كلية الحاسبات والذكاء الإصطناعي
Faculty of Computers & Artificial Intelligence

جامعة حلوان
HELWAN UNIVERSITY

EDUGATE

Submitted in partial fulfilment of the requirements for the degree of Bachelor of Science in Computers & Artificial Intelligence, at the Computer Science Department, the Faculty of Computers & Artificial Intelligence, Helwan University

Supervised by:

[*Dr. Laila Abdelhamid*] **(June 2024)**

Acknowledgement

We extend our deepest appreciation to our supervisor, Dr. Laila Abdelhamid, for her invaluable guidance, support, and motivation throughout our project. Her expertise and valuable input have significantly shaped the direction and scope of our work, and we are sincerely grateful for her unwavering dedication to our success.

We would like to express our gratitude to the staff and faculty at Helwan University for providing us with exceptional resources and educational opportunities. The nurturing environment and supportive community have played a pivotal role in our personal and professional growth. We are truly grateful for the tremendous effort they have put into creating such an enriching learning experience.

Our heartfelt thanks go to our families and friends who have stood by us with unwavering support and encouragement, especially during the challenging moments. Their love and motivation have been a constant source of inspiration, and we consider ourselves fortunate to have them in our lives.

Lastly, we wish to acknowledge the invaluable contributions of all the participants who generously dedicated their time and insights to our study. Their active involvement has been instrumental in making this project possible.

We would like to express our sincere appreciation to everyone for their unwavering support and encouragement throughout our journey.

Table of Contents

Abstract

The education of a child is one of the most crucial aspects of their development, shaping their future prospects and opportunities. As parents, it is natural to strive for the best possible educational environment for our children, and sometimes that may involve considering a change of schools. However, the process of changing schools can be a daunting and complex task, presenting parents with numerous challenges and uncertainties.

This graduation project aims to explore and provide solutions to the problems encountered by parents when they decide to switch schools for their children. By understanding the difficulties involved and proposing effective strategies, we aim to support parents in making informed decisions and facilitating a smooth transition for their children.

By providing comprehensive information about school locations, comparing school fees, and offering visual resources like pictures and virtual tours, our project aims to streamline the school change process and make it smoother for parents. This will save them valuable time and effort, allowing them to make well-informed decisions for their children's education.

our graduation project can enhance the user experience and provide personalized school suggestions. By creating a platform that connects a database full of relevant information with the user, we can offer tailored recommendations based on their specific needs and preferences.

Chapter 1
Introduction

1.1 Overview

The education of a child is a critical factor in their overall development and future opportunities. As parents, always strive to provide the best educational environment for their children, which sometimes involves considering a change of schools. However, the process of switching schools can be overwhelming, presenting parents with various challenges and uncertainties. This graduation project aims to address these challenges by exploring and providing solutions that support parents in making informed decisions and ensuring a smooth transition for their children.

Challenges Faced by Parents:
To fully understand the difficulties encountered by parents when changing schools, we have identified several key challenges:

1. Visiting Multiple Schools: When parents consider switching schools, they often need to visit multiple institutions to gather information, assess facilities, meet staff, and understand the overall school environment. However, this can be time-consuming and exhausting, particularly if there are numerous potential schools to consider.

2. Distance: The proximity of the new school to the family's home is a crucial consideration for parents. However, relocating to a new area or finding a suitable school within a reasonable distance can pose challenges and necessitate thorough research.

3. School Fees: Financial considerations play a significant role in the decision-making process. Private or specialized schools often come with higher fees, and parents need to evaluate the affordability and long-term financial implications associated with different options.

4. School Environment: The internal dynamics, extracurricular activities, and overall school culture are essential factors to consider when changing schools. Parents must assess whether the new school's environment aligns with their child's needs and preferences.

1.2 Objectives:

The objectives of our graduation project are as follows:

1. Provide Comprehensive Information: We aim to gather and present detailed information about the locations of various schools. This will assist parents in assessing the convenience and feasibility of different school options.

2. Facilitate School Fee Comparison: We will compile and present a comparative analysis of school fees, including any available scholarship opportunities. This will enable parents to make well-informed decisions based on their budget and financial considerations.

3. Offer Visual Resources: To provide insight into the internal environment of schools, our project will include a collection of pictures and virtual tours showcasing classrooms, facilities, playgrounds, libraries, and other areas. These visual resources will provide parents with a sense of the school's atmosphere and amenities, saving them the effort of physically visiting multiple schools.

4. Personalized School Suggestions: By creating a user-friendly platform, we aim to connect a comprehensive database of schools with users. Through user profiles, we will gather information about their location, educational preferences, budget, and specific requirements. Using a matching algorithm, we will generate personalized school recommendations based on factors such as proximity, academic programs, fees, and user preferences.

5. User Feedback and Ratings: To enhance the platform, we will incorporate a user feedback and rating system. Users with previous experience or knowledge about recommended schools can provide valuable feedback, helping others make more informed decisions.

By achieving these objectives, our graduation project seeks to streamline the school change process, save parents valuable time and effort, and provide them with the necessary tools and information to make well-informed decisions for their children's education.

1.3 Purpose:

The purpose of this graduation project is to develop a comprehensive platform that addresses the challenges faced by parents when switching schools for their children. The project aspires to achieve the following:

1. Facilitate Informed Decision-Making: The platform aims to provide parents with the necessary information and resources to make well-informed decisions about changing schools. By offering comprehensive school profiles, fee comparisons, and visual resources, parents can evaluate various options and choose the most suitable school for their children.

2. Streamline the School Change Process: The project aims to simplify the process of switching schools by providing a user-friendly platform that centralizes relevant information and resources. By offering personalized school suggestions based on user preferences, parents can save time and effort in researching and visiting multiple schools.

3. Support Smooth Transition: The platform aims to support parents in facilitating a smooth transition for their children when changing schools. By providing insights into the internal environment of schools through visual resources, parents can assess whether a particular school aligns with their child's needs and preferences, ensuring a more seamless adjustment for the child.

1.4 Scope:

The scope of this project encompasses the following activities:

1. Planning: Defining project objectives, requirements, and timelines. Identifying the necessary resources, including the technology stack, personnel, and budget.

2. Designing: Developing the user interface (UI) and user experience (UX) design for the platform. Creating a database structure to store school information and user profiles. Designing algorithms for personalized school recommendations.

3. Implementing: Developing the backend and frontend functionalities of the platform. Integrating data sources for comprehensive school information. Implementing the matching algorithm to generate personalized school suggestions.

4. Testing: Conducting rigorous testing to ensure the platform functions as intended. Identifying and resolving any bugs or issues to ensure a smooth user experience.

5. Documentation: Preparing comprehensive project documentation, including project overview, objectives, methodologies, and technical specifications. Documenting the development process, including system architecture, data flow, and user guides.

1.5 General Constraints:

Throughout the project, several constraints hindered the completion timeline and workflow. These constraints include:

1. Time Constraint: The project had a limited timeframe, typically due to academic or graduation requirements. This constraint may have impacted the depth and extent of certain project features or functionalities.

2. Scope Ambiguity: Unclear or evolving project scope during the initial stages may have led to challenges in task prioritization, resource allocation, and feature implementation.

3. Data Accessibility: Difficulties in collecting comprehensive and up-to-date school information or accessing reliable data sources may have posed challenges in ensuring the accuracy and completeness of the platform's database.

4. Technical Limitations: Technical constraints, such as hardware or software limitations, may have affected the implementation or performance of certain features within the platform.

Chapter 2
Planning and analysis

2.1 Project planning

2.1.1 Feasibility Study:

- Technical Feasibility:

The technical feasibility of the project is high, as the development of a website to provide comprehensive information, school comparisons, and virtual tours is a well-established practice. The required technologies and tools are readily available, and the project can be implemented using web development frameworks and content management systems. The project team possesses the necessary technical skills and expertise to execute the development and maintenance of the website effectively.

- Operational Feasibility:

The project is operationally feasible as it aligns with the needs and expectations of parents seeking to change schools for their children. By providing a user-friendly platform with comprehensive information and personalized recommendations, the website will streamline the school change process and meet the demands of parents. The project team will conduct thorough research, gather accurate data, and establish partnerships with relevant educational institutions to ensure the website's effectiveness and relevance.

- Economic Feasibility:

The economic feasibility of the project is favorable. The revenue model will be based on advertising, partnerships with schools, and premium features. The project team will explore potential revenue streams, such as sponsored content and targeted advertisements, to ensure financial sustainability. Additionally, the cost of maintaining and updating the website will be balanced against the benefits derived from improved decision-making and convenience for parents.

- Legal and Ethical Feasibility:

The project will comply with all relevant legal and ethical considerations, including data protection and privacy regulations. User data will be handled securely and transparently, with appropriate consent obtained for data collection and processing. The project team will ensure that the website provides accurate and unbiased information, avoiding any conflicts of interest or misleading content. Intellectual property rights will be respected, and copyright permissions will be obtained for any visual resources used on the website.

Conclusion:

Based on the technical, operational, economic, legal, and ethical feasibility analysis, the school website project is deemed feasible and holds significant potential to address the challenges faced by parents when changing schools for their children. The project team is confident in its ability to develop and maintain the website successfully, providing valuable support to parents in their decision-making process.

2.1.2 Estimated Cost:

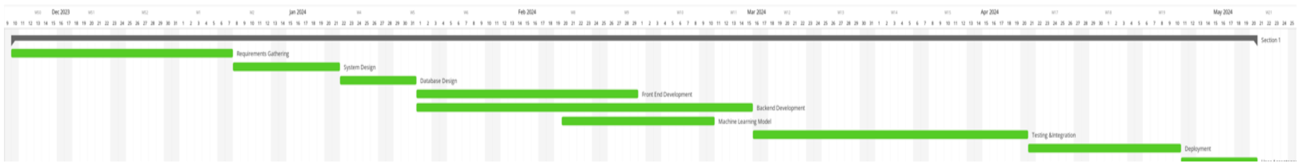
The estimated cost for developing and launching the school website project is as follows:

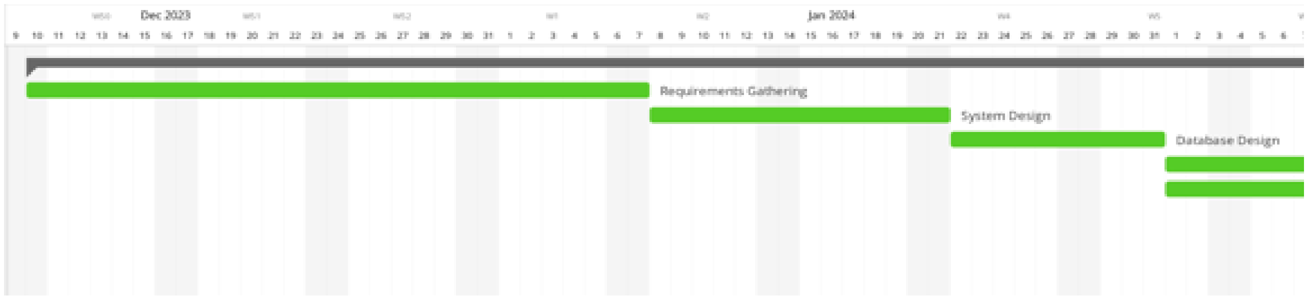
1. Website Development:
   - Web development team salaries: 3000L.E
   - Design and user experience: 2000L.E
   - Content creation and acquisition: 1500L.E
   - Website hosting and domain registration: 3000L.E
2. Database Management:
   - Database setup and maintenance: 2000L.E
3. Marketing and Promotion:
   - Digital marketing and advertising: 2000L.E
   - Partnerships and collaborations: 1000L.E
4. Ongoing Operations and Maintenance:
   - Website updates and improvements: 1000L.E
   - Customer support and administration: 2000L.E
5. Miscellaneous Expenses:
   - Legal and compliance: 1500L.E
   - Contingency: 1000L.E

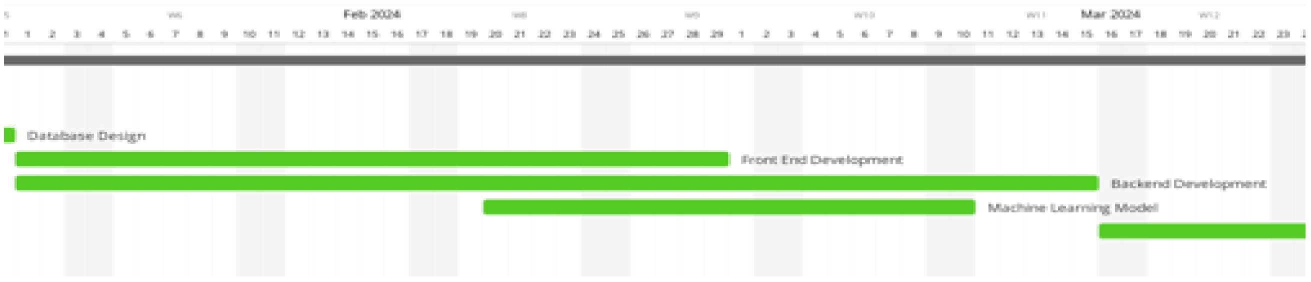
Total Estimated Cost: 20000L.E

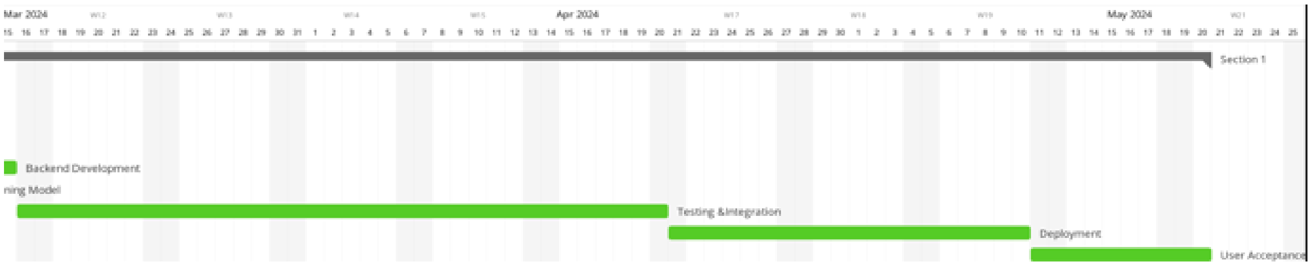
Please note that the above cost estimates are approximate and can vary depending on various factors, including the scope of the project, specific requirements, and the rates charged by service providers. A detailed cost analysis will be conducted during the project planning phase to ensure accurate budgeting and resource allocation.

2.1.3 Gantt Chart









2.2 Analysis and Limitation of Existing System:

The current system used by parents when changing schools for their children has several limitations that hinder its efficiency and effectiveness. These limitations include:

1. Limited Information: The current system lacks comprehensive information about different schools, including their location, facilities, academic programs, and extracurricular activities. This limited information makes it difficult for parents to make informed decisions and compare schools effectively.

2. Time-Consuming Process: Visiting multiple schools individually is a time-consuming and exhausting task for parents. They need to invest significant time and effort in gathering information, meeting with school staff, and assessing the facilities, which can lead to delays in the decision-making process.

3. Incomplete Comparisons: Due to the lack of a centralized platform, parents cannot easily compare different schools side by side. This leads to incomplete comparisons and makes it challenging to evaluate schools based on factors such as proximity, fees, academic programs, and school culture.

4. Subjectivity and Bias: The current system heavily relies on word-of-mouth recommendations and personal experiences, which can be subjective and biased. Parents may receive conflicting information or recommendations that do not align with their child's specific needs and preferences.

5. Limited Accessibility: The current system may not be easily accessible to all parents, especially those in remote areas or with limited internet access. This restricts their ability to explore a wide range of school options and find the most suitable one for their child.

These limitations contribute to a slow and inefficient process of changing schools, causing frustration and uncertainty for parents. Therefore, there is a need for a new system that overcomes these limitations and provides a more streamlined and effective solution.

2.3 Need for the New System:

The weaknesses of the old system highlight the need for a new system that addresses these shortcomings. The main reasons to migrate from the old system to a new system are:

1. Comprehensive Information: The new system aims to provide parents with comprehensive and up-to-date information about various schools. This includes detailed profiles, location information, academic programs, fees, and visual resources such as pictures and virtual tours. By having access to comprehensive information, parents can make well-informed decisions regarding their child's educational future.

2. Streamlined Process: The new system will streamline the school change process by offering a centralized platform where parents can explore multiple schools, compare them side by side, and generate personalized recommendations based on their preferences and requirements. This will save time and effort for parents, allowing them to make decisions more efficiently.

3. Unbiased Recommendations: Unlike the old system, which heavily relies on subjective recommendations, the new system will utilize algorithms and user profiles to generate unbiased and personalized school recommendations. This will ensure that parents receive relevant suggestions based on their specific needs, preferences, and constraints.

4. Enhanced Accessibility: The new system will be designed to be accessible to a wide range of parents, regardless of their location or internet connectivity. This will ensure that all parents have equal opportunities to explore school options and make informed decisions. By addressing the weaknesses of the old system and providing improved functionality and accessibility, the new system offers more efficient solution for parents when changing schools for their children.

2.4 Analysis of the New System:

2.4.1 User Requirements:

The new system should meet the following user requirements:

- Easy-to-use interface for parents to navigate and access information.

- Comprehensive school profiles with details on location, facilities, academic programs, and extracurricular activities.

- Advanced search and filtering options for parents to find schools based on specific criteria such as proximity, fees, and educational approach.

- Personalized recommendations based on user profiles and preferences.

- Ability to compare multiple schools side by side.

- Integration of visual resources like pictures and virtual tours to provide a glimpse into the school environment.

2.4.2 System Requirements:

The new system should fulfill the following system requirements:

- Stable and secure hosting infrastructure to ensure reliable access and data protection.

- Scalability to handle a growing database of schools and user profiles.

- Responsive design to ensure compatibility across different devices and screen sizes.

- Integration with external data sources for accurate and up-to-date information.

- Robust search functionality to enable quick and accurate retrieval of school information.

- Efficient data management and storage to handle large volume of schools.

2.4.3 Domain Requirements:

The new system should consider the following domain requirements:

- Compliance with data protection and privacy regulations.

- Collaboration with educational institutions to ensure accurate and relevant information.

- Regular updates and maintenance of the database to reflect changes in school profiles.

2.4.4 Functional Requirements:

The new system should include the following functional requirements:

- User registration and profile creation.

- School profile management for administrators.

- Advanced search and filtering options.

- Side-by-side school comparison feature.

- Personalized recommendation engine.

