**Name**:Nishar Bafna

**Email**: nisharbafna7@gmail.com

**Contact no**.: 9356532670

**Job Role:** Backend Developer Intern

**Challenge Details:**

Objective: Develop a robust and scalable Node.js backend using MongoDB as your database.

**Required Features:**

User Authentication: Implement a secure login/logout system.

MongoDB Integration: Use MongoDB for data storage. Demonstrate how to perform CRUD (Create, Read, Update, Delete) operations.

API Development: Create RESTful APIs for the following functionalities:

User Profile Management: Allow users to create, view, and edit their profiles.

Post Creation and Retrieval: Enable users to create posts and retrieve them.

Commenting System: Users should be able to comment on posts.

Error Handling: Implement comprehensive error handling and logging.

Documentation: Provide clear documentation on the setup and usage of your backend.

**Step by Step Making of the Project**

**1)Initializing Node.js Project**

Run the following command to initialize a new Node.js project: npm init

npm will generate a package.json file. This file contains metadata about your project and its dependencies.

Folder of **Package.json**

{

"name": "node\_task",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"keywords": [],

"author": "",

"license": "ISC",

"dependencies": {

"bcrypt": "^5.1.1",

"body-parser": "^1.20.2",

"ejs": "^3.1.9",

"express": "^4.18.2",

"mongodb": "^6.3.0",

"mongoose": "^8.0.2",

"multer": "^1.4.5-lts.1",

"yarn": "^1.22.21"

}

}

//color is because code is copied from VS CODE

Dependencies required for this task are as follows:

* npm install express
* npm install mongoose
* npm install bcrypt
* Npm install multer

**2)User Authentication**

Implement User Model

Created a User schema using Mongoose to define the structure of user data in MongoDB. In your models/User.js file:

const mongoose = require("mongoose");

mongoose.connect("mongodb://[127.0.0.1:27017/](http://127.0.0.1:27017/Backend_Task)User\_Management\_System");

const express = require("express");

const app = express();

//for user routes

const userRoute = require('./routes/userRoute');

app.use('/userRoute')

app.listen(3000, function(){

console.log("Server is running..");

});

User Registration

Developed an API endpoint to handle user registration. Used Express to define a route that receives user data, hashes the password using Bcrypt, and stores it in MongoDB.

//color is because of the code is copied from vs code

const User = require('../models/usermodel');

const bcrypt = require('bcrypt');

const securePassword = async(password)=>{

try{

const passwordHash = await bcrypt.hash(password, 10);

return passwordHash;

}

catch (error){

console.log(error.message);

}

}

const loadRegister = async(req, res) =>{

try{

res.render('registration');

}

catch(error){

console.log(error.message);

}

}

const insertUser = async(req, res) =>

{

try{

const spasswod = securePassword(req.body.password);

const user = new User({

name:req.body.name,

email:req.body.email,

mobile:req.body.mno,

image:req.file.filname,

name:req.body.password,

is\_admin:0

});

const userData = await user.save();

if(userData){

res.render('registration', {message:"Successfully registered, Please verify email"});

}

else{

res.render('registration', {message:"Registration Failed"});

}

}catch(error){

console.log(error.message);

}

}

Module.exports = { loadRegister}

**3. MongoDB Integration**

* Connected to MongoDB,we used Mongoose, a MongoDB object modeling tool. First, installed Mongoose as a dependencies
* Performed CRUD Operations(Create, Read, Update, Delete)

**4. API Development**

Set up an Express.js application to handle API routes. In your main application file (e.g., index.js), initialize Express and set up middleware

Created API for all the required functions such as:

* Insert user
* Register User
* Edit Profile and many more…

onst express = require("express");

const user\_route = express();

user\_route.set('view engine', 'ejs');

user\_route.set('views', './views/users');

const bodyParser = require('body-parser');

user\_route.use(bodyParser.json());

user\_route.use(bodyParser.urlencoded({extended:true}))

const multer = require("multer");

const path = require("path");

user\_route.use(expres.static('public'));

const storage = multer.diskStorage({

destination:function(req,file,cb){

cb(null, path.join(\_\_\_dirname, '/public/userImages'));

