '../services/pdf.js'
import getAmounts from '../utils/getAmounts.js'
import sendMail from '../services/sendMail.js'
import { getDocDefinition } from '../services/summary1.js'

```

We created a **getstudent** method , to retrieve student details!
This one will be used for the Administrator (Welfare office).



```

export const getStudents = async (req, res) => {
  try {
    const student = await Student.find()
    res.status(200).json(student)
  } catch (error) {
    res.status(400).json({ message: error.message })
  }
}
```

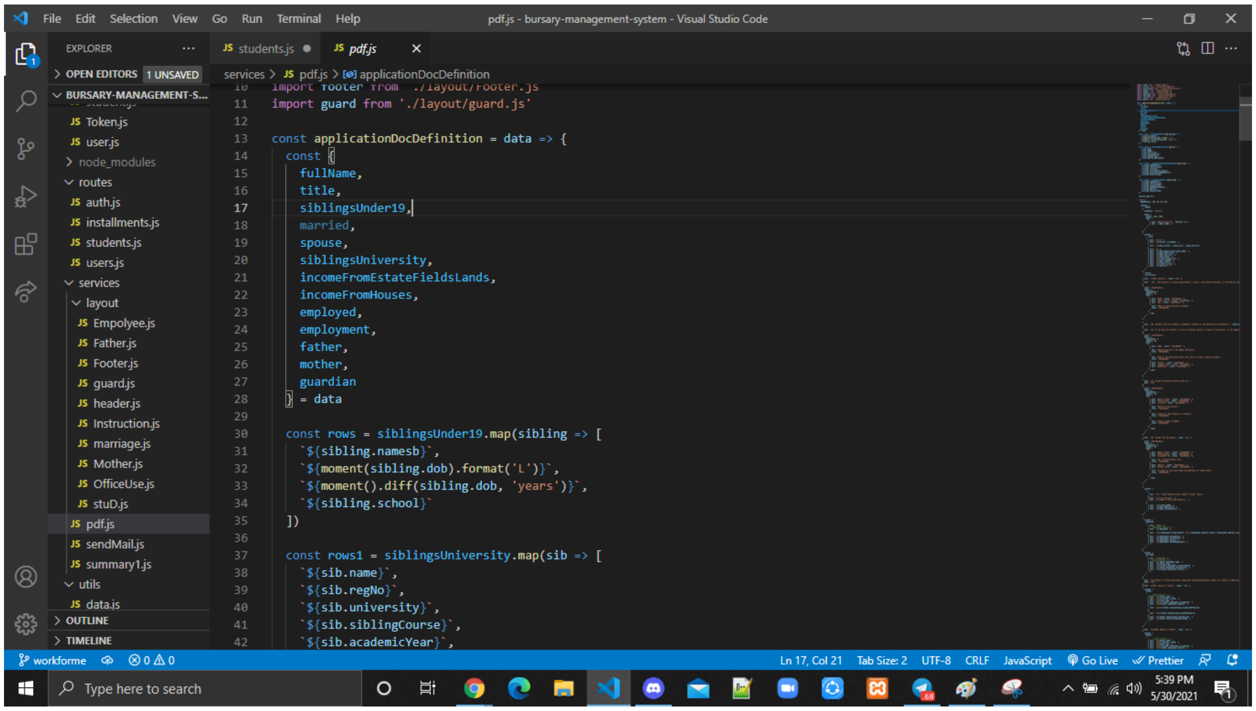
Also we need to focus on **createStudent.** Here we used better things, pdf creation , getting student details as input and sending email. Yes , this method is for students.