- Integration of visual resources.

- User feedback and rating system.

2.4.5 Non-Functional Requirements:

The new system should adhere to the following non-functional requirements:

- Performance: The system should provide fast response times and handle concurrent user requests efficiently.

- Security: The system should implement robust security measures to protect user data and prevent unauthorized access.

- Usability: The system should have an intuitive and user-friendly interface to ensure ease of use for parents.

- Reliability: The system should be reliable, with minimal downtime and data loss.

- Scalability: The system should be able to handle an increasing number of users and schools without compromising performance.

- Compatibility: The system should be compatible with different web browsers and devices.

2.5 Advantages of the New System:

The new system offers several advantages over the old system, including:

1. Comprehensive Information: The new system provides parents with access to comprehensive and up-to-date information about various schools, allowing them to make informed decisions based on a complete understanding of each school's offerings.

2. Time Efficiency: The new system streamlines the school change process by offering a centralized platform where parents can explore multiple schools, compare them side by side, and generate personalized recommendations. This saves parents valuable time and effort by eliminating the need for multiple visits and extensive research.

3. Personalized Recommendations: The new system utilizes user profiles and preferences to generate personalized school recommendations. This ensures that parents receive suggestions that align with their specific needs, preferences, and constraints, enhancing the likelihood of finding the most suitable school for their child.

4. Improved Accessibility: The new system is designed to be accessible to a wide range of parents, regardless of their location or internet connectivity. This ensures that all parents have equal opportunities to explore school options and make informed decisions, promoting inclusivity and equal access to educational opportunities.

5. Enhanced Decision-Making: With the new system's advanced search and filtering options, parents can easily find schools based on specific criteria such as

proximity, fees, and educational approach. The ability to compare schools side by side enables parents to make more informed and confident decisions, taking into consideration multiple factors simultaneously.

6. Visual Resources: The integration of visual resources such as pictures and virtual tours provides parents with a visual representation of the school environment. This allows them to gain a better understanding of the facilities, classrooms, and overall atmosphere, aiding in the decision-making process.

2.6 Risk and Risk Management:

The implementation of the new system carries certain risks that need to be identified and managed effectively. Some potential risks include:

1. Technical Risks: These include issues related to system stability, scalability, and data security. Mitigation strategies may involve conducting thorough testing, implementing robust security measures, and regularly monitoring and updating the system to address any technical vulnerabilities.

2. Data Privacy Risks: The collection and handling of user data pose risks related to data privacy and compliance with regulations. Risk management strategies may involve implementing appropriate data protection measures, obtaining user consent for data collection and processing, and ensuring compliance with relevant privacy laws.

3. User Adoption Risks: There may be challenges associated with user adoption and acceptance of the new system. To mitigate these risks, comprehensive user training and support should be provided, along with effective communication and change management strategies to ensure users understand the benefits and value of the new system.

4. Integration Risks: If the new system needs to integrate with existing systems or external data sources, there may be risks related to compatibility and data integration. Proper planning, coordination, and testing should be conducted to ensure seamless integration and minimize disruption to existing processes.

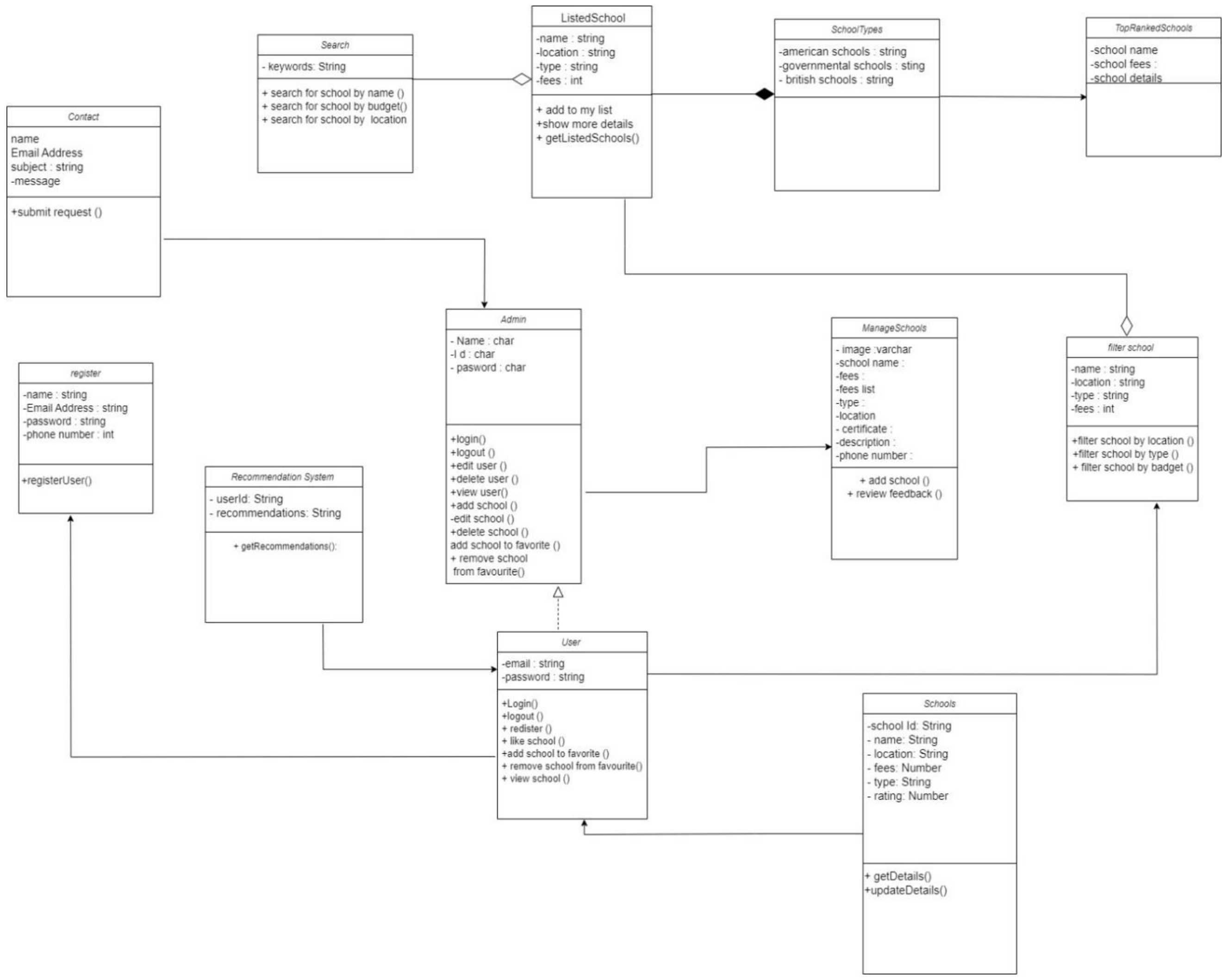
5. Financial Risks: The development and maintenance of the new system may involve financial risks, including cost overruns or insufficient revenue generation.

Effective financial planning, budgeting, and revenue generation strategies should be implemented to mitigate these risks.

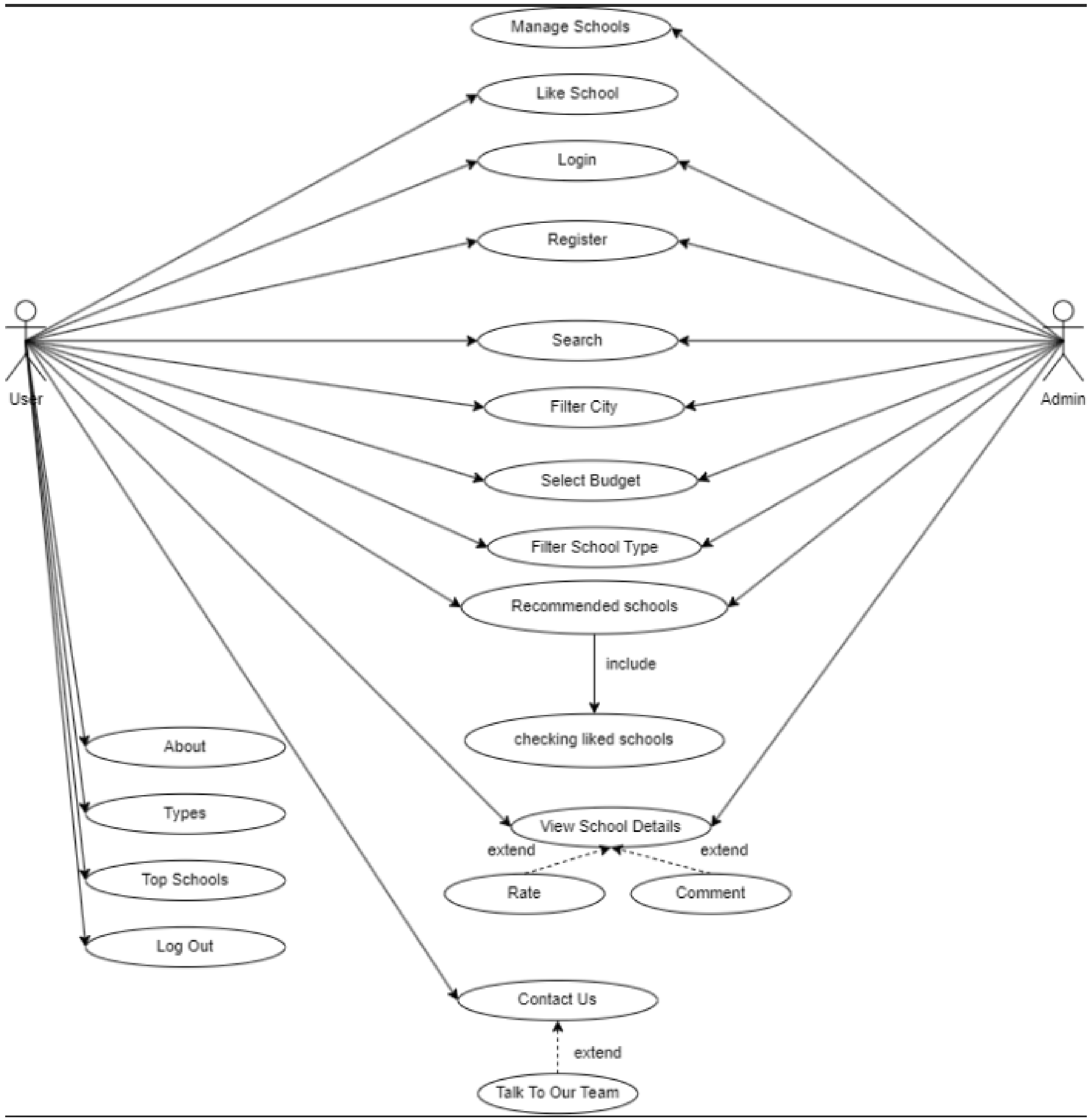
Risk management strategies should include proactive identification and assessment of risks, implementation of appropriate controls and safeguards, contingency planning, and ongoing monitoring and evaluation to address any emerging risks throughout the project lifecycle.

Chapter 3
Diagrams

3.1.1 class diagram



3.2 Use Case


Manage Schools
Like School
Login
Register
Search
Filter City
Select Budget
Filter School Type
Recommended schools
include
checking liked schools
View School Details
extend
extend
Rate
Comment
Contact Us
extend
Talk To Our Team
About
Types
Top Schools
Log Out
User
Admin

Actors

1. **User**
2. **Admin**

Use Cases and Descriptions

1. Register

- **Description**: Allows a new user to create an account.
- **Precondition**: User is not logged in.
- **Post condition**: User account is created and user is logged in.

2. Login

- **Description**: Allows a user to access their account by entering credentials.
- **Precondition**: User must be registered.
- **Post condition**: User is logged into the system.

3. Search

- **Description**: Allows the user to search for schools based on criteria.
- **Precondition**: User must be logged in.
- **Post condition**: Search results are displayed based on the criteria.

4. Filter City

- **Description**: Allows the user to filter schools by city.
- **Precondition**: Search has been performed.
- **Post condition**: Search results are filtered by the selected city.

5. Select Budget

- **Description**: Allows the user to filter schools by budget.
- **Precondition**: Search has been performed.
- **Post condition**: Search results are filtered by the selected budget.

6. Filter School Type

- **Description**: Allows the user to filter schools by type (e.g., public, private).
- **Precondition**: Search has been performed.
- **Post condition**: Search results are filtered by the selected school type.

7. Recommended Schools

- **Description**: Displays a list of recommended schools based on user's preferences.
- **Precondition**: User has set preferences (city, budget, school type).
- **Post condition**: Recommended schools are displayed to the user.

8. Checking Liked Schools

- **Description**: Allows the user to view schools they have liked.
- **Precondition**: User has liked at least one school.
- **Post condition**: Liked schools are displayed to the user.

9. View School Details

- **Description**: Displays detailed information about a selected school.
- **Precondition**: User has selected a school from the search results or recommended list.
- **Post condition**: Detailed information about the selected school is displayed.

10. Rate

- **Description**: Allows the user to rate a school.
- **Precondition**: User is viewing school details.
- **Post condition**: User's rating is saved for the school.

11. Comment

- **Description**: Allows the user to comment on a school.
- **Precondition**: User is viewing school details.
- **Post condition**: User's comment is saved and displayed for the school.

12. Manage Schools (Admin)

- **Description**: Allows the admin to add, edit, or delete school information.
- **Precondition**: Admin is logged in.
- **Post condition**: School information is updated in the system.

13. Like School

- **Description**: Allows the user to like a school.
- **Precondition**: User is viewing school details.
- **Post condition**: School is added to the user's list of liked schools.

14. About

- **Description**: Displays information about the application.
- **Precondition**: None.
- **Post condition**: About information is displayed.

15. Types

- **Description**: Displays information about different types of schools.
- **Precondition**: None.
- **Post condition**: Information about school types is displayed.

16. Top Schools

- **Description**: Displays a list of top schools.
- **Precondition**: None.
- **Post condition**: Top schools are displayed to the user.

17. Log Out

- **Description**: Logs the user out of the system.
- **Precondition**: User is logged in.
- **Post condition**: User is logged out and returned to the login screen.

18. Contact Us

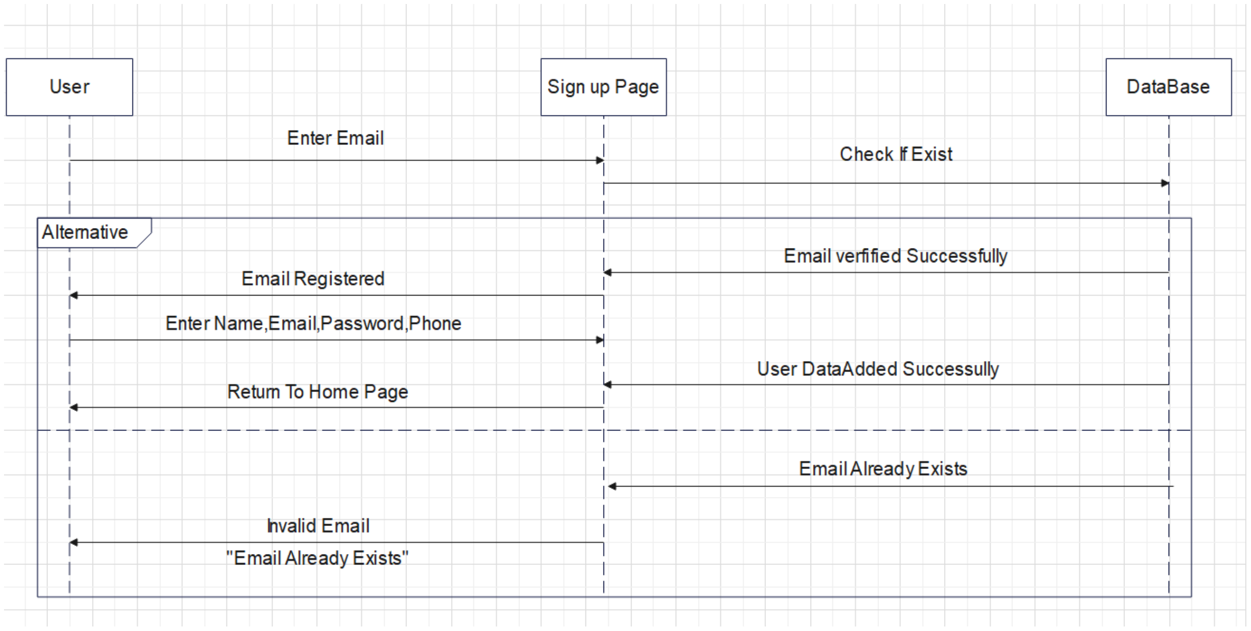
- **Description**: Allows the user to contact the support team.
- **Precondition**: None.
- **Post condition**: User is provided with a form or contact information to reach the support team.

19. Talk to Our Team (extends Contact Us)

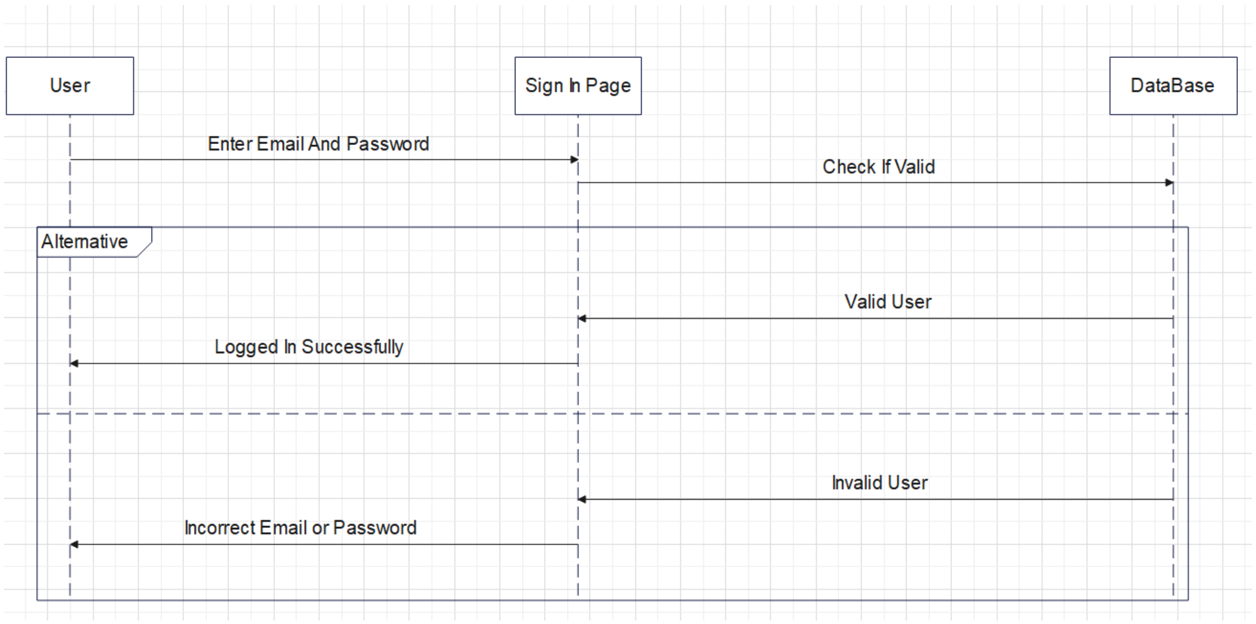
- **Description**: Provides a direct communication channel with the support team.
- **Precondition**: User has chosen to contact the support team.
- **Post condition**: User is able to talk to a support team member.