},

filename:function(){

const name = Date.now() + '-'+file.originalname;

cb(null,name);

}

});

const upload = multer ({storage : storage});

const usercontoller = require("../controllers/userController");

user\_route.get('/register', usercontoller.loadRegister);

user\_route.post('/register', upload.single('image'),usercontoller.insertUser);

user\_route.get('/', usercontoller.loginLoad);

user\_route.get('/login', usercontoller.loginLoad);

user\_route.get('/logout', auth.isLogin, userController.userLogout);

user\_route.get('/home', auth.isLogin, usercontoller.loadHome);

user\_route.get('/edit', auth.isLogin, usercontoller.editLoad);

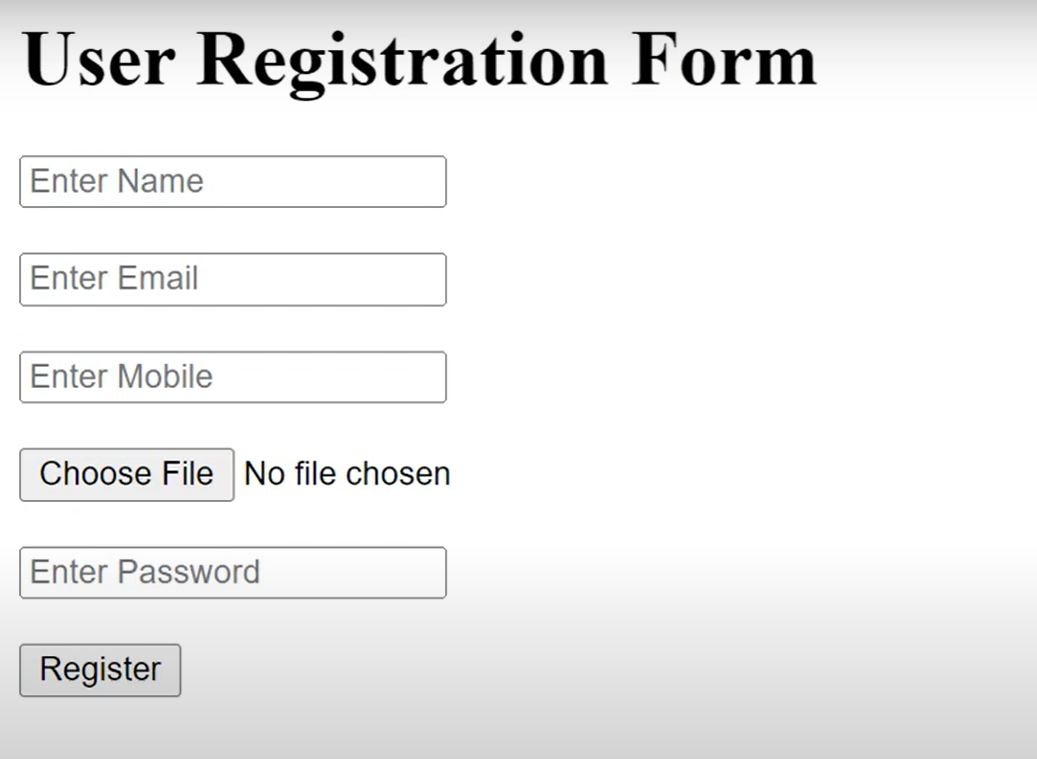
user\_route.post('/edit', upload.single('image'), userContoller.updateProfile);

module.exports = user\_route;

