**Appendix A: Frontend**

The frontend of EduFlex is responsible for managing the user interface and interactions, providing a seamless and responsive experience for users. It consists of the following technologies:

HTML (HyperText Markup Language):

HTML is the standard language used to structure content for the web. It defines the layout and structure of EduFlex, including headings, forms, buttons, and other elements necessary for user interaction.

CSS (Cascading Style Sheets):

CSS is used to style the EduFlex platform, ensuring that the interface is visually appealing and responsive across various screen sizes. CSS provides control over the layout, typography, colors, and overall appearance of the platform.

JavaScript:

JavaScript (JS) is employed for interactivity on the platform, enabling dynamic content updates, form validations, and API requests without reloading the page. JS improves the overall user experience by making the interface more interactive and responsive.

**Frontend Installation Instructions:**

Install Node.js (prerequisite):

Download the latest LTS version of Node.js from nodejs.org.

Verify the installation by running node -v and npm -v in your terminal.

**Appendix B: Backend**

The backend of EduFlex is responsible for managing data processing, authentication, and communication with external APIs. It ensures that user actions on the frontend are handled securely and efficiently.

Node.js:

Node.js is a JavaScript runtime environment used for server-side operations in EduFlex. It handles the API requests, processes data, and serves the frontend with a responsive and scalable architecture.

Express.js:

Express.js is a web application framework for Node.js, simplifying the development of RESTful APIs. It manages the backend logic, including routing, middleware integration, and data handling between the frontend and external services.

Python:

Python is used to handle backend automation tasks, such as processing large datasets and performing scheduled operations. Its simplicity and extensive libraries make it ideal for backend workflows in EduFlex.

Flask:

Flask is a lightweight Python web framework used for specific backend processes. It provides a minimalistic and flexible environment for automating tasks like certificate verification and background processing in the EduFlex platform.

**Backend Installation Instructions:**

Install Node.js:

Download and install Node.js from nodejs.org.

Initialize the project directory using npm init and follow the prompts to set up a new project.

Install required packages from the packages.json by running command:

npm i

Create a .env file for server secret key and add a variable :

SERVER\_SEC\_KEY = "ANY\_KEY\_WORD"

Set up a basic backend server:

//import modules

const serverSK = process.env.SERVER\_SEC\_KEY;

const server = express();

server.use(cookieParser());

server.use(bodyParser.json());

server.use(bodyParser.urlencoded({ extended: true }));

// Use CORS middleware

server.use(cors({

    origin: '\*', // an coming request

    methods: ['GET', 'POST'],

    allowedHeaders: ['Content-Type', 'Authorization']

}));

mongoose.connect("mongodb://127.0.0.1/RMS");

server.post("/register", async (req, res) => {

    let userIP = req.headers['x-forwarded-for'] || req.socket.remoteAddress;

    if (Array.isArray(userIP)) {

        userIP = userIP[0];

    } else if (userIP.includes(',')) {

        userIP = userIP.split(',')[0].trim();

    }

    const credlylink\_template = "https://www.credly.com/users/"

    const { firstname , lastname, email , ph\_no , reguserUsername, reguserPwd, confuserPwd,credlylink, interface } = req.body;

    const specialCharRegex = /[!@#\$%\^&\\*\(\)\_\-=+]/;

    const passwordMinLength = 8;

    const existingUser = await User.findOne({ username: reguserUsername });

    if (existingUser) {

        return res.status(400).json({message : "Username already exists "})

    }

    if (reguserPwd.length < passwordMinLength) {

        return res.status(400).json({message :"Password should be at least 8 characters long."});

    }

    if (reguserPwd !== confuserPwd) {

        return res.status(400).json({message : "Passwords do not match."});

    }

    if(interface == "Webapp"){

        const newUser = new User({

            username: reguserUsername,

            password: reguserPwd,

            email : email,

            user\_type: "Student",

        });

        await newUser.save();

        logMessage(`[=] ${interface} ${userIP} : New student registered: ${reguserUsername}`);

        return res.status(200).redirect("/loginpage");

    }

    logMessage(`[=] ${interface} ${userIP} : New student registered: ${reguserUsername}`);

    return res.status(200).json({ message: "User registered successfully" });

});

server.listen(8000, () => {

    console.log(`http://localhost:8000`);

  });

Install Python and Flask:

Download and install Python from python.org.

Install required modules by the command:

pip install -r requirements.txt

create the folders named : apiuploads, hashtag\_extraction,uploads

Create a simple Flask app (app.py):

#import the modules

app = Flask(\_\_name\_\_)

CORS(app)

UPLOAD\_FOLDER = 'apiuploads'

if not os.path.exists(UPLOAD\_FOLDER):

    os.makedirs(UPLOAD\_FOLDER)

HASHTAG\_FOLDER = 'hashtag\_extraction'

if not os.path.exists(UPLOAD\_FOLDER):

    os.makedirs(UPLOAD\_FOLDER)

CERT\_DETECTION = 'uploads'

app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER

app.config['HASHTAG\_FOLDER'] = HASHTAG\_FOLDER

app.config['CERT\_DETECTION'] = CERT\_DETECTION

app.run(debug=True)

Run the Flask app by executing python app.py.

**Appendix C: API Integration**

EduFlex integrates with external services to enhance functionality. The Credly API is used to manage and issue digital badges and certificates for students' achievements.

Credly API:

The Credly API allows EduFlex to securely issue and validate digital credentials, ensuring that students’ achievements are recognized and stored on the platform. Through this API, EduFlex can manage badge creation, issue certificates, and display verified achievements on the students' profiles.

Credly API Integration Instructions:

Register and obtain API key:

Register for an account on Credly and request access to the API through the developer portal.

Generate an API key for authenticating requests.

Install Axios:

Install Axios, a promise-based HTTP client for Node.js, by running:

npm install axios

Set up API Requests:

Use Axios in your backend (server.js) to interact with the Credly API:

const axios = require('axios');

axios.get('https://api.credly.com/v1/badges', {

headers: {

Authorization: `Bearer ${process.env.CREDLY\_API\_KEY}`

}

})

.then(response => {

console.log(response.data);

})

.catch(error => {

console.error(error);

});