3.3 Sequence Diagrams

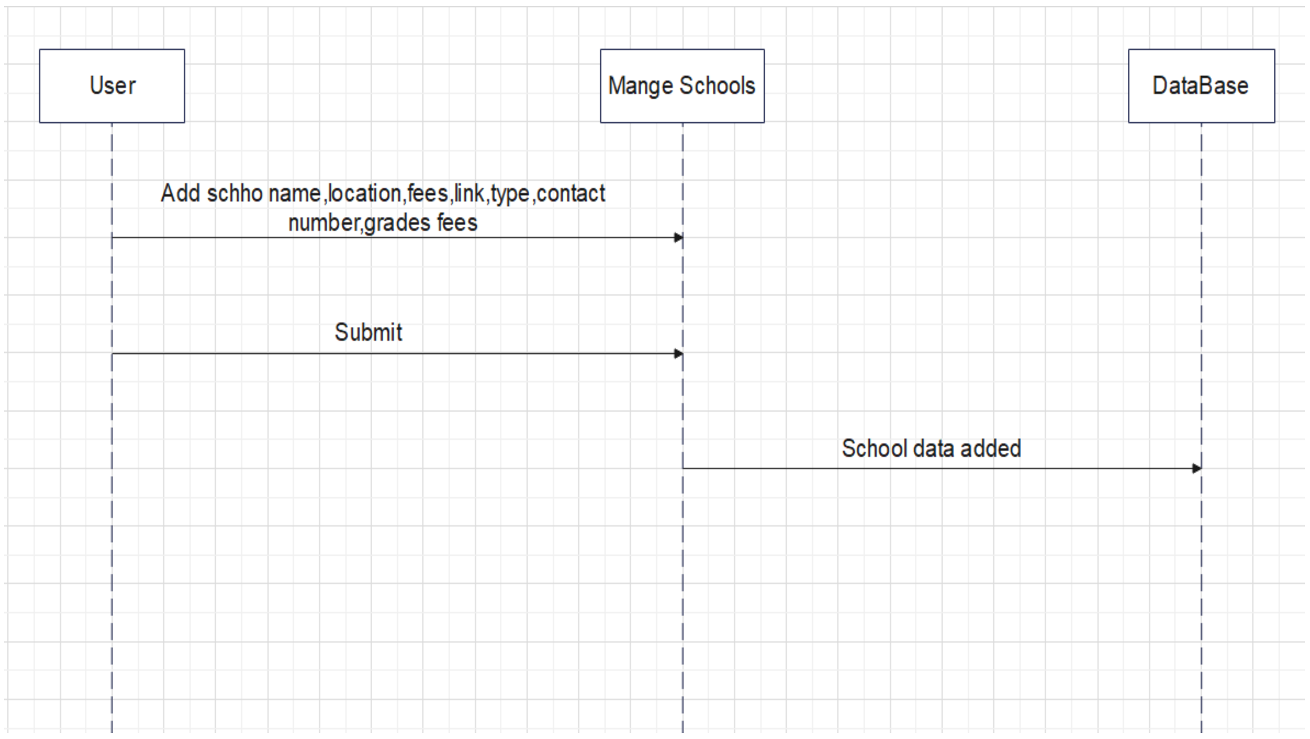
3.3.1 Register



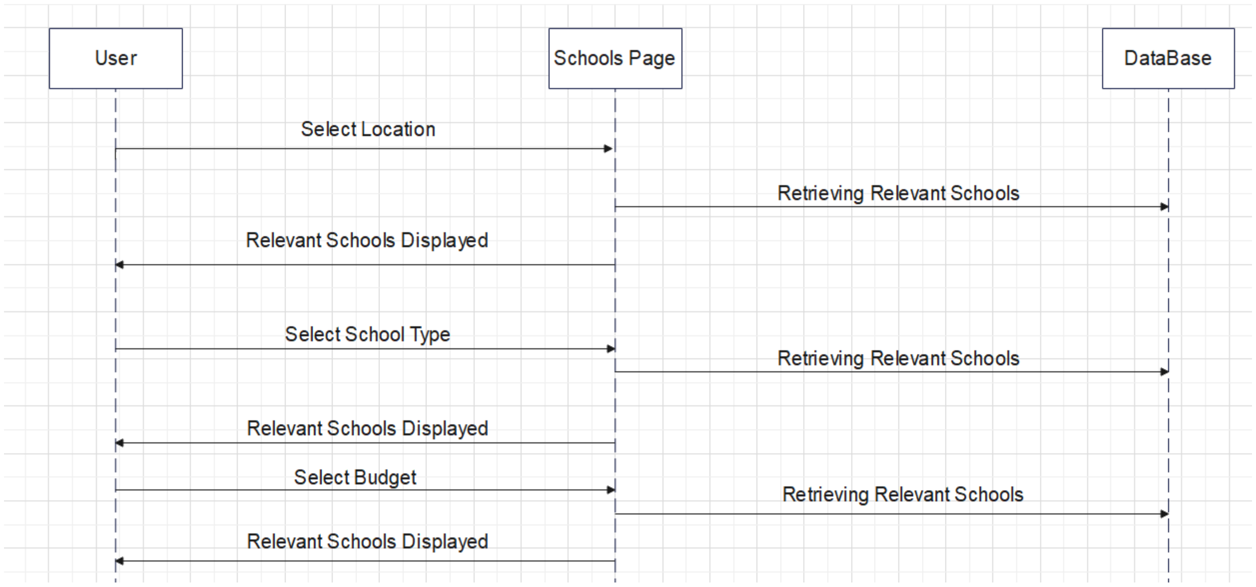
3.3.2 Sign In


User
Sign In Page
DataBase
Enter Email And Password
Check If Valid
Alternative
Valid User
Logged In Successfully
Invalid User
Incorrect Email or Password

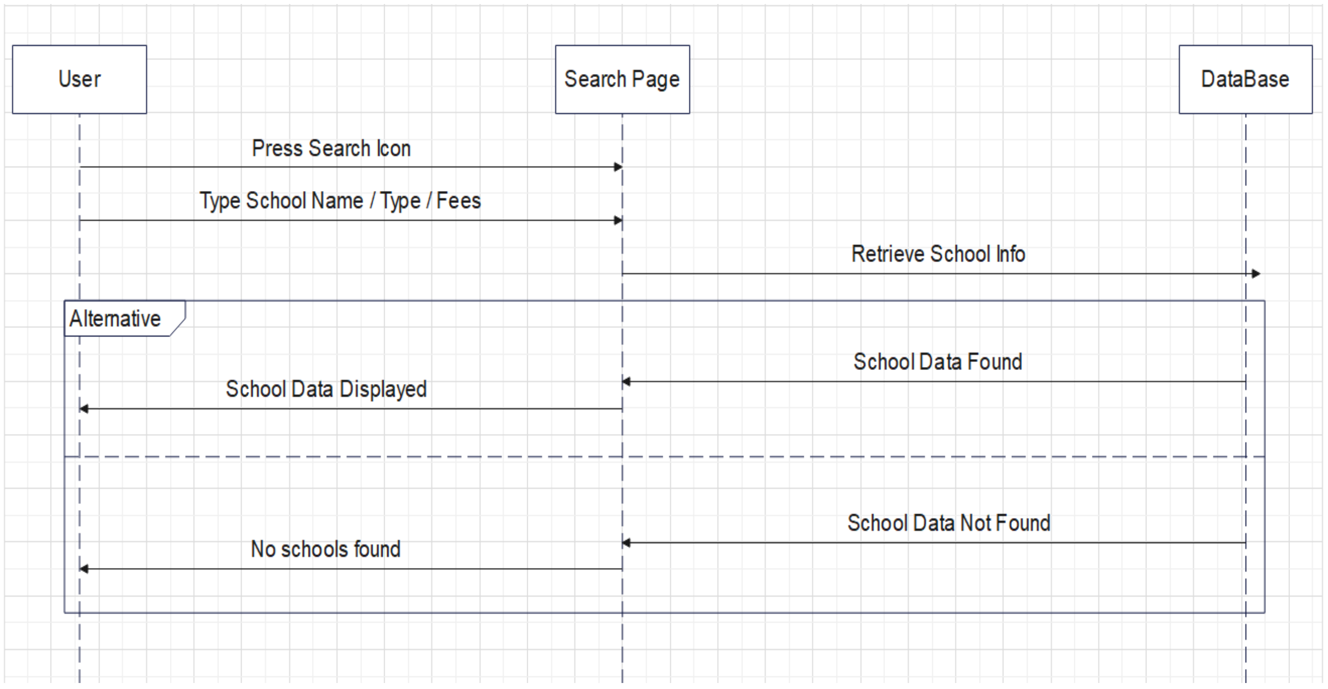
3.3.3 Manage Schools


User
Mange Schools
DataBase
Add schho name,location,fees,link,type,contact number,grades fees
Submit
School data added

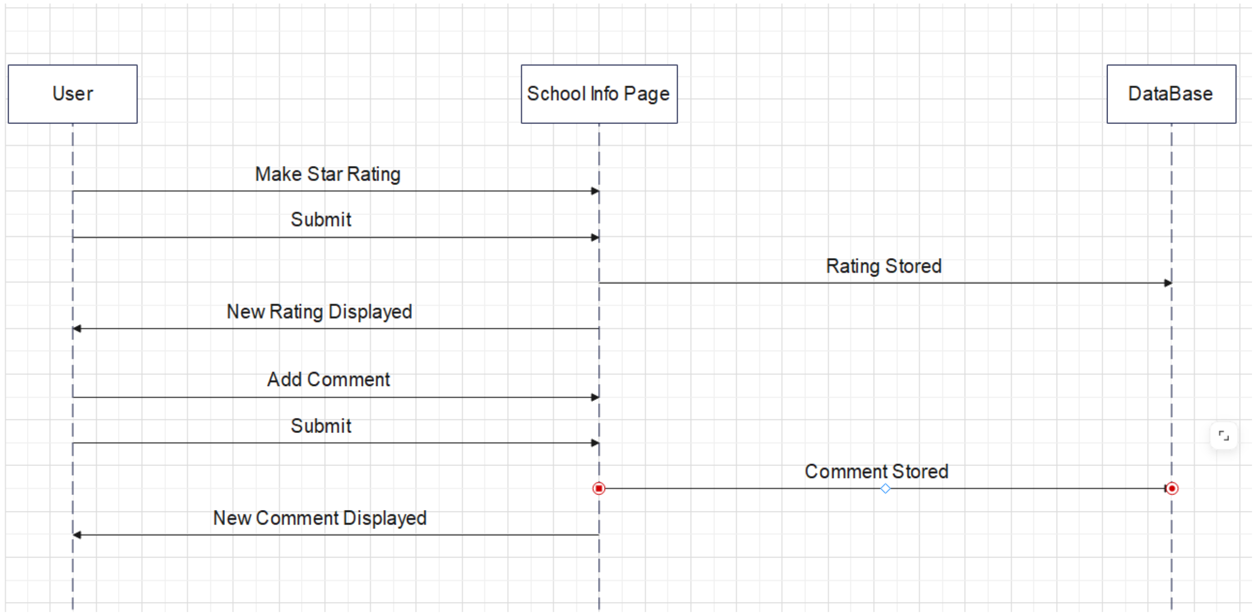
3.3.4 Filter


User
Schools Page
DataBase
Select Location
Retrieving Relevant Schools
Relevant Schools Displayed
Select School Type
Retrieving Relevant Schools
Relevant Schools Displayed
Select Budget
Retrieving Relevant Schools
Relevant Schools Displayed

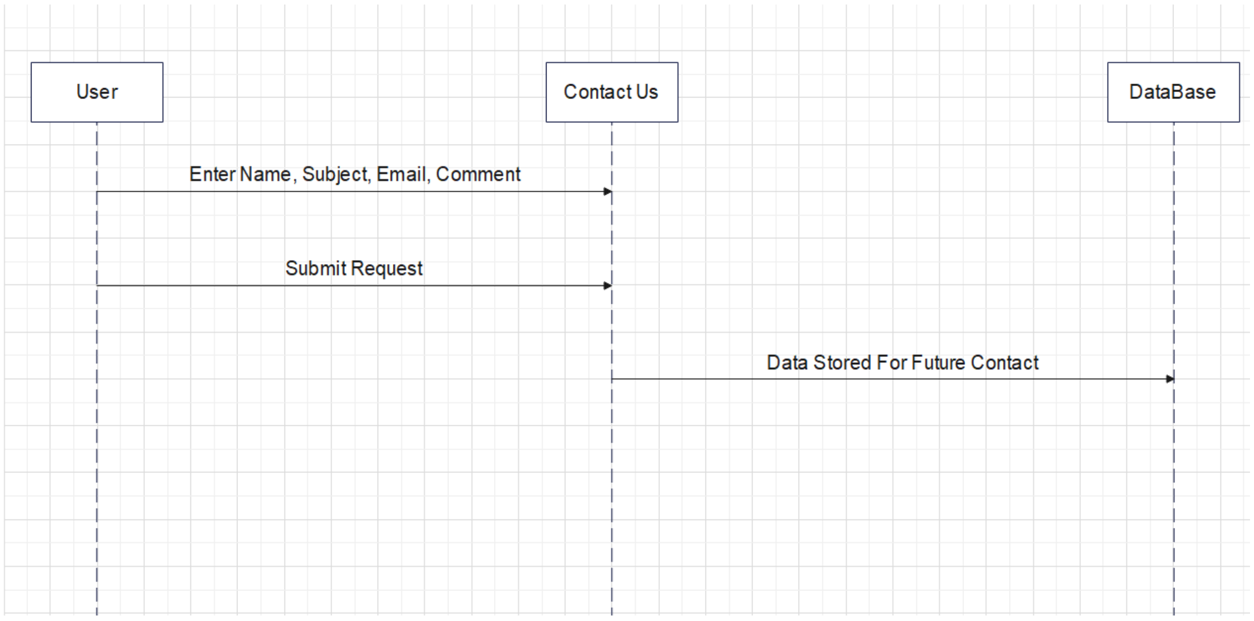
3.3.5 Search School



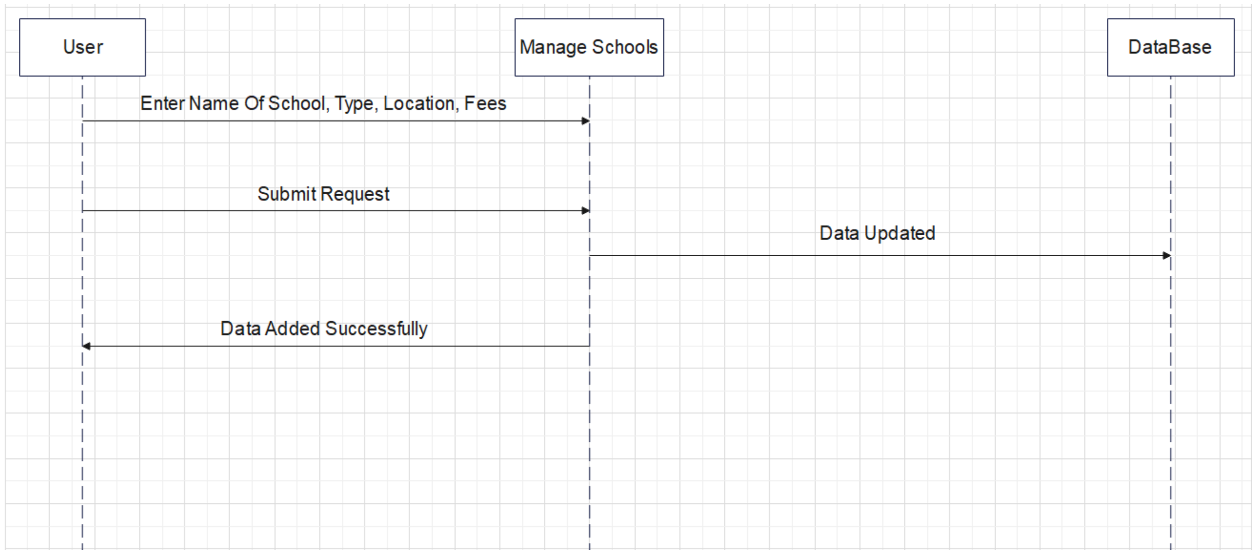
3.3.6 View Details & Rate



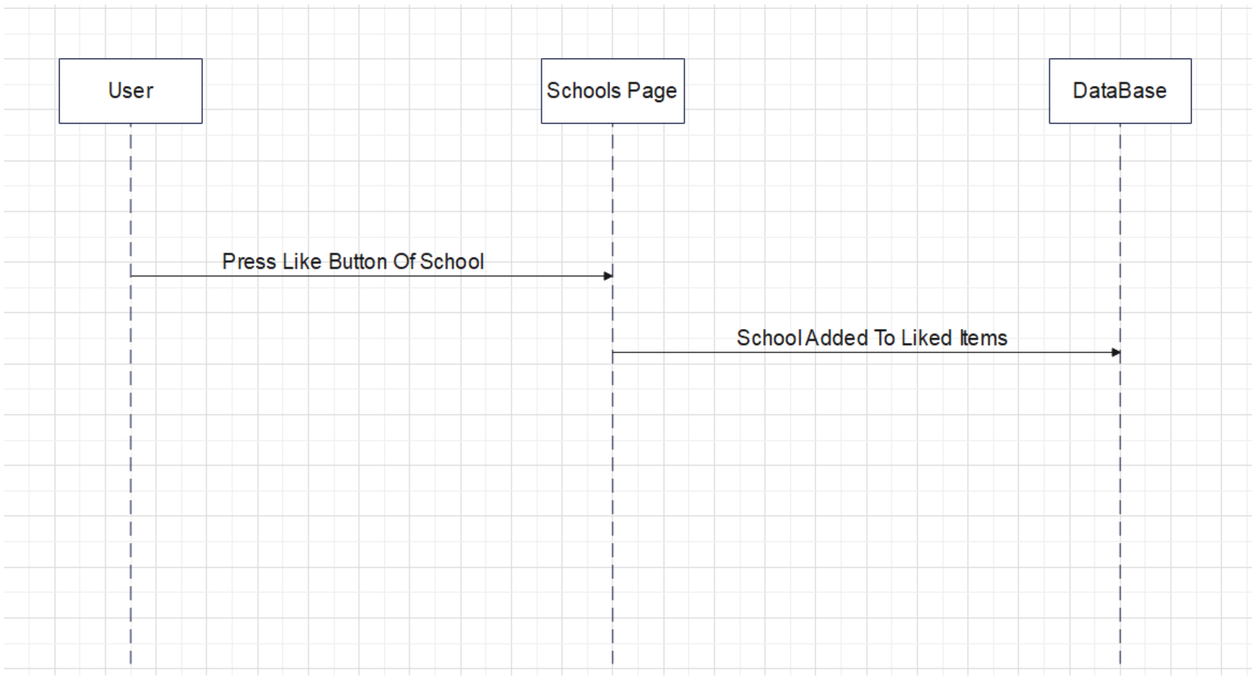
3.3.7 Talk To Our Team


User
Contact Us
DataBase
Enter Name, Subject, Email, Comment
Submit Request
Data Stored For Future Contact