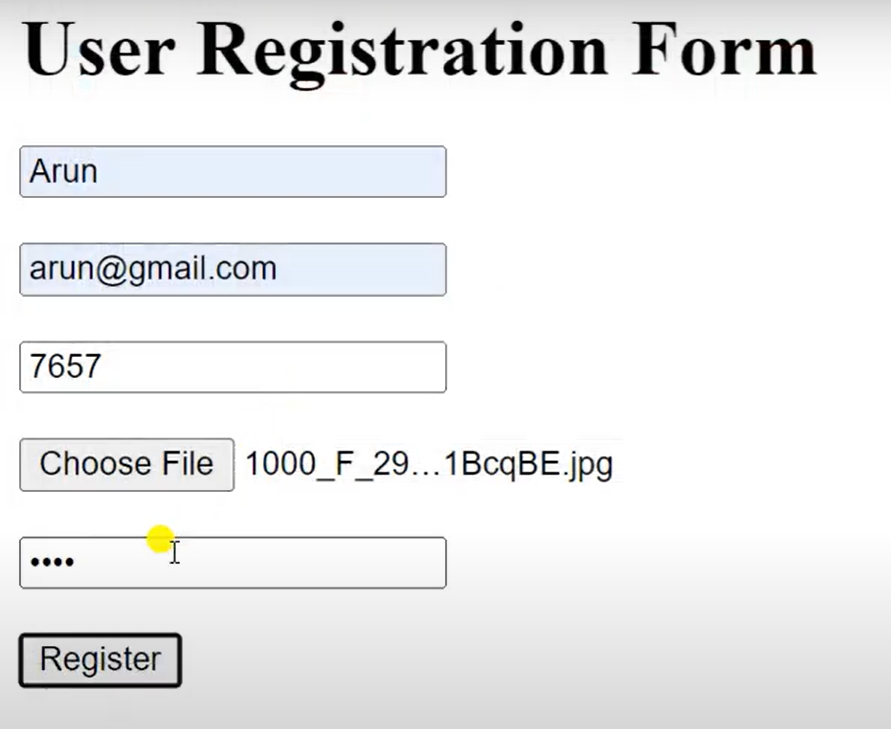
**4) Additional Feature**

I added an additional feature of User Verification via Email which updates on the MongoDB i.e Database in real-time.