```
export const createStudent = async (req, res, next) => {
  try {
    const [netAmount, capAmount] = getAmounts(req.body)
    const isValidCandidate = netAmount <= capAmount
    const newStudent = new Student({
      ...req.body,
      userId: req.user._id,
      netAmount,
      capAmount,
      isValidCandidate
    })
    await newStudent.save()
    const pdfDoc = pdfMake.createPdf(
      getDocumentDefinition('application', req.body)
    )
    pdfDoc.getBase64(data => {
      res.writeHead(200, {
        'Content-Type': 'application/pdf',
        'Content-Disposition': 'attachment;filename="filename.pdf"'
      })
```

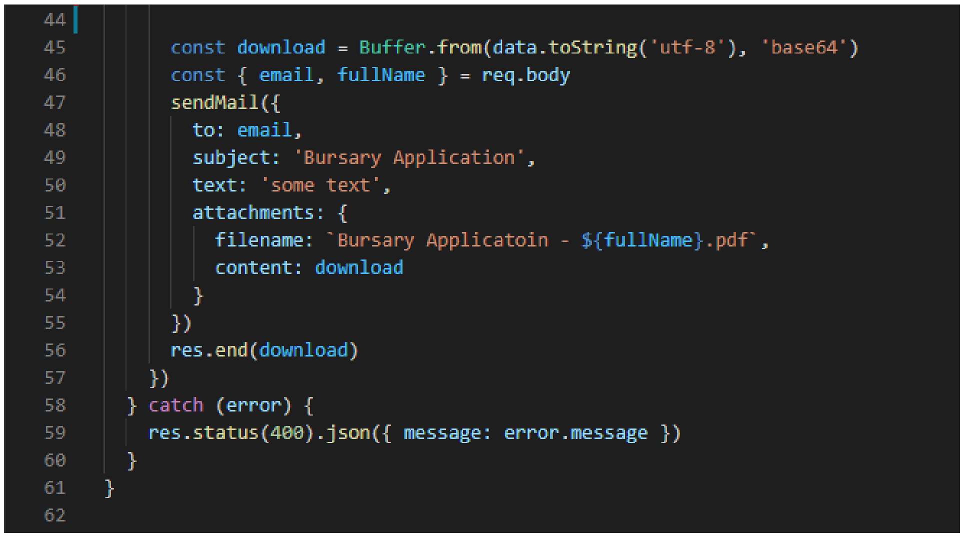
We get the net amount and cap amount from the imported getAmount file and find the valid student for bursary after that using input data from student we save their given details , also the amount details and valid for bursary or not in the database.

By stored data pdf will be created with the help of document definition (named as 'getDocumentDefinition') . To create the pdf we used **pdfMake.**



```javascript
import Footer from './layout/Footer.js'
import guard from './layout/guard.js'

const applicationDocDefinition = data => {
  const {
    fullName,
    title,
    siblingsUnder19,
    married,
    spouse,
    siblingsUniversity,
    incomeFromEstateFieldsLands,
    incomeFromHouses,
    employed,
    employment,
    father,
    mother,
    guardian
  } = data

  const rows = siblingsUnder19.map(sibling => [
    `${sibling.namesb}`,
    `${moment(sibling.dob).format('L')}`,
    `${moment().diff(sibling.dob, 'years')}`,
    `${sibling.school}`
  ])

  const rows1 = siblingsUniversity.map(sib => [
    `${sib.name}`,
    `${sib.regNo}`,
    `${sib.university}`,
    `${sib.siblingCourse}`,
    `${sib.academicYear}`,
```

Another thing is , when a student submits his/her form, their pdf will be downloaded and also he/she will get an email. To get mail we used **nodeMailer** .



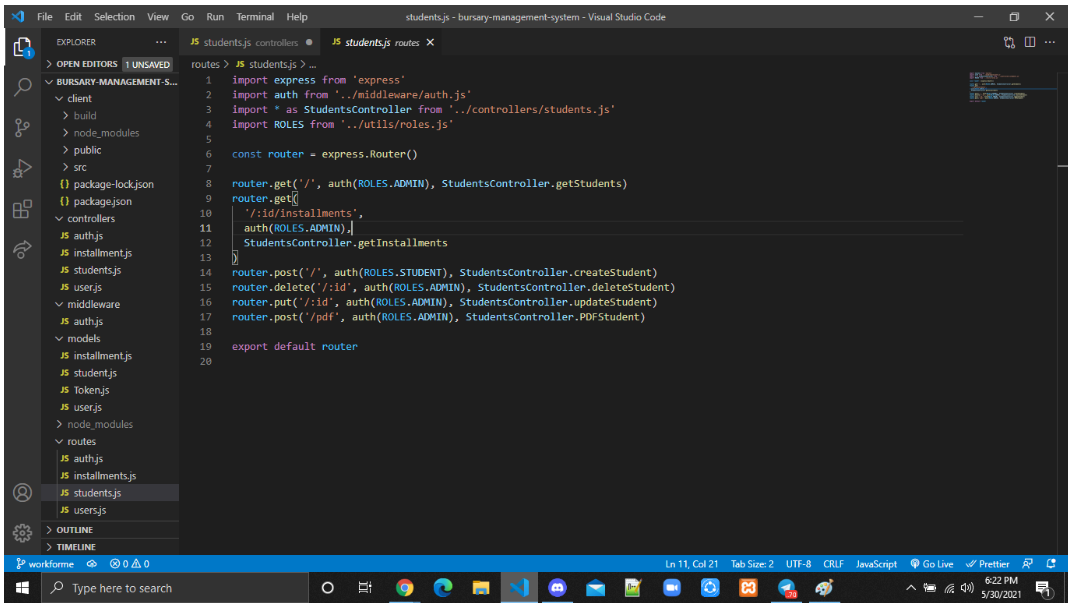
```javascript