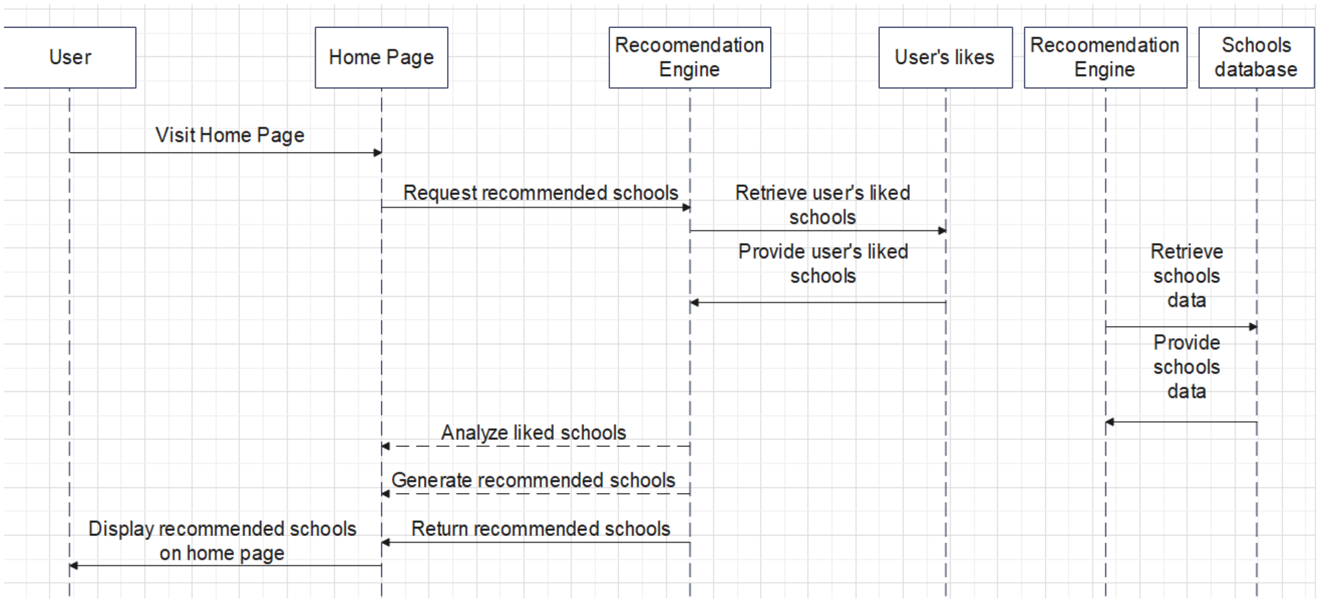
3.3.8 Manage Schools



3.3.9 Liked Schools



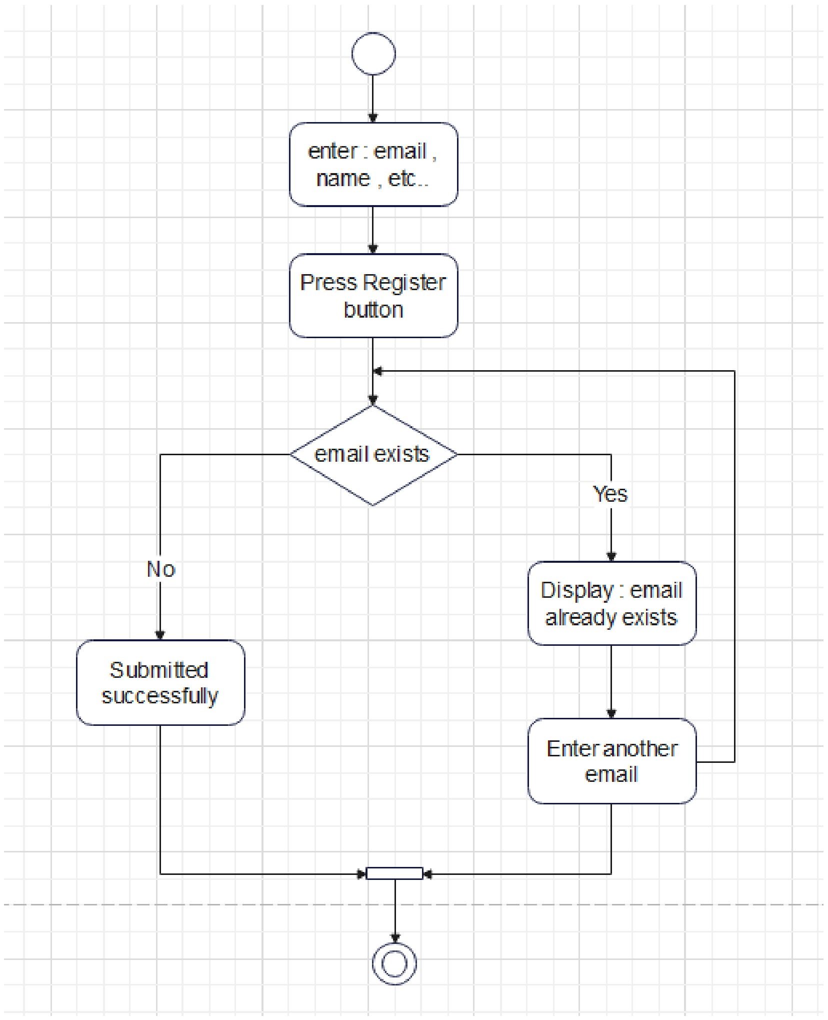
3.3.10 Recommendation system



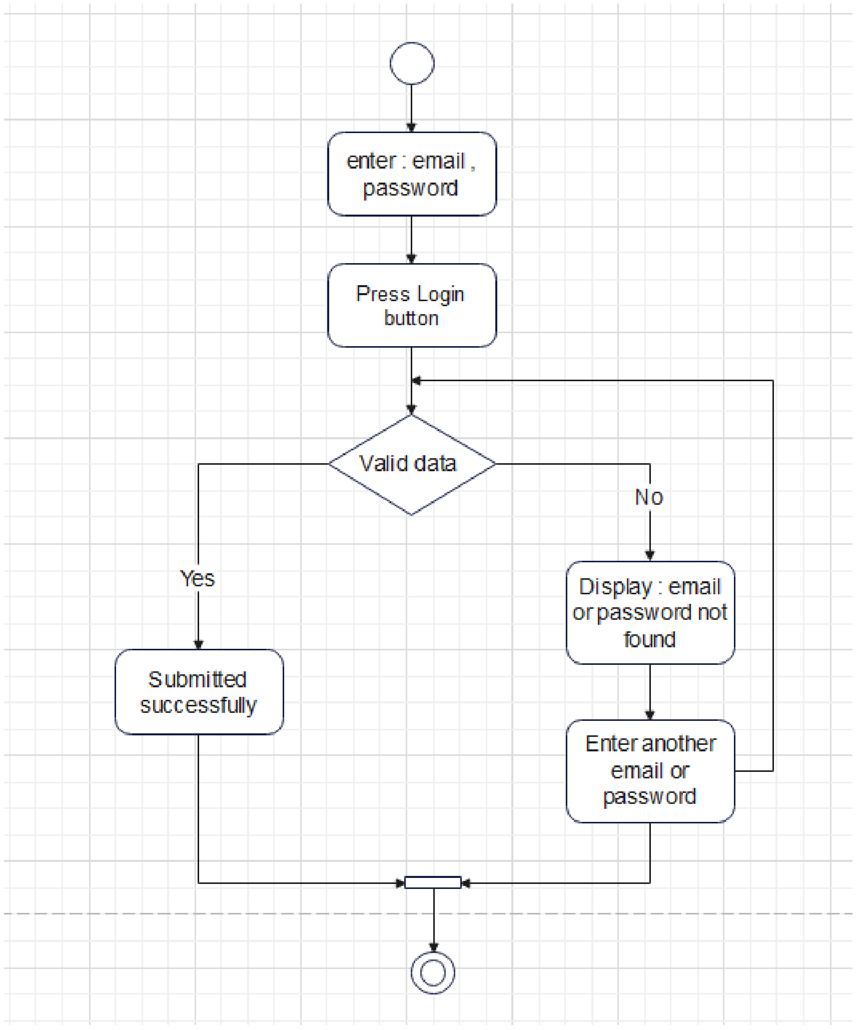
User
Home Page
Recoomendation Engine
User's likes
Recoomendation Engine
Schools database
Visit Home Page
Request recommended schools
Retrieve user's liked schools
Provide user's liked schools
Retrieve schools data
Provide schools data
Analyze liked schools
Generate recommended schools
Display recommended schools on home page
Return recommended schools

3.4 Activity Diagrams

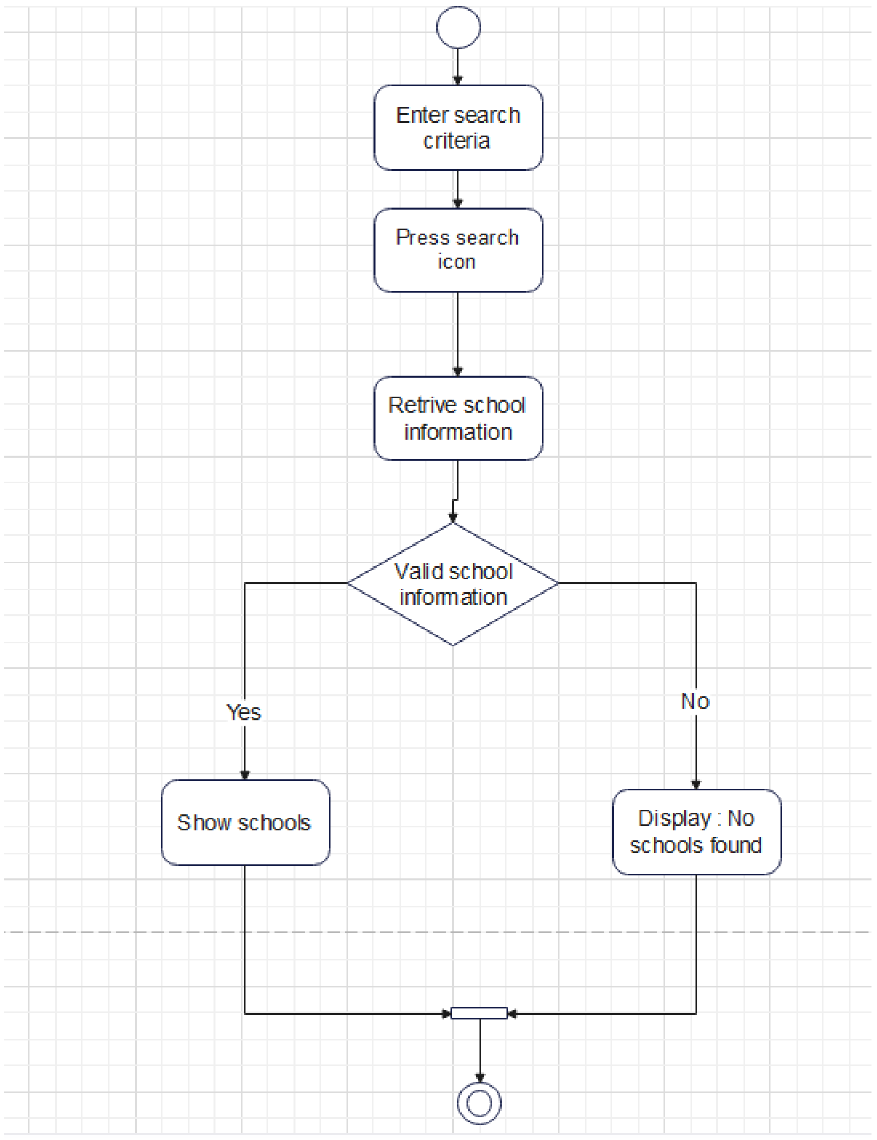
3.4.1 Register



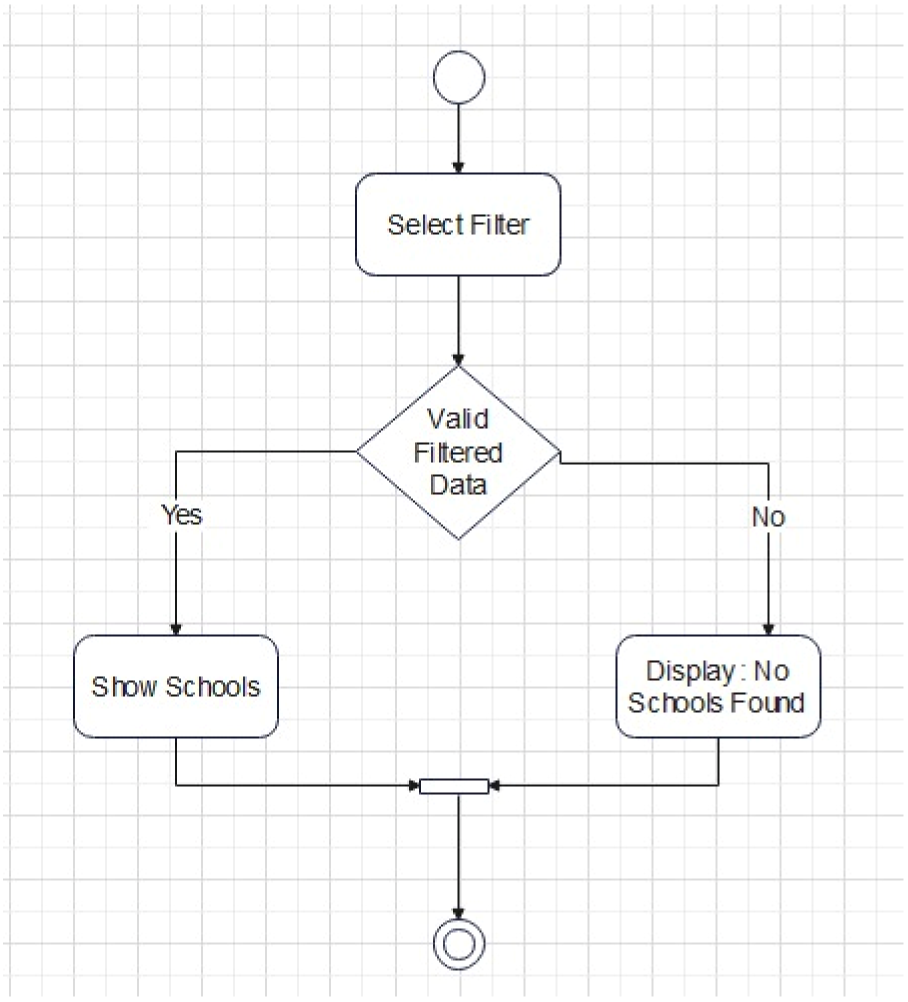
3.4.2 Log In


enter : email , password
Press Login button
Valid data
No
Yes
Display : email or password not found
Submitted successfully
Enter another email or password

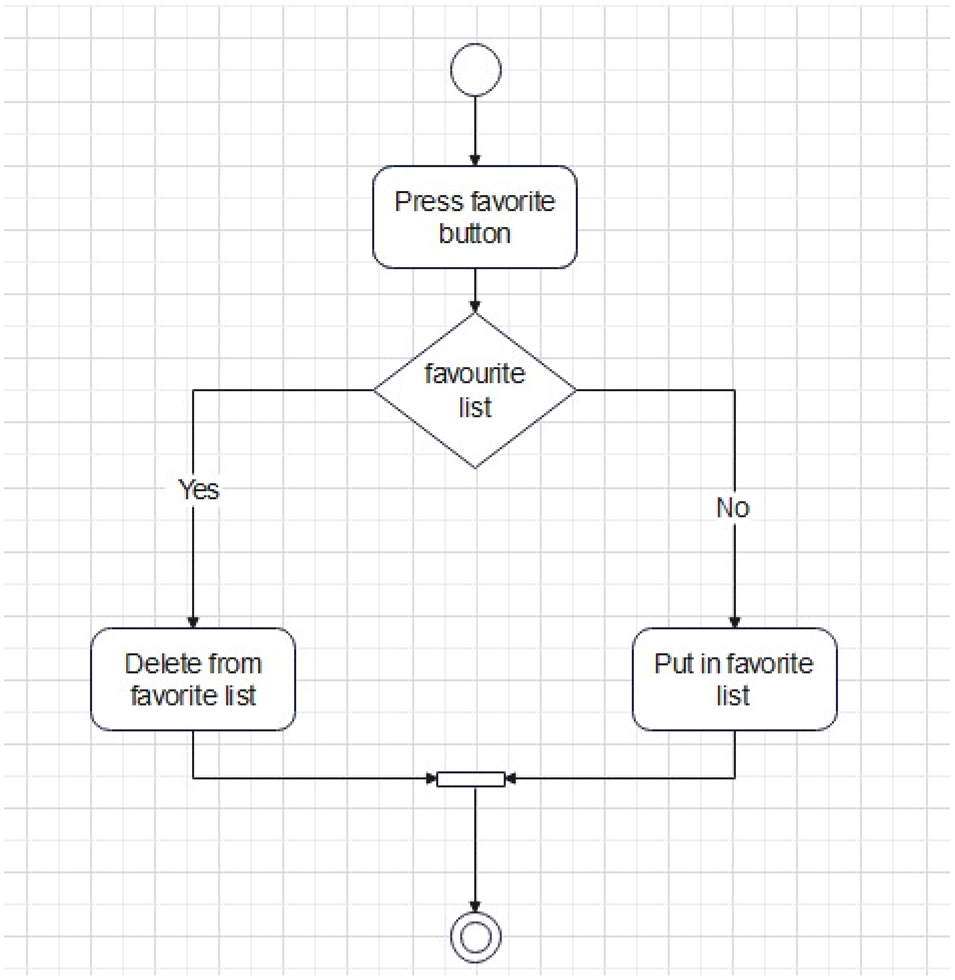
3.4.3 Search


Enter search criteria
Press search icon
Retrive school information
Valid school information
Yes
No
Show schools
Display : No schools found