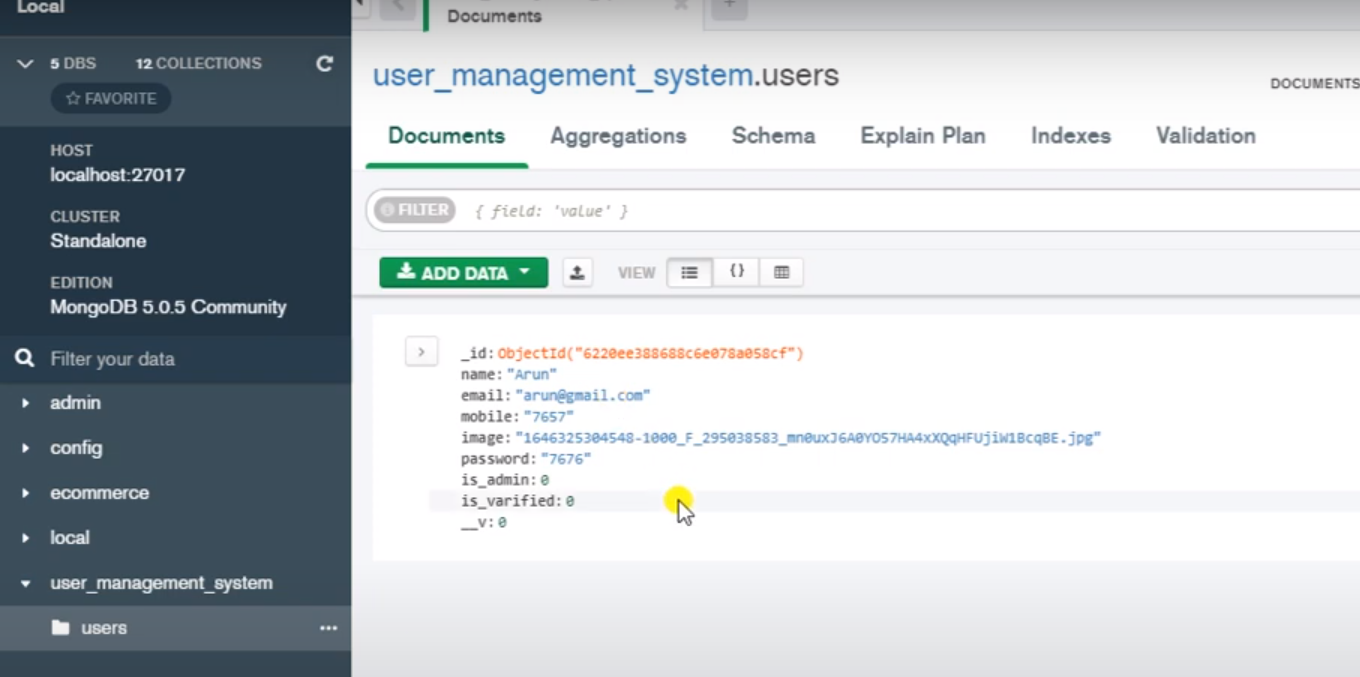
**5)Screenshots**

****

**Fig 1: Created User Registration Form**

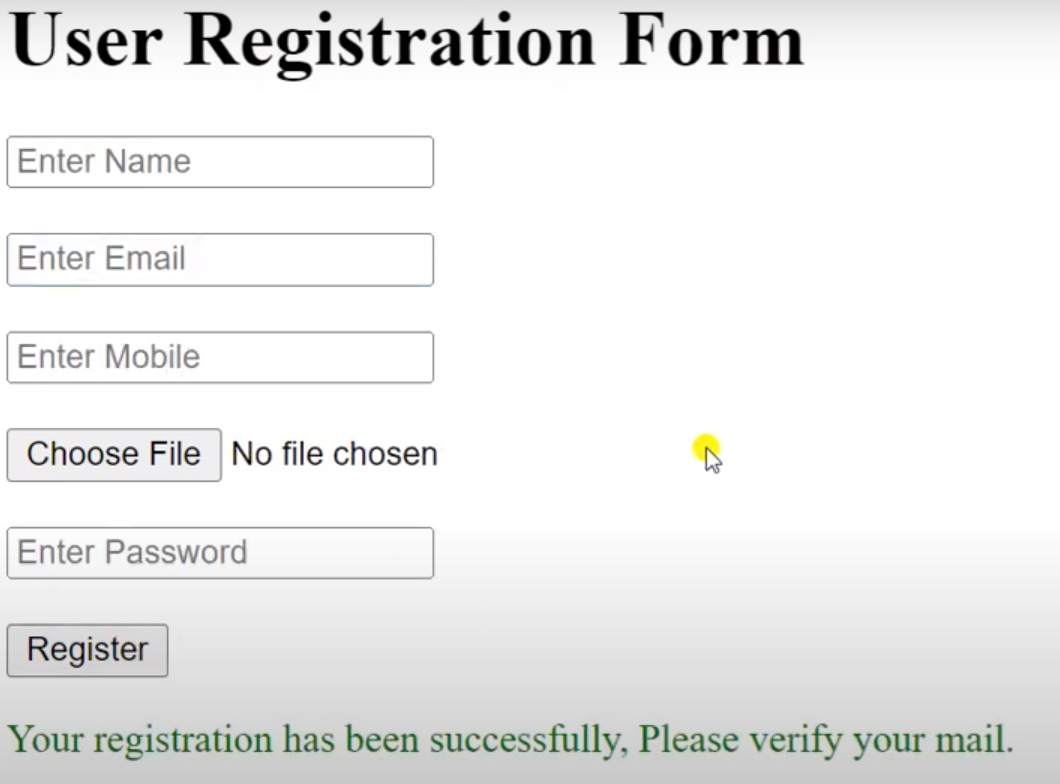
****

**Fig 2: Filled User Details**

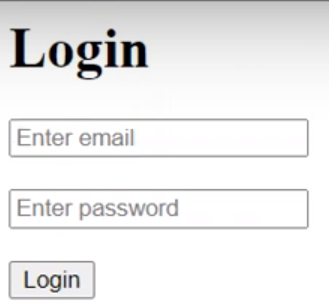
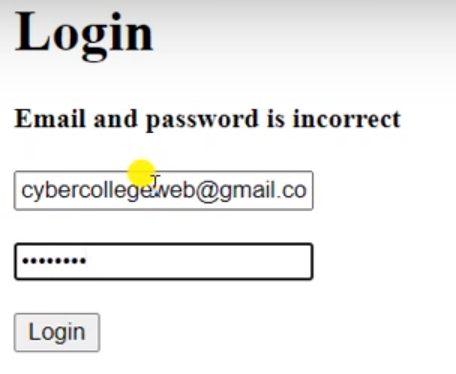
****

**Fig 3: Data Updated in MongoDB**

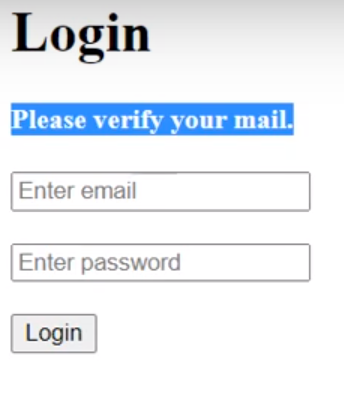