        const download = Buffer.from(data.toString('utf-8'), 'base64')
        const { email, fullName } = req.body
        sendMail({
          to: email,
          subject: 'Bursary Application',
          text: 'some text',
          attachments: {
            filename: `Bursary Applicatoin - ${fullName}.pdf`,
            content: download
          }
        })
        res.end(download)
      })
    } catch (error) {
      res.status(400).json({ message: error.message })
    }
  }

```

As usual we created methods to delete , update and so on for the administrator.
Other controllers also have these types of methods.

Here , you want to know about middleware(auth.js) and the controller auth.js.
We used it to separate the roles of login.
Different roles will see different types of UI. So, we used middleware to make them accessible according to their roles.
Here we used **jsonwebtoken** and a timeout session was also included in auth.js controller.
In the user controller , we have register, delete , and so on methods .

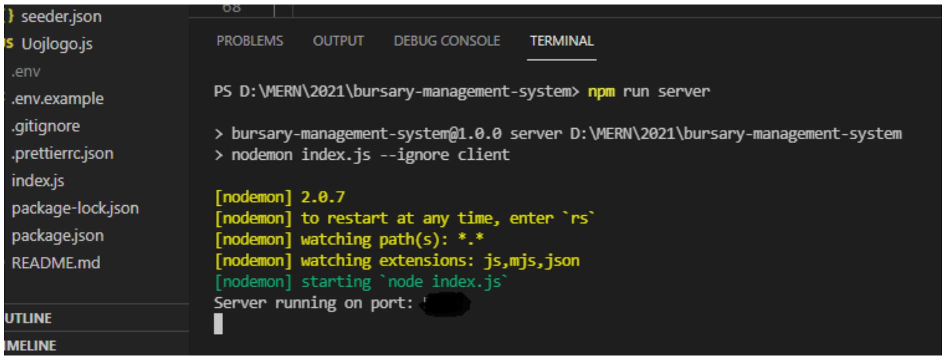
Through the router files we checked our url and tested our system's functionalities in a unique form by using postman.

Router file seems like,



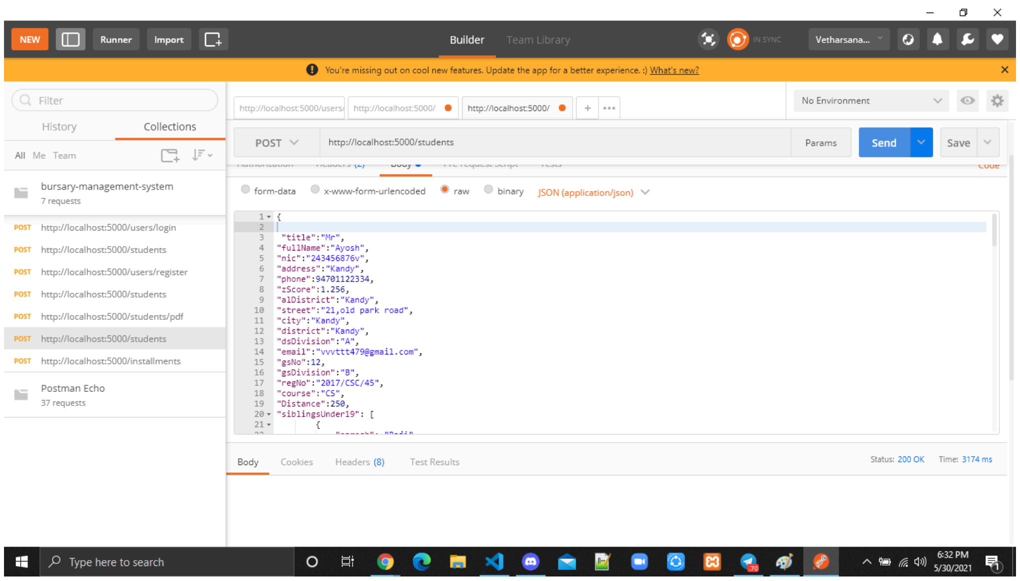
```
import express from 'express'
import auth from '../middleware/auth.js'
import * as StudentsController from '../controllers/students.js'
import ROLES from '../utils/roles.js'