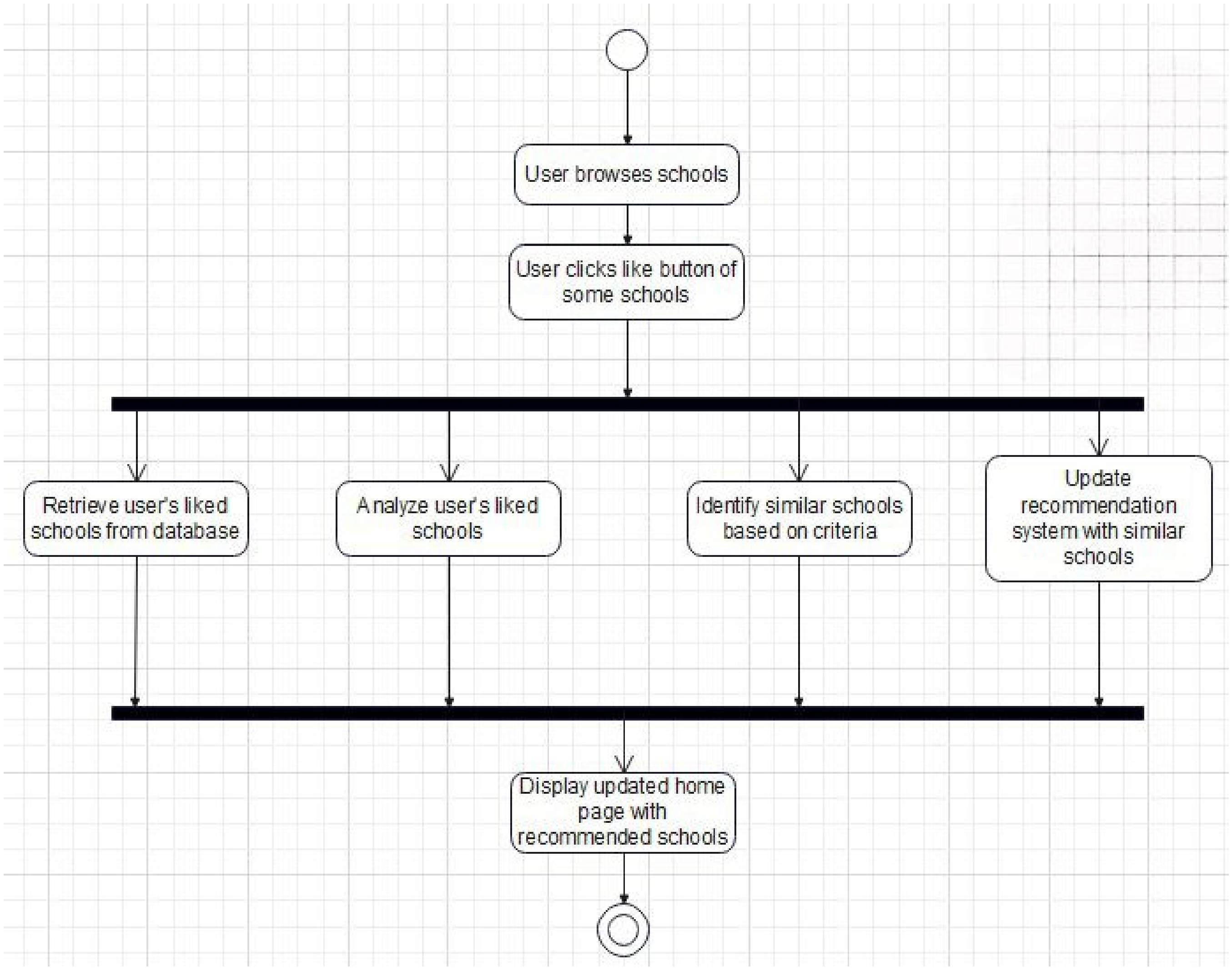
3.4.4 Filter


Select Filter
Valid Filtered Data
Yes
No
Show Schools
Display: No Schools Found

3.4.5 Favorites



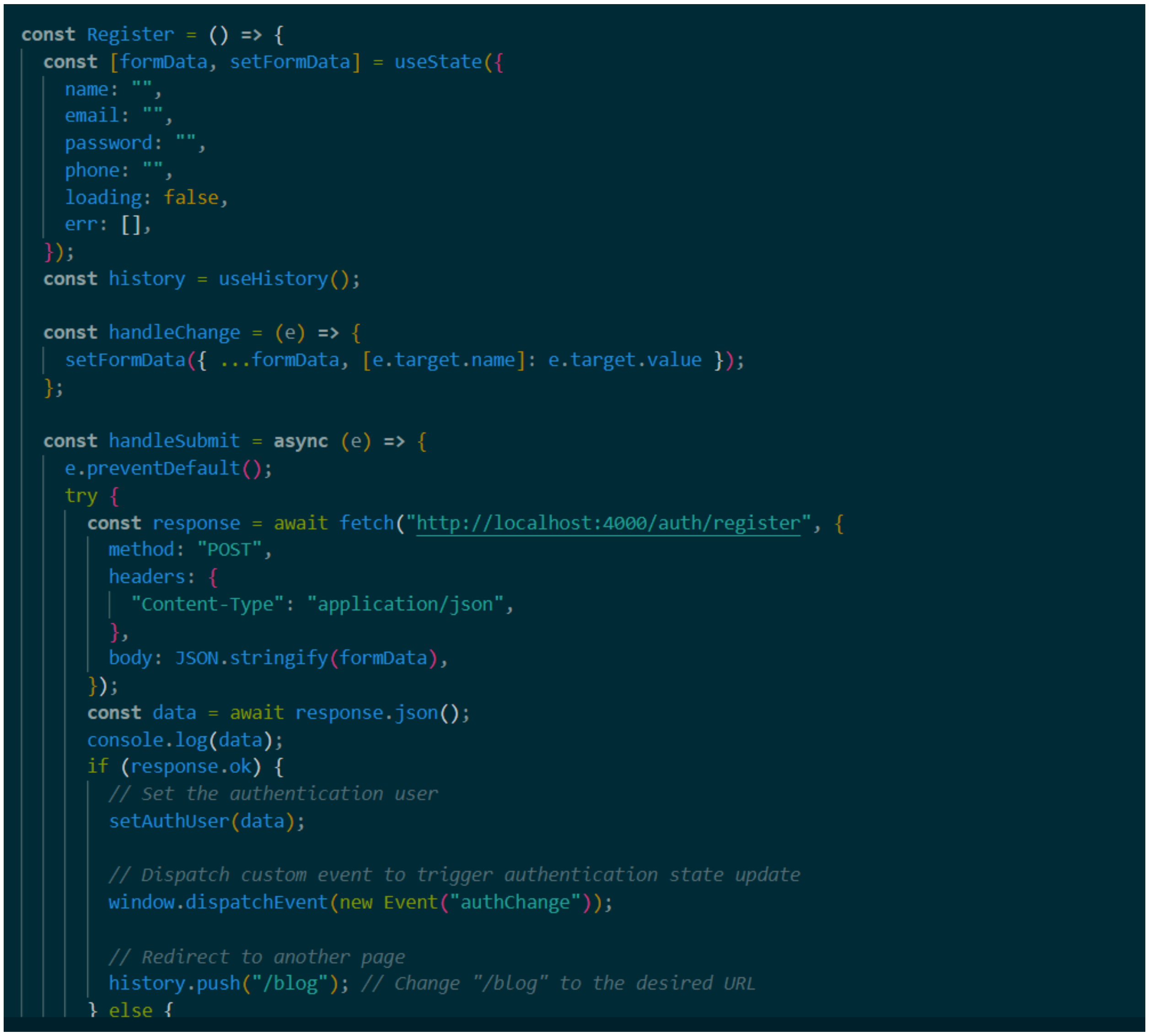
3.4.6 Recommended Schools


User browses schools
User clicks like button of some schools
Retrieve user's liked schools from database
Analyze user's liked schools
Identify similar schools based on criteria
Update recommendation system with similar schools
Display updated home page with recommended schools

Chapter4
implementation

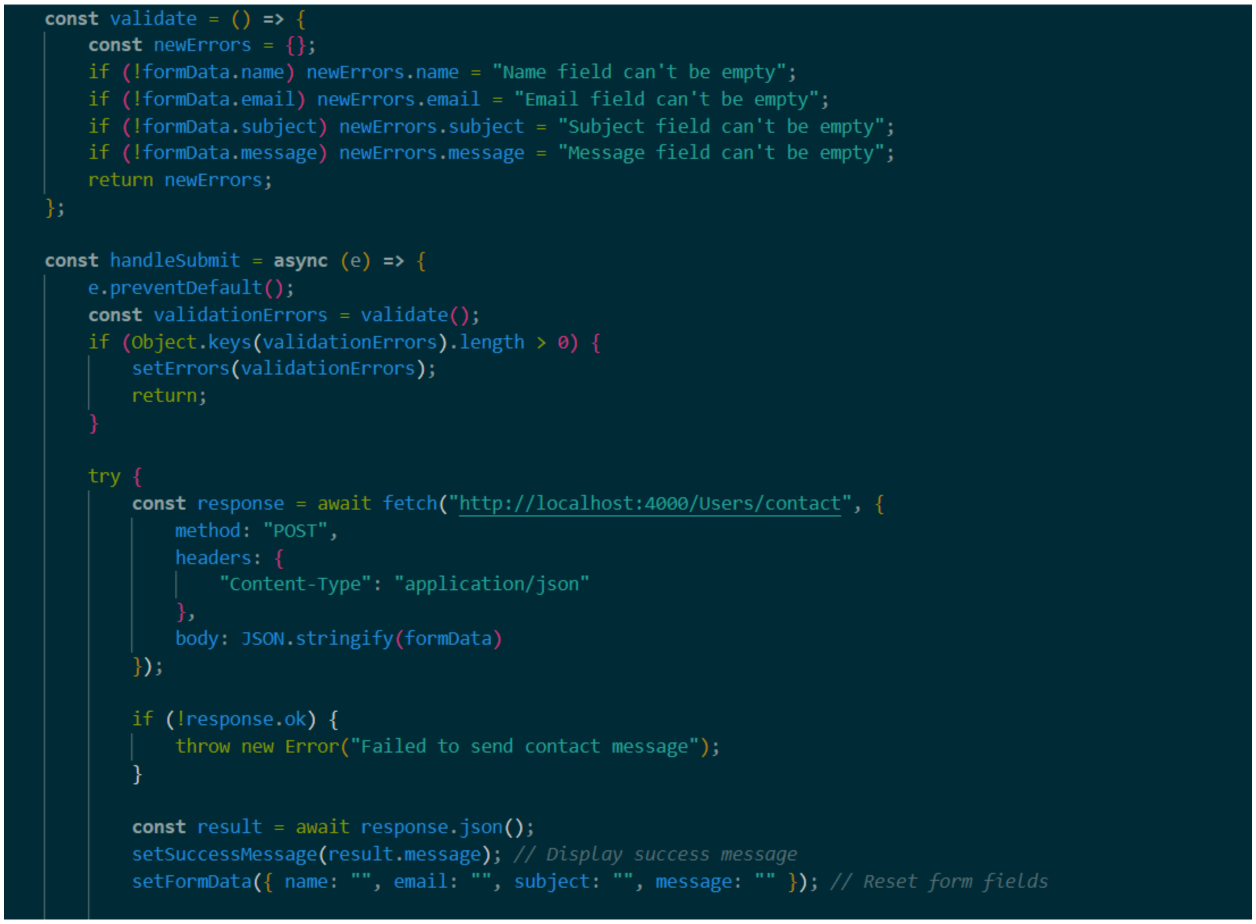
Front end

4.1.1 user register



```
const Register = () => {
  const [formData, setFormData] = useState({
    name: "",
    email: "",
    password: "",
    phone: "",
    loading: false,
    err: [],
  });
  const history = useHistory();

  const handleChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      const response = await fetch("http://localhost:4000/auth/register", {
        method: "POST",
        headers: {
          "Content-Type": "application/json",
        },
        body: JSON.stringify(formData),
      });
      const data = await response.json();
      console.log(data);
      if (response.ok) {
        // Set the authentication user
        setAuthUser(data);

        // Dispatch custom event to trigger authentication state update
        window.dispatchEvent(new Event("authChange"));

        // Redirect to another page
        history.push("/blog"); // Change "/blog" to the desired URL
      } else {
```

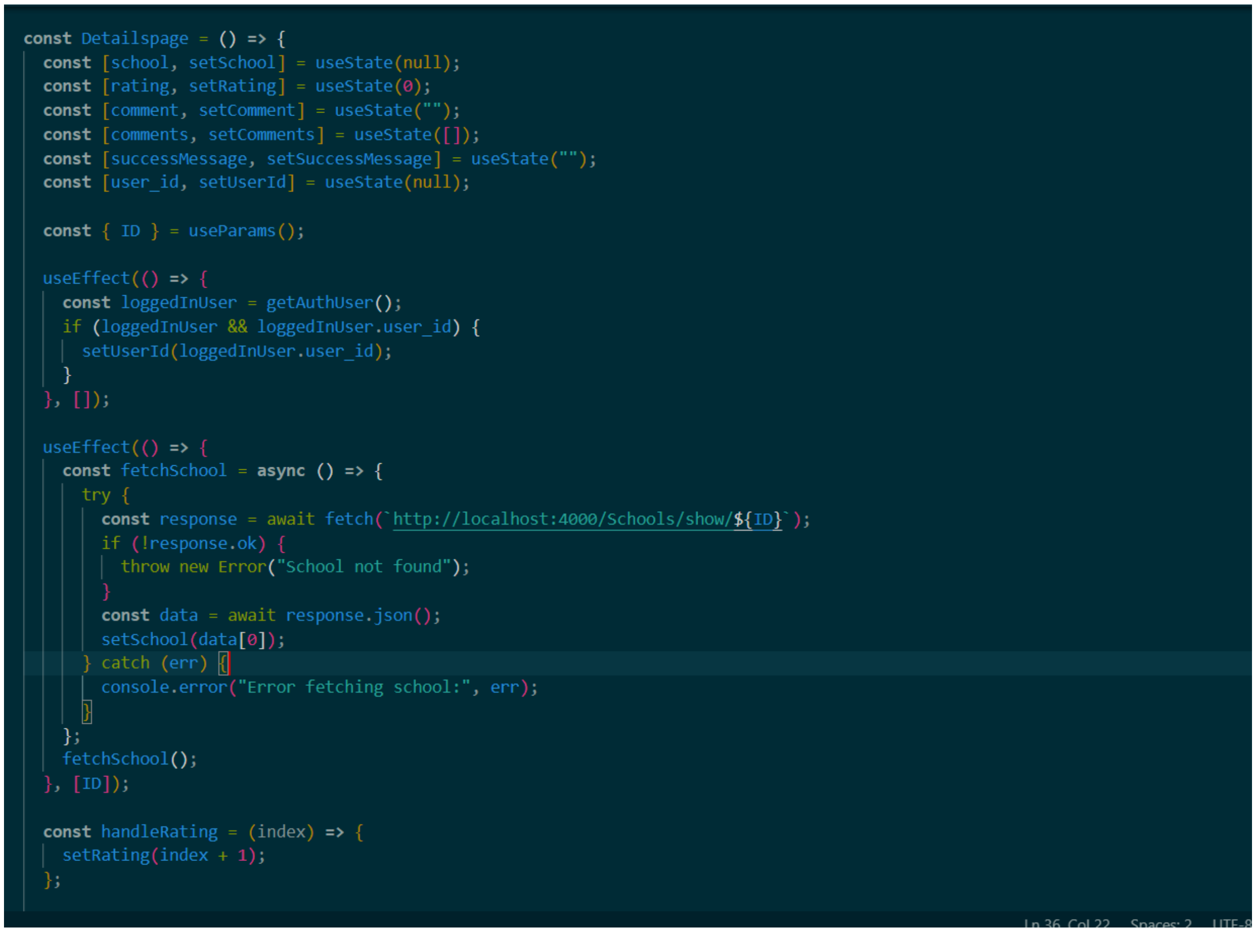
4.1.2 Contact us



```javascript
const validate = () => {
    const newErrors = {};
    if (!formData.name) newErrors.name = "Name field can't be empty";
    if (!formData.email) newErrors.email = "Email field can't be empty";
    if (!formData.subject) newErrors.subject = "Subject field can't be empty";
    if (!formData.message) newErrors.message = "Message field can't be empty";
    return newErrors;
};

const handleSubmit = async (e) => {
    e.preventDefault();
    const validationErrors = validate();
    if (Object.keys(validationErrors).length > 0) {
        setErrors(validationErrors);
        return;
    }

    try {
        const response = await fetch("http://localhost:4000/Users/contact", {
            method: "POST",
            headers: {
                "Content-Type": "application/json"
            },
            body: JSON.stringify(formData)
        });

        if (!response.ok) {
            throw new Error("Failed to send contact message");
        }

        const result = await response.json();
        setSuccessMessage(result.message); // Display success message
        setFormData({ name: "", email: "", subject: "", message: "" }); // Reset form fields
```