****

**Fig 4: Validation Of user(Additional) Fig 5: Password is encrypted**



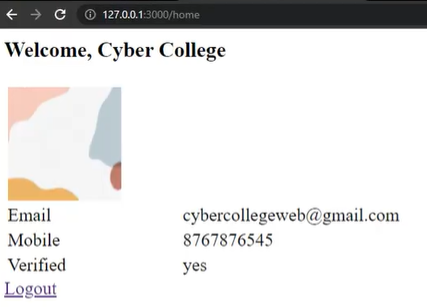
**Fig 6:Created Login Form Fig 7: Validation of Login Form**



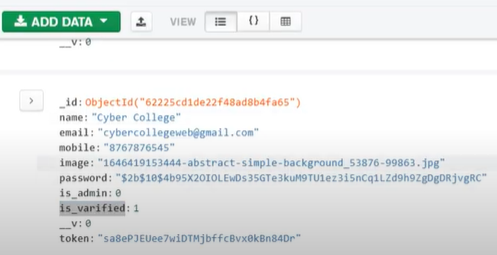
**Fig 8: Verification Of Email(Additional) Fig 9: Home Page i.e User Profile Page**



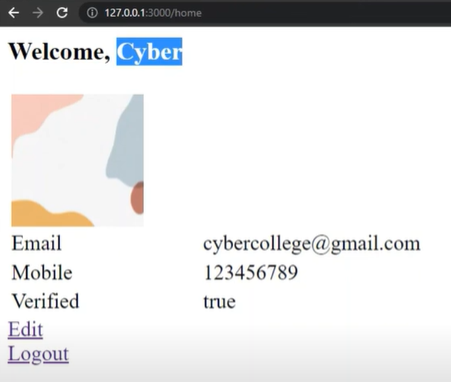
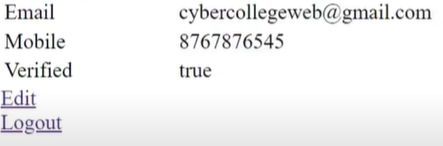
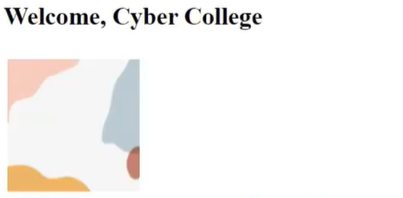
**Fig 9:Created Logout Link**