const router = express.Router()

router.get('/', auth(ROLES.ADMIN), StudentsController.getStudents)
router.get(
  '/:id/installments',
  auth(ROLES.ADMIN),
  StudentsController.getInstallments
)
router.post('/', auth(ROLES.STUDENT), StudentsController.createStudent)
router.delete('/:id', auth(ROLES.ADMIN), StudentsController.deleteStudent)
router.put('/:id', auth(ROLES.ADMIN), StudentsController.updateStudent)
router.post('/pdf', auth(ROLES.ADMIN), StudentsController.PDFStudent)

export default router
```

When we checked through the postman ,
First we need to run the server,



```
PS D:\MERN\2021\bursary-management-system> npm run server

> bursary-management-system@1.0.0 server D:\MERN\2021\bursary-management-system
> nodemon index.js --ignore client

[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
Server running on port: [redacted]
```

As , a student if they submit their form like this,



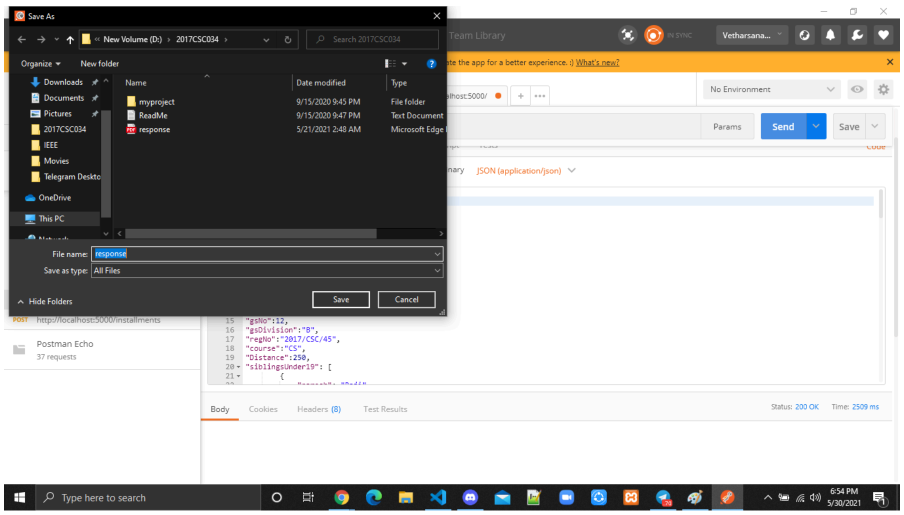
POST http://localhost:5000/students

Body: raw — JSON (application/json)