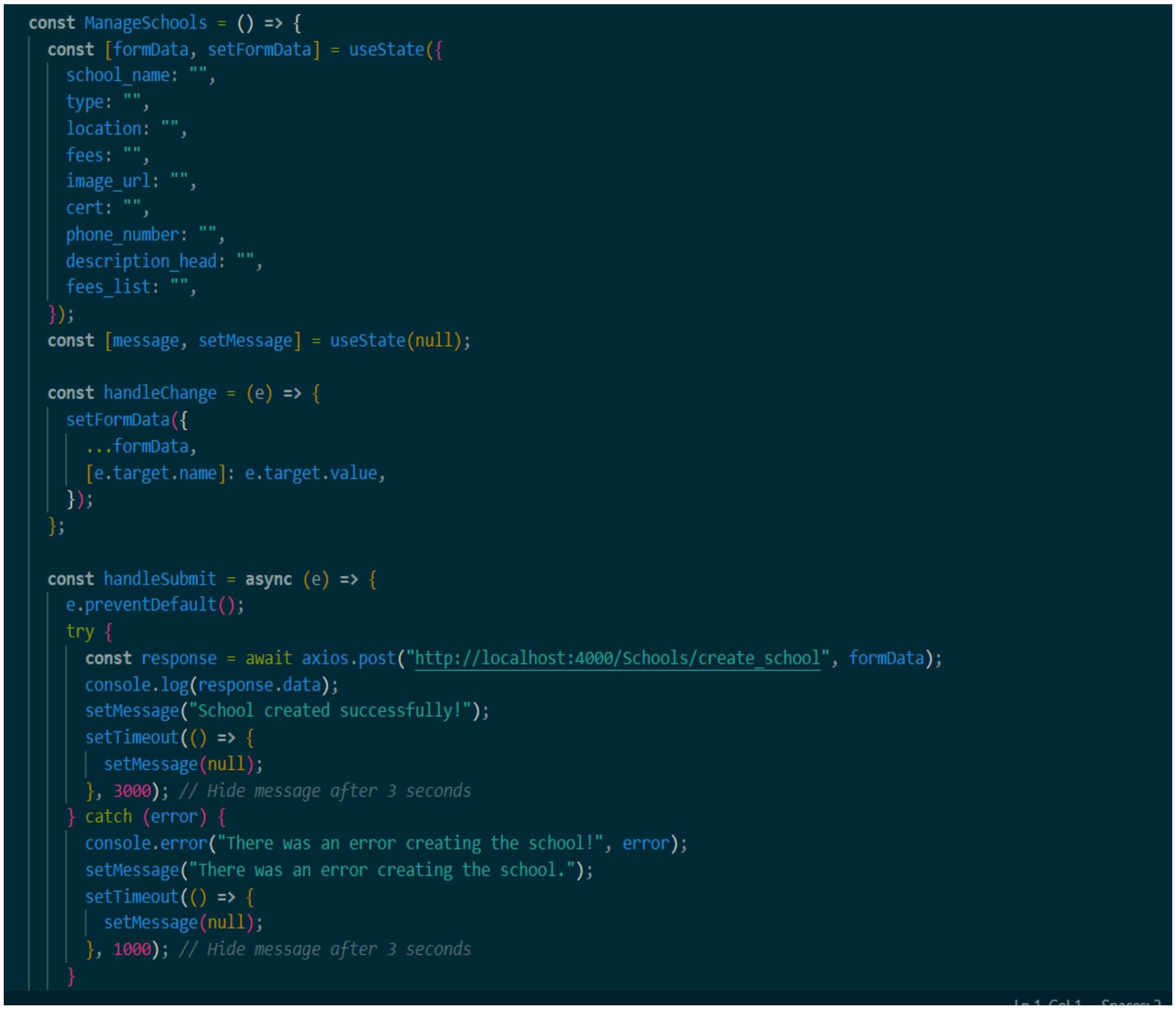
4.1.3 Details page

Details page component is designed to display details of a specific school, it retrieves the school ID from the URL and fetches school details from the server and manages user input for ratings and comments.



```
const Detailspage = () => {
  const [school, setSchool] = useState(null);
  const [rating, setRating] = useState(0);
  const [comment, setComment] = useState("");
  const [comments, setComments] = useState([]);
  const [successMessage, setSuccessMessage] = useState("");
  const [user_id, setUserId] = useState(null);

  const { ID } = useParams();

  useEffect(() => {
    const loggedInUser = getAuthUser();
    if (loggedInUser && loggedInUser.user_id) {
      setUserId(loggedInUser.user_id);
    }
  }, []);

  useEffect(() => {
    const fetchSchool = async () => {
      try {
        const response = await fetch(`http://localhost:4000/Schools/show/${ID}`);
        if (!response.ok) {
          throw new Error("School not found");
        }
        const data = await response.json();
        setSchool(data[0]);
      } catch (err) {
        console.error("Error fetching school:", err);
      }
    };
    fetchSchool();
  }, [ID]);

  const handleRating = (index) => {
    setRating(index + 1);
  };
```

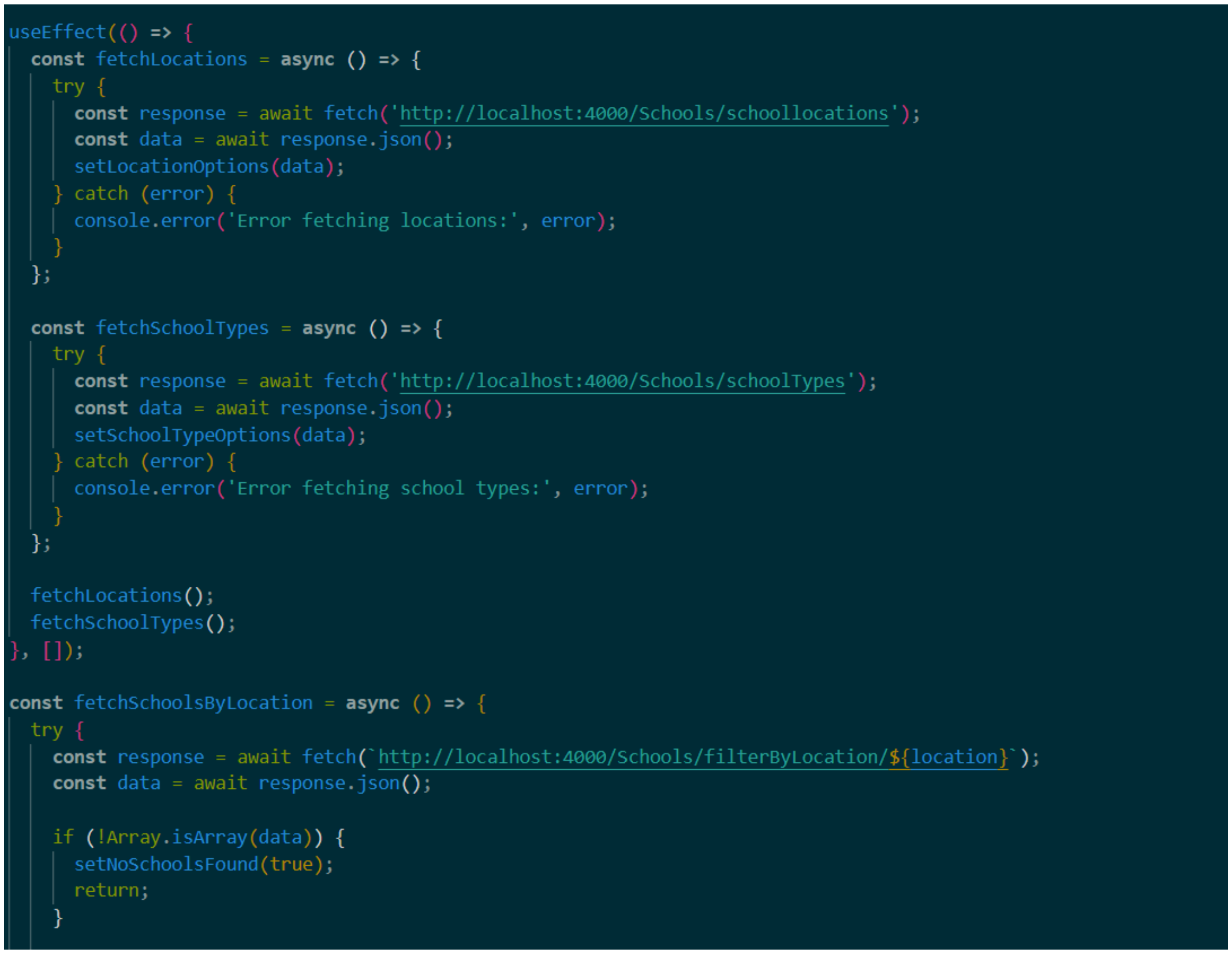
4.1.4 manage schools



```
const ManageSchools = () => {
  const [formData, setFormData] = useState({
    school_name: "",
    type: "",
    location: "",
    fees: "",
    image_url: "",
    cert: "",
    phone_number: "",
    description_head: "",
    fees_list: "",
  });
  const [message, setMessage] = useState(null);

  const handleChange = (e) => {
    setFormData({
      ...formData,
      [e.target.name]: e.target.value,
    });
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      const response = await axios.post("http://localhost:4000/Schools/create_school", formData);
      console.log(response.data);
      setMessage("School created successfully!");
      setTimeout(() => {
        setMessage(null);
      }, 3000); // Hide message after 3 seconds
    } catch (error) {
      console.error("There was an error creating the school!", error);
      setMessage("There was an error creating the school.");
      setTimeout(() => {
        setMessage(null);
      }, 1000); // Hide message after 3 seconds
    }
```

4.1.5 Filter by location

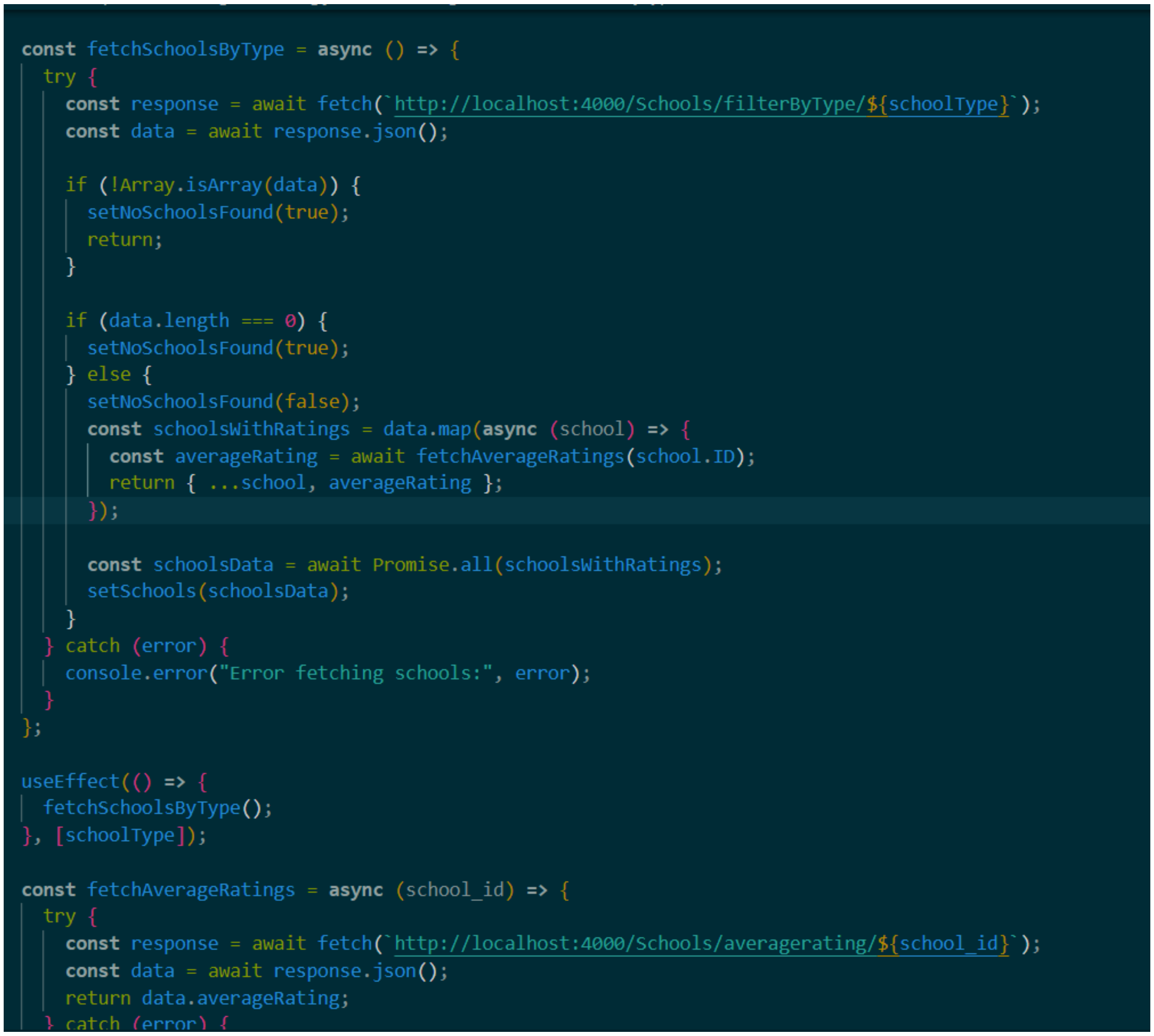
Fetches schools based on the selected location.
Data Fetching: Sends a request to the server to get schools filtered by the specified location.



```
useEffect(() => {
  const fetchLocations = async () => {
    try {
      const response = await fetch('http://localhost:4000/Schools/schoollocations');
      const data = await response.json();
      setLocationOptions(data);
    } catch (error) {
      console.error('Error fetching locations:', error);
    }
  };

  const fetchSchoolTypes = async () => {
    try {
      const response = await fetch('http://localhost:4000/Schools/schoolTypes');
      const data = await response.json();
      setSchoolTypeOptions(data);
    } catch (error) {
      console.error('Error fetching school types:', error);
    }
  };

  fetchLocations();
  fetchSchoolTypes();
}, []);

const fetchSchoolsByLocation = async () => {
  try {
    const response = await fetch(`http://localhost:4000/Schools/filterByLocation/${location}`);
    const data = await response.json();

    if (!Array.isArray(data)) {
      setNoSchoolsFound(true);
      return;
    }
```

4.1.6 Filter by type

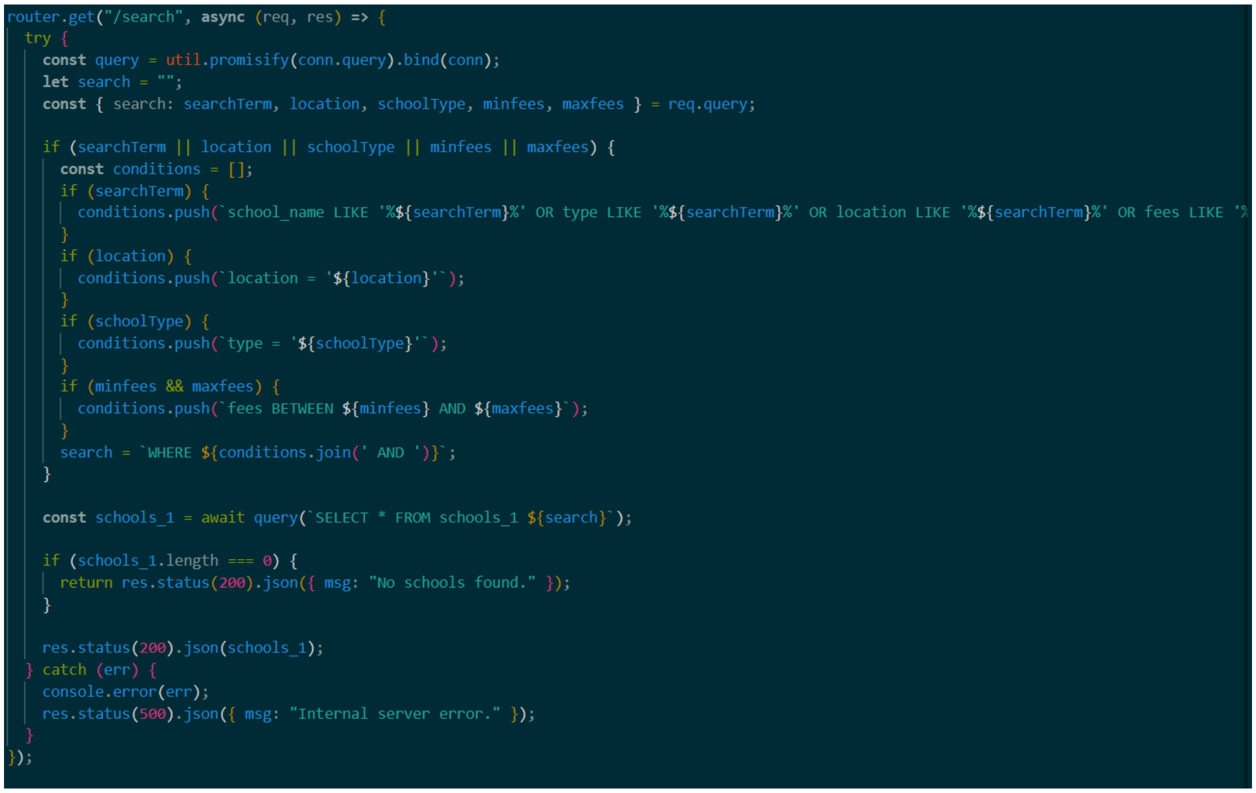
Fetches schools based on the selected type.
Data Fetching: Sends a request to the server to get schools filtered by the specified type



```
const fetchSchoolsByType = async () => {
  try {
    const response = await fetch(`http://localhost:4000/Schools/filterByType/${schoolType}`);
    const data = await response.json();

    if (!Array.isArray(data)) {
      setNoSchoolsFound(true);
      return;
    }

    if (data.length === 0) {
      setNoSchoolsFound(true);
    } else {
      setNoSchoolsFound(false);
      const schoolsWithRatings = data.map(async (school) => {
        const averageRating = await fetchAverageRatings(school.ID);
        return { ...school, averageRating };
      });

      const schoolsData = await Promise.all(schoolsWithRatings);
      setSchools(schoolsData);
    }
  } catch (error) {
    console.error("Error fetching schools:", error);
  }
};

useEffect(() => {
  fetchSchoolsByType();
}, [schoolType]);

const fetchAverageRatings = async (school_id) => {
  try {
    const response = await fetch(`http://localhost:4000/Schools/averagerating/${school_id}`);
    const data = await response.json();
    return data.averageRating;
  } catch (error) {
```