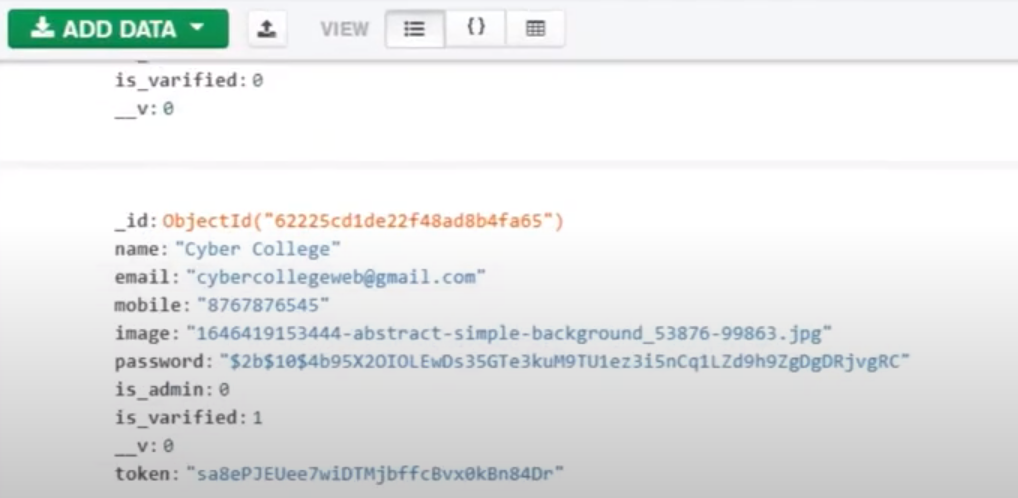
**Fig 10: User Profile Page with user details**



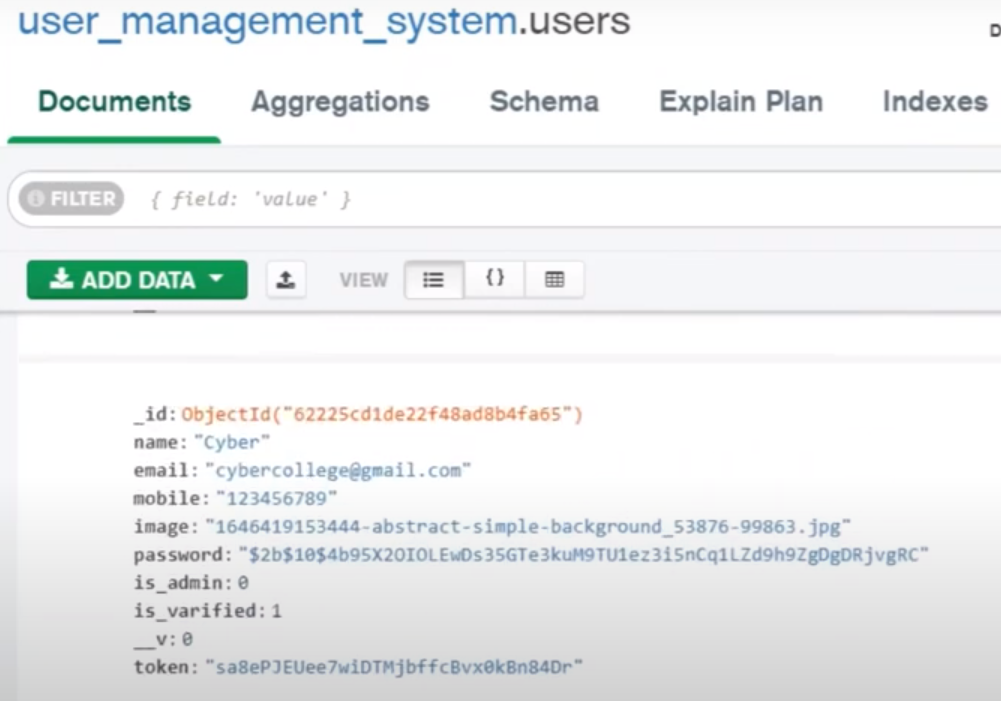
**Fig 11:Data Updated on DataBase**



**Fig 12:Created EDIT option**



**Fig 13:Edited USer Profile Fig 14: Old Data of User Profile**



**Fig 15: Updated (NEW) Data of User**