```
{

 "title":"Mr",
"fullName":"Ayosh",
"nic":"243456876v",
"address":"Kandy",
"phone":94701122334,
"zScore":1.256,
"alDistrict":"Kandy",
"street":"21,old park road",
"city":"Kandy",
"district":"Kandy",
"dsDivision":"A",
"email":"vvvttt479@gmail.com",
"gsNo":12,
"gsDivision":"B",
"regNo":"2017/CSC/45",
"course":"CS",
"Distance":250,
"siblingsUnder19": [
      {
```

Status: 200 OK   Time: 3174 ms

Their submitted form will be changed as pdf and it will be downloaded like this,

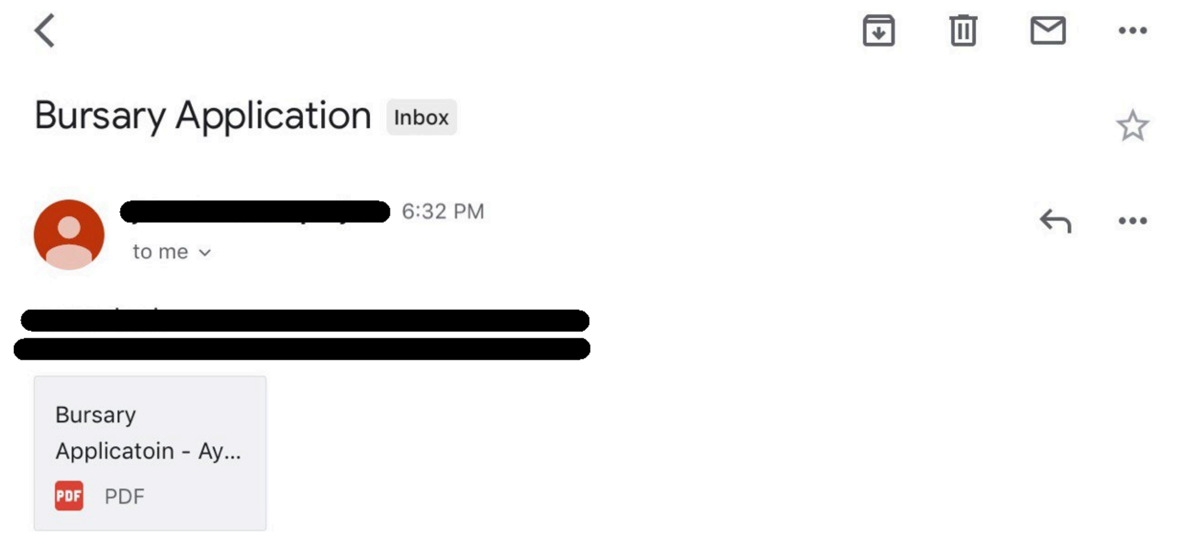


Name	Date modified	Type
myproject	9/15/2020 9:45 PM	File folder
ReadMe	9/15/2020 9:47 PM	Text Document
response	5/21/2021 2:48 AM	Microsoft Edge

File name: response
Save as type: All Files

Status: 200 OK   Time: 2509 ms

Also will receive mail like this,



Bursary Application Inbox

[redacted] 6:32 PM

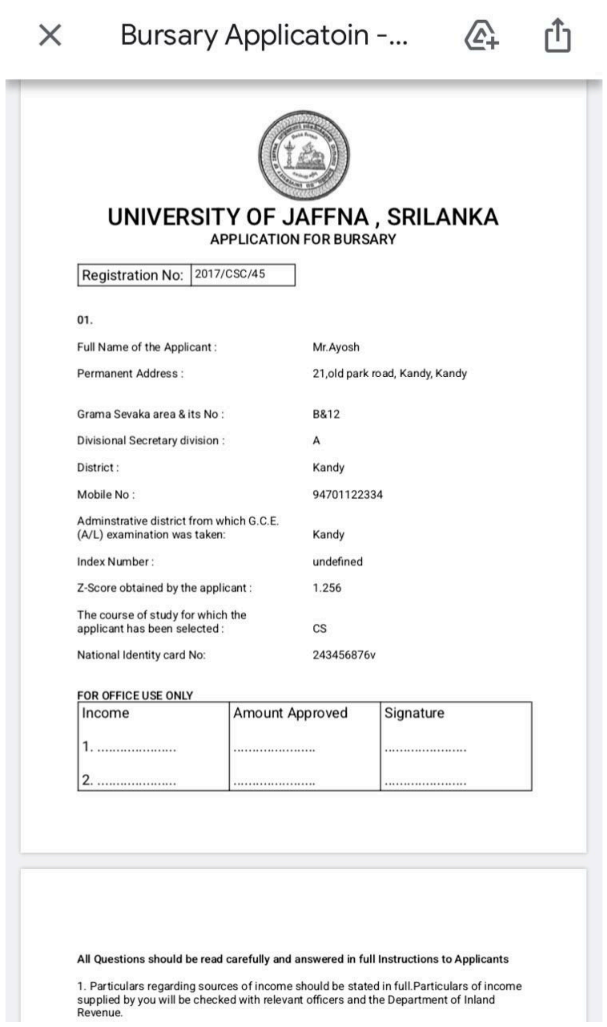
to me

[redacted]

Bursary Applicatoin - Ay...

PDF

And the pdf will be..



Bursary Applicatoin -...