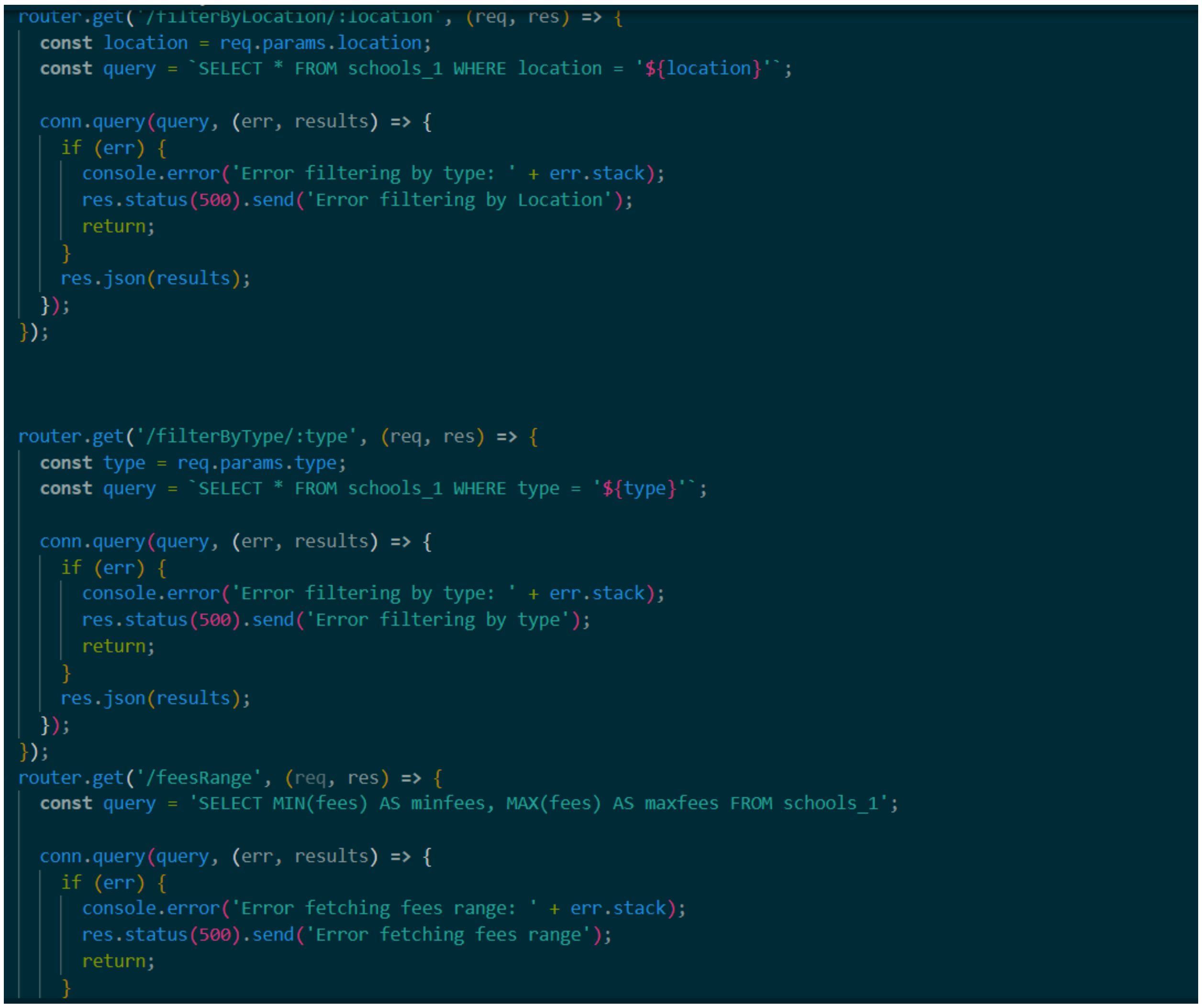
Back end

4.1.7 Search school



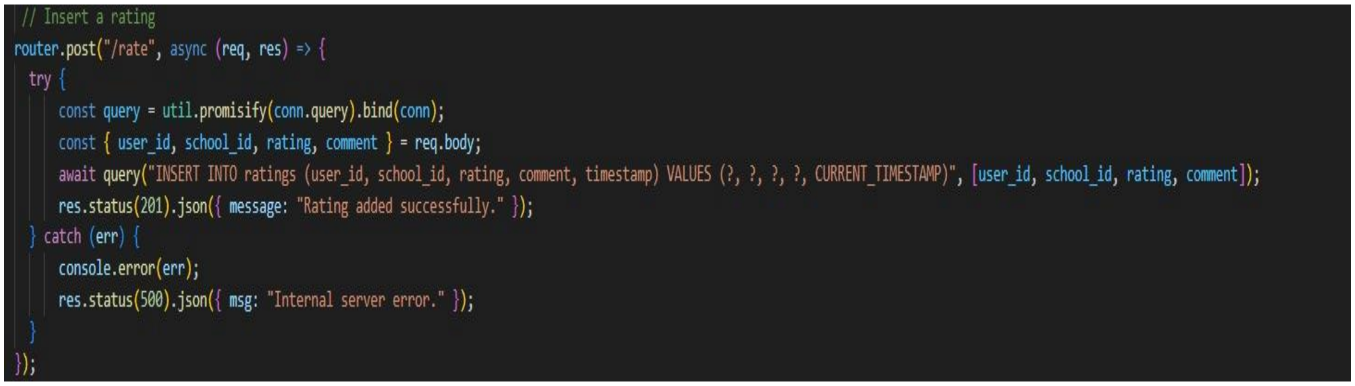
```
router.get("/search", async (req, res) => {
  try {
    const query = util.promisify(conn.query).bind(conn);
    let search = "";
    const { search: searchTerm, location, schoolType, minfees, maxfees } = req.query;

    if (searchTerm || location || schoolType || minfees || maxfees) {
      const conditions = [];
      if (searchTerm) {
        conditions.push(`school_name LIKE '%${searchTerm}%' OR type LIKE '%${searchTerm}%' OR location LIKE '%${searchTerm}%' OR fees LIKE '%
      }
      if (location) {
        conditions.push(`location = '${location}'`);
      }
      if (schoolType) {
        conditions.push(`type = '${schoolType}'`);
      }
      if (minfees && maxfees) {
        conditions.push(`fees BETWEEN ${minfees} AND ${maxfees}`);
      }
      search = `WHERE ${conditions.join(' AND ')}`;
    }

    const schools_1 = await query(`SELECT * FROM schools_1 ${search}`);

    if (schools_1.length === 0) {
      return res.status(200).json({ msg: "No schools found." });
    }

    res.status(200).json(schools_1);
  } catch (err) {
    console.error(err);
    res.status(500).json({ msg: "Internal server error." });
  }
});
```

4.1.8 Filter school by location and fees and type



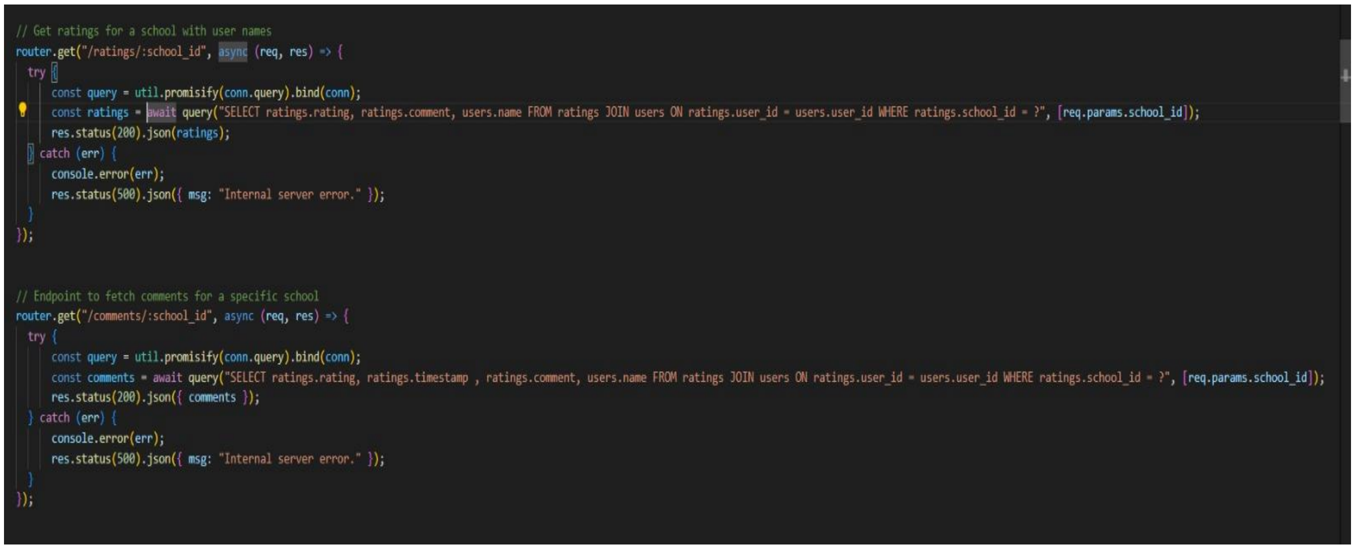
```
router.get('/filterByLocation/:location', (req, res) => {
  const location = req.params.location;
  const query = `SELECT * FROM schools_1 WHERE location = '${location}'`;

  conn.query(query, (err, results) => {
    if (err) {
      console.error('Error filtering by type: ' + err.stack);
      res.status(500).send('Error filtering by Location');
      return;
    }
    res.json(results);
  });
});



router.get('/filterByType/:type', (req, res) => {
  const type = req.params.type;
  const query = `SELECT * FROM schools_1 WHERE type = '${type}'`;

  conn.query(query, (err, results) => {
    if (err) {
      console.error('Error filtering by type: ' + err.stack);
      res.status(500).send('Error filtering by type');
      return;
    }
    res.json(results);
  });
});
router.get('/feesRange', (req, res) => {
  const query = 'SELECT MIN(fees) AS minfees, MAX(fees) AS maxfees FROM schools_1';

  conn.query(query, (err, results) => {
    if (err) {
      console.error('Error fetching fees range: ' + err.stack);
      res.status(500).send('Error fetching fees range');
      return;
    }
```

4.1.9 Insert Rating and Comment



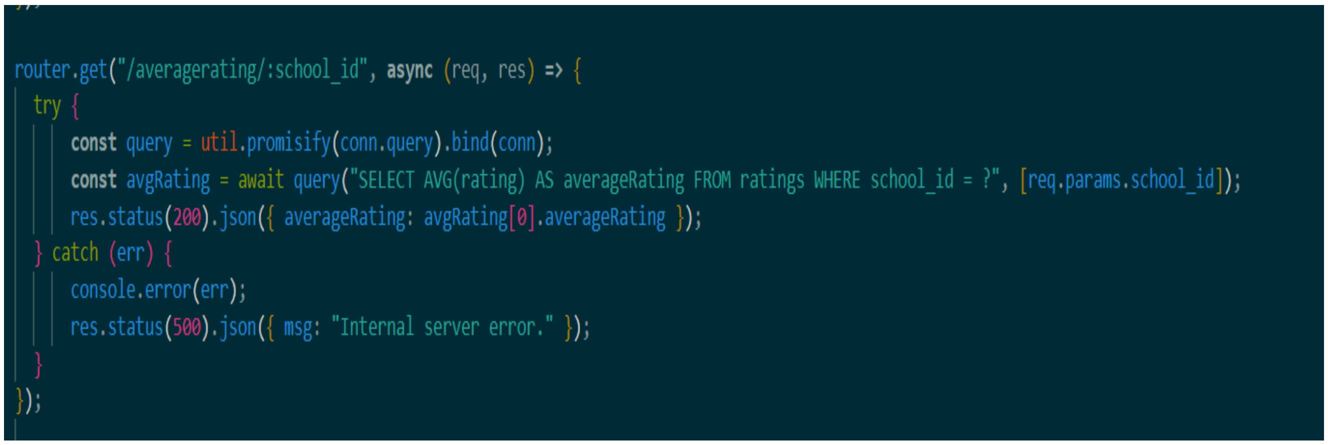
```javascript
// Insert a rating
router.post("/rate", async (req, res) => {
  try {
    const query = util.promisify(conn.query).bind(conn);
    const { user_id, school_id, rating, comment } = req.body;
    await query("INSERT INTO ratings (user_id, school_id, rating, comment, timestamp) VALUES (?, ?, ?, ?, CURRENT_TIMESTAMP)", [user_id, school_id, rating, comment]);
    res.status(201).json({ message: "Rating added successfully." });
  } catch (err) {
    console.error(err);
    res.status(500).json({ msg: "Internal server error." });
  }
});
```

4.1.10 Retrieve Rate and Comment:



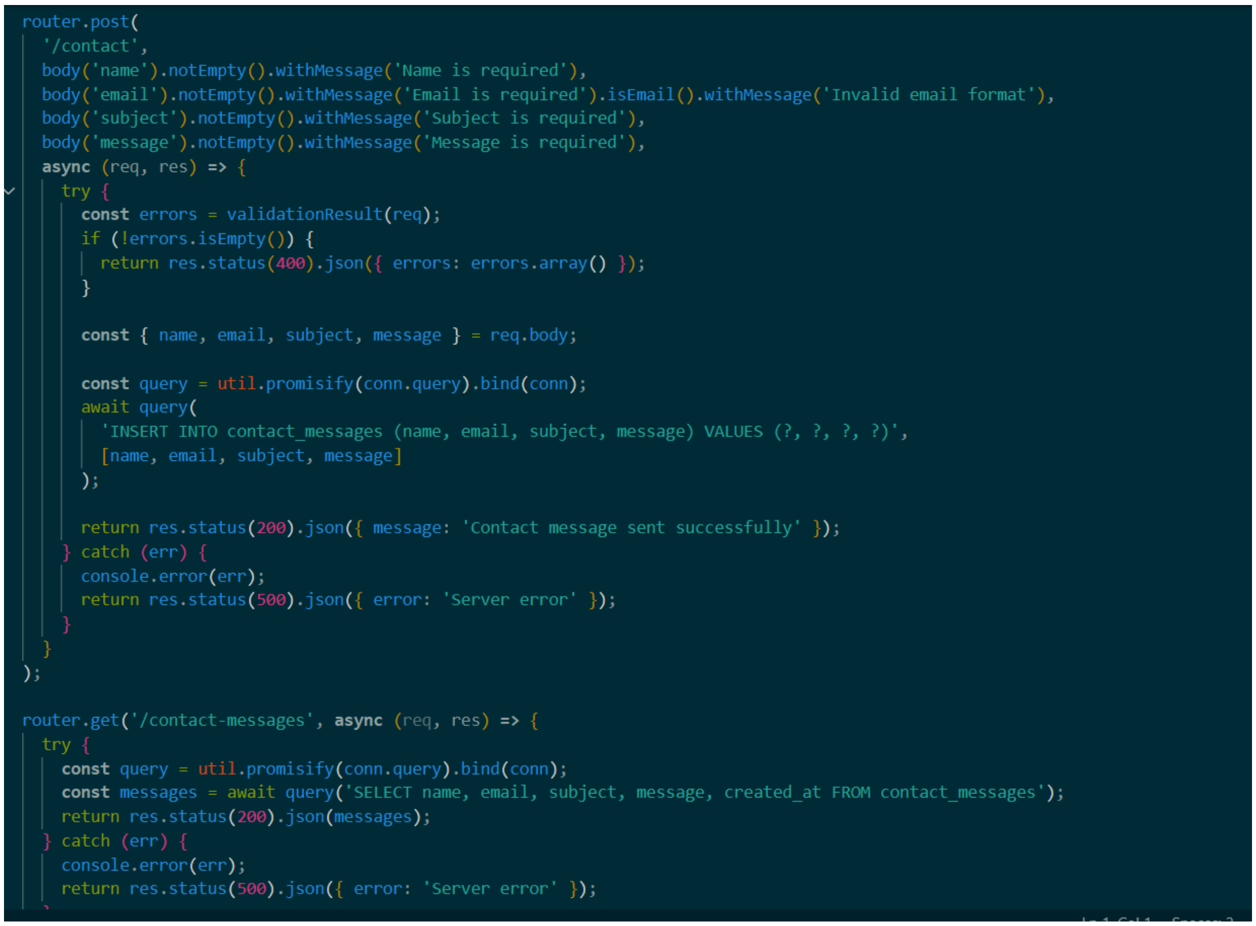
```javascript
// Get ratings for a school with user names
router.get("/ratings/:school_id", async (req, res) => {
  try {
    const query = util.promisify(conn.query).bind(conn);
    const ratings = await query("SELECT ratings.rating, ratings.comment, users.name FROM ratings JOIN users ON ratings.user_id = users.user_id WHERE ratings.school_id = ?", [req.params.school_id]);
    res.status(200).json(ratings);
  } catch (err) {
    console.error(err);
    res.status(500).json({ msg: "Internal server error." });
  }
});


// Endpoint to fetch comments for a specific school
router.get("/comments/:school_id", async (req, res) => {
  try {
    const query = util.promisify(conn.query).bind(conn);
    const comments = await query("SELECT ratings.rating, ratings.timestamp , ratings.comment, users.name FROM ratings JOIN users ON ratings.user_id = users.user_id WHERE ratings.school_id = ?", [req.params.school_id]);
    res.status(200).json({ comments });
  } catch (err) {
    console.error(err);
    res.status(500).json({ msg: "Internal server error." });
  }
});
```