UNIVERSITY OF JAFFNA , SRILANKA

APPLICATION FOR BURSARY

Registration No:	2017/CSC/45

01.

Full Name of the Applicant :	Mr.Ayosh
Permanent Address :	21,old park road, Kandy, Kandy
Grama Sevaka area & its No :	B&12
Divisional Secretary division :	A
District :	Kandy
Mobile No :	94701122334
Adminstrative district from which G.C.E. (A/L) examination was taken:	Kandy
Index Number :	undefined
Z-Score obtained by the applicant :	1.256
The course of study for which the applicant has been selected :	CS
National Identity card No:	243456876v

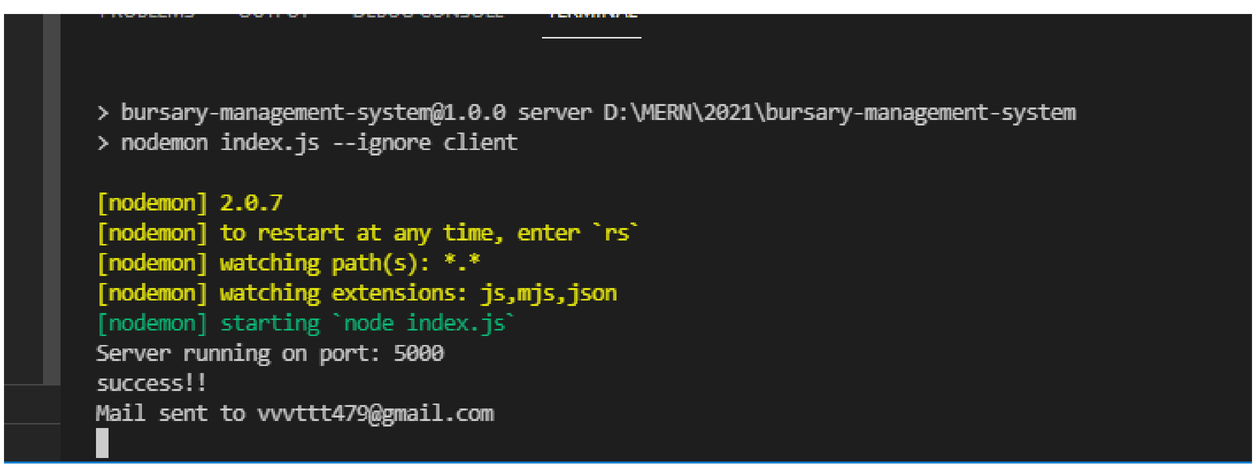
FOR OFFICE USE ONLY

Income	Amount Approved	Signature
1.	....	....
2.	....	....

All Questions should be read carefully and answered in full Instructions to Applicants

1. Particulars regarding sources of income should be stated in full.Particulars of income supplied by you will be checked with relevant officers and the Department of Inland Revenue.

We will receive a success message in the terminal.



```
> bursary-management-system@1.0.0 server D:\MERN\2021\bursary-management-system
> nodemon index.js --ignore client

[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
Server running on port: 5000
success!!
Mail sent to vvvttt479@gmail.com
```

This way we checked the system logic methods!!

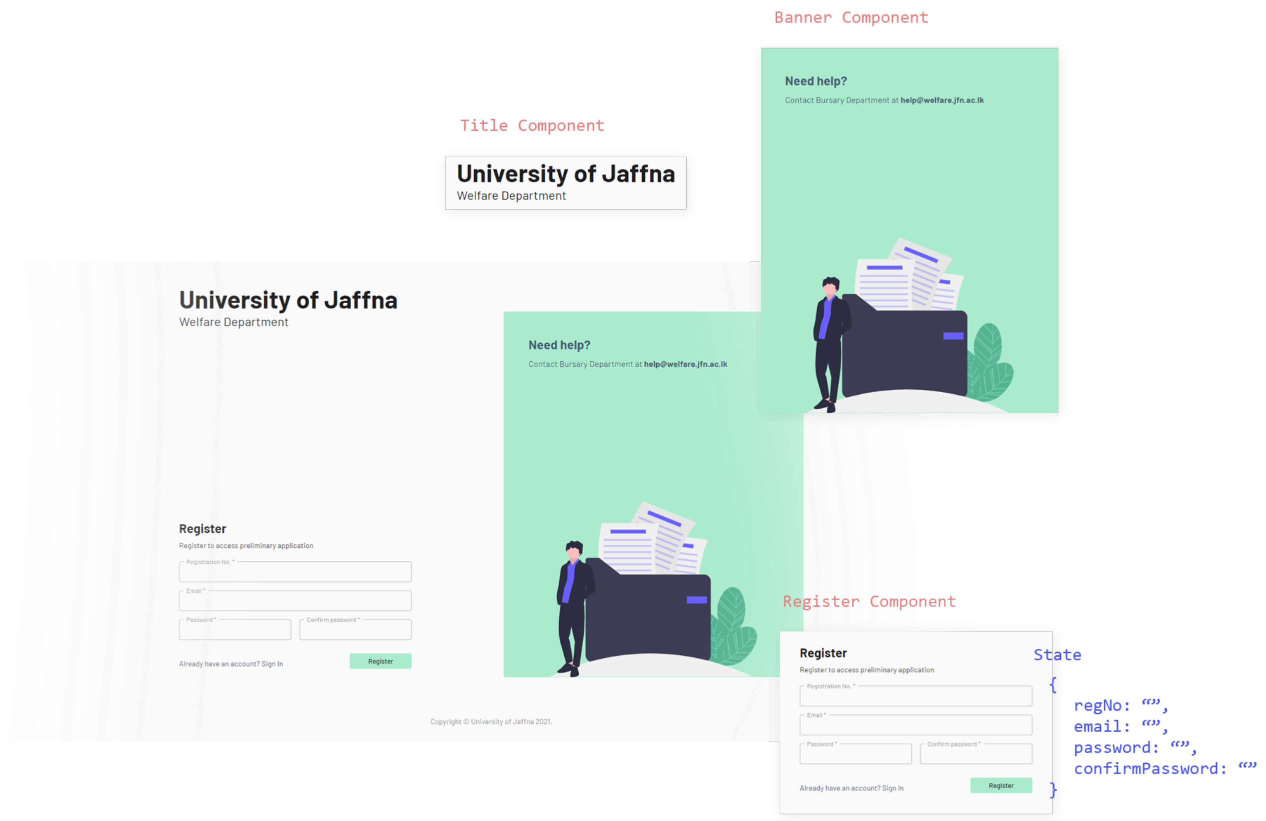
Here As I said already cap amount will be based on the student's siblings and net amount will be calculated based on father/mother/spouse/.. Salary. Our system will check those amounts and will give you a result as to whether the applied student is approved or not.

For Admin, we have added an installment controller to retrieve the installment details and approved student details.

Through the validation part in the system , our system will validate and will give error messages !!

Now let's see about the frond end part,

The team was challenged to discover and learn popular choices of technologies. ReactJS was an ideal front-end library for effortlessly building complex and robust user interfaces. React being a single-page-application, provides a clean architecture to use reusable components which manages its own state.



```
{
    regNo: “”,
    email: “”,
    password: “”,
    confirmPassword: “”
}
```

Material UI, uses Google's design system to provide a unified and consistent UI with rich react components.
[Material-UI: A popular React UI framework]

The system manages its state with redux, which makes it behave consistently via its components.
[Redux - A predictable state container for JavaScript apps. | Redux]

Client side validation lets the user flawlessly interact with the system.

Graphs lets the user visualize the data.

F

Best Practices?