4.1.11 Average rating



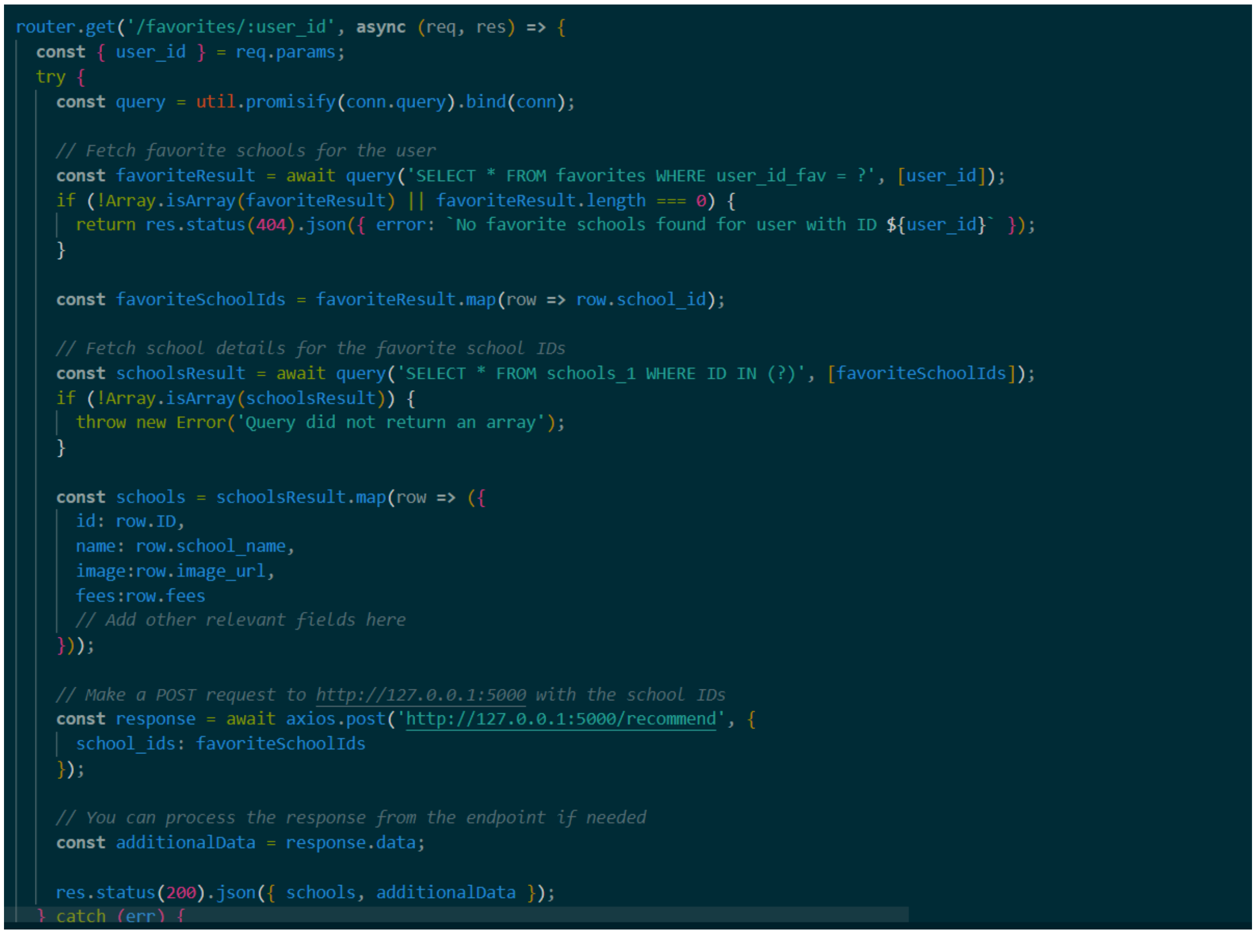
```
router.get("/averagerating/:school_id", async (req, res) => {
  try {
    const query = util.promisify(conn.query).bind(conn);
    const avgRating = await query("SELECT AVG(rating) AS averageRating FROM ratings WHERE school_id = ?", [req.params.school_id]);
    res.status(200).json({ averageRating: avgRating[0].averageRating });
  } catch (err) {
    console.error(err);
    res.status(500).json({ msg: "Internal server error." });
  }
});
```

4.1.12 contact us



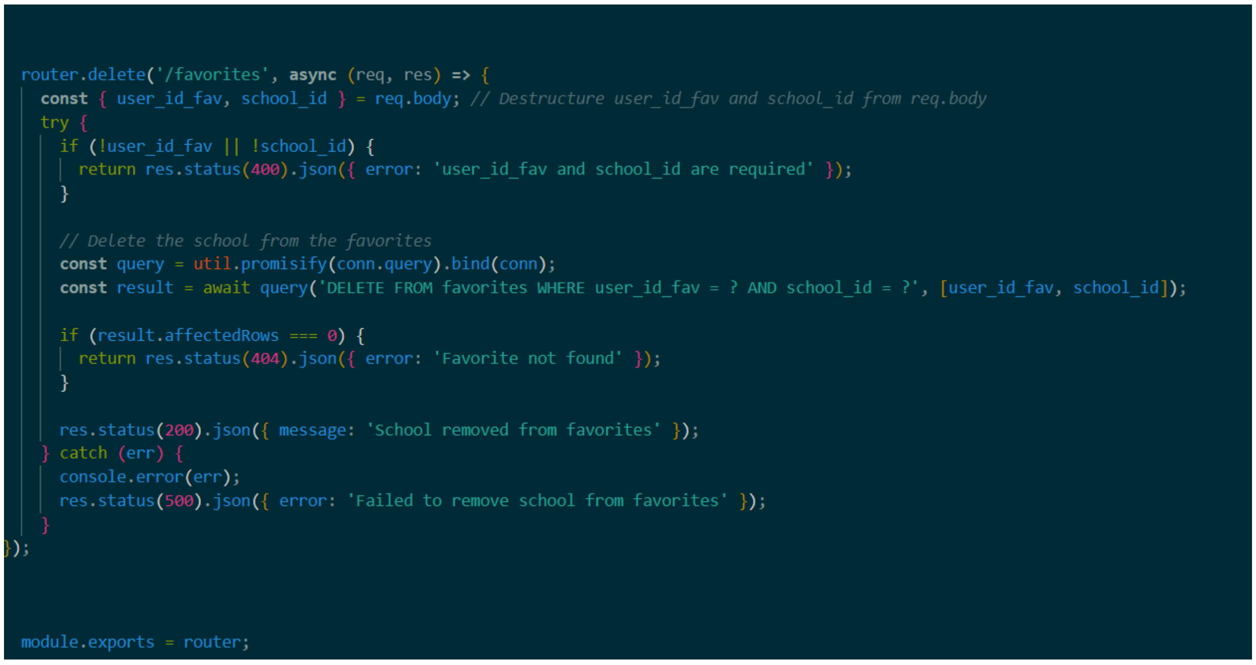
```
router.post(
  '/contact',
  body('name').notEmpty().withMessage('Name is required'),
  body('email').notEmpty().withMessage('Email is required').isEmail().withMessage('Invalid email format'),
  body('subject').notEmpty().withMessage('Subject is required'),
  body('message').notEmpty().withMessage('Message is required'),
  async (req, res) => {
    try {
      const errors = validationResult(req);
      if (!errors.isEmpty()) {
        return res.status(400).json({ errors: errors.array() });
      }

      const { name, email, subject, message } = req.body;

      const query = util.promisify(conn.query).bind(conn);
      await query(
        'INSERT INTO contact_messages (name, email, subject, message) VALUES (?, ?, ?, ?)',
        [name, email, subject, message]
      );

      return res.status(200).json({ message: 'Contact message sent successfully' });
    } catch (err) {
      console.error(err);
      return res.status(500).json({ error: 'Server error' });
    }
  }
);

router.get('/contact-messages', async (req, res) => {
  try {
    const query = util.promisify(conn.query).bind(conn);
    const messages = await query('SELECT name, email, subject, message, created_at FROM contact_messages');
    return res.status(200).json(messages);
  } catch (err) {
    console.error(err);
    return res.status(500).json({ error: 'Server error' });
```

4.1.13 Add school to favorite



```
router.get('/favorites/:user_id', async (req, res) => {
  const { user_id } = req.params;
  try {
    const query = util.promisify(conn.query).bind(conn);

    // Fetch favorite schools for the user
    const favoriteResult = await query('SELECT * FROM favorites WHERE user_id_fav = ?', [user_id]);
    if (!Array.isArray(favoriteResult) || favoriteResult.length === 0) {
      return res.status(404).json({ error: `No favorite schools found for user with ID ${user_id}` });
    }

    const favoriteSchoolIds = favoriteResult.map(row => row.school_id);

    // Fetch school details for the favorite school IDs
    const schoolsResult = await query('SELECT * FROM schools_1 WHERE ID IN (?)', [favoriteSchoolIds]);
    if (!Array.isArray(schoolsResult)) {
      throw new Error('Query did not return an array');
    }

    const schools = schoolsResult.map(row => ({
      id: row.ID,
      name: row.school_name,
      image:row.image_url,
      fees:row.fees
      // Add other relevant fields here
    }));

    // Make a POST request to http://127.0.0.1:5000 with the school IDs
    const response = await axios.post('http://127.0.0.1:5000/recommend', {
      school_ids: favoriteSchoolIds
    });

    // You can process the response from the endpoint if needed
    const additionalData = response.data;

    res.status(200).json({ schools, additionalData });
  } catch (err) {
```

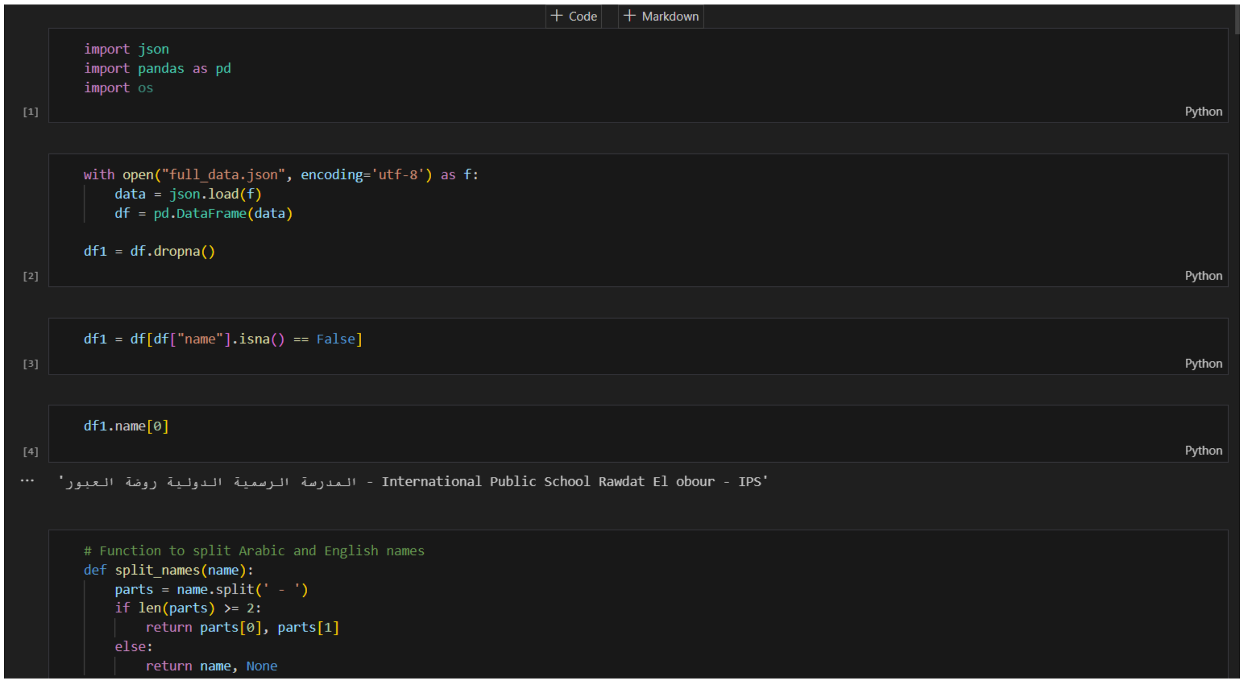
4.1.14 Remove school from favorite



```
router.delete('/favorites', async (req, res) => {
  const { user_id_fav, school_id } = req.body; // Destructure user_id_fav and school_id from req.body
  try {
    if (!user_id_fav || !school_id) {
      return res.status(400).json({ error: 'user_id_fav and school_id are required' });
    }

    // Delete the school from the favorites
    const query = util.promisify(conn.query).bind(conn);
    const result = await query('DELETE FROM favorites WHERE user_id_fav = ? AND school_id = ?', [user_id_fav, school_id]);

    if (result.affectedRows === 0) {
      return res.status(404).json({ error: 'Favorite not found' });
    }

    res.status(200).json({ message: 'School removed from favorites' });
  } catch (err) {
    console.error(err);
    res.status(500).json({ error: 'Failed to remove school from favorites' });
  }
});



module.exports = router;
```

Machine Model

4.1.15 Preprocessing File



```
import json
import pandas as pd
import os
```

[1] Python

```
with open("full_data.json", encoding='utf-8') as f:
    data = json.load(f)
    df = pd.DataFrame(data)

df1 = df.dropna()
```

[2] Python

```
df1 = df[df["name"].isna() == False]
```

[3] Python

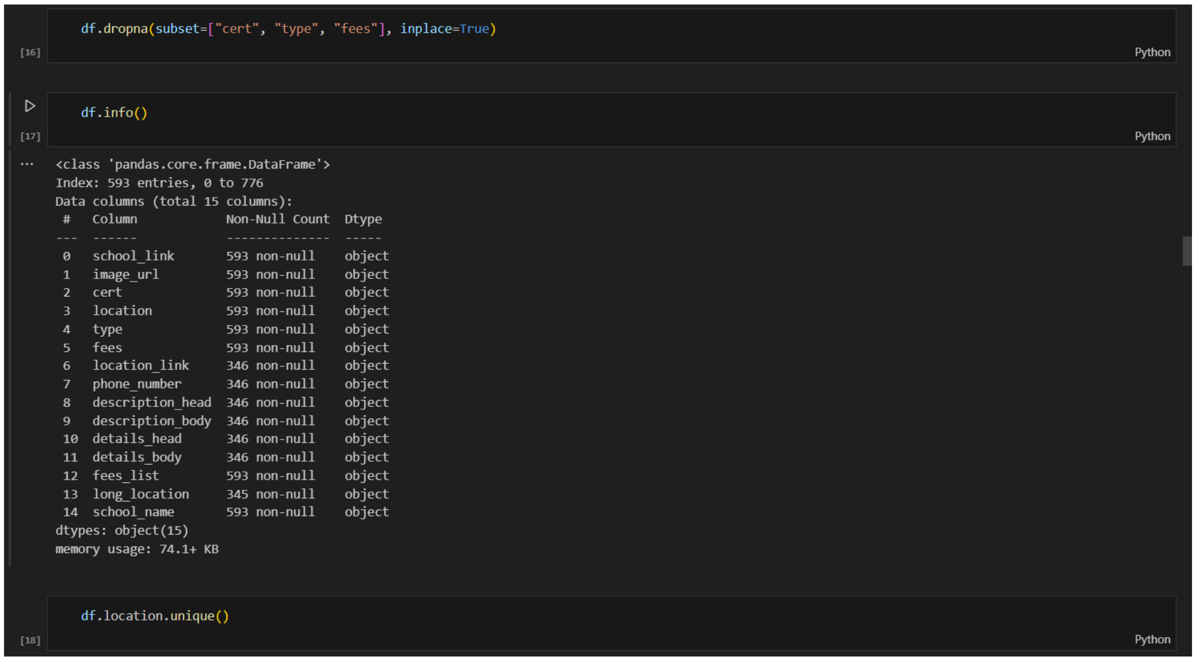
```
df1.name[0]
```

[4] Python

```
'المدرسة الرسمية الدولية روضة العبور - International Public School Rawdat El obour - IPS'
```

```
# Function to split Arabic and English names
def split_names(name):
    parts = name.split(' - ')
    if len(parts) >= 2:
        return parts[0], parts[1]
    else:
        return name, None
```

This block of code contains the beginning of preprocessing and cleaning of our dataset. First thing we imported all the relevant libraries that we will be needing then we viewed the dataset and we made a function to split Arabic names from English names in the "Name".



```
df.dropna(subset=["cert", "type", "fees"], inplace=True)
```

[16] Python

```
df.info()
```

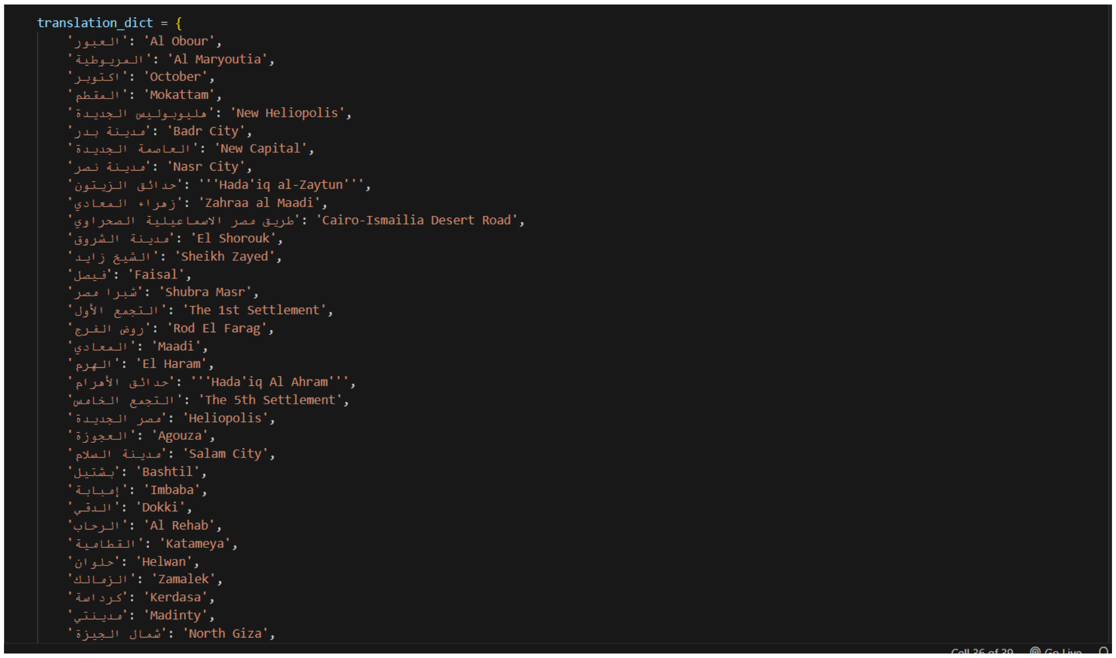
[17] Python

```
<class 'pandas.core.frame.DataFrame'>
Index: 593 entries, 0 to 776
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   school_link       593 non-null    object
 1   image_url         593 non-null    object
 2   cert              593 non-null    object
 3   location          593 non-null    object
 4   type              593 non-null    object
 5   fees              593 non-null    object
 6   location_link     346 non-null    object
 7   phone_number      346 non-null    object
 8   description_head  346 non-null    object
 9   description_body  346 non-null    object
 10  details_head      346 non-null    object
 11  details_body      346 non-null    object
 12  fees_list         593 non-null    object
 13  long_location     345 non-null    object
 14  school_name       593 non-null    object
dtypes: object(15)
memory usage: 74.1+ KB
```

```
df.location.unique()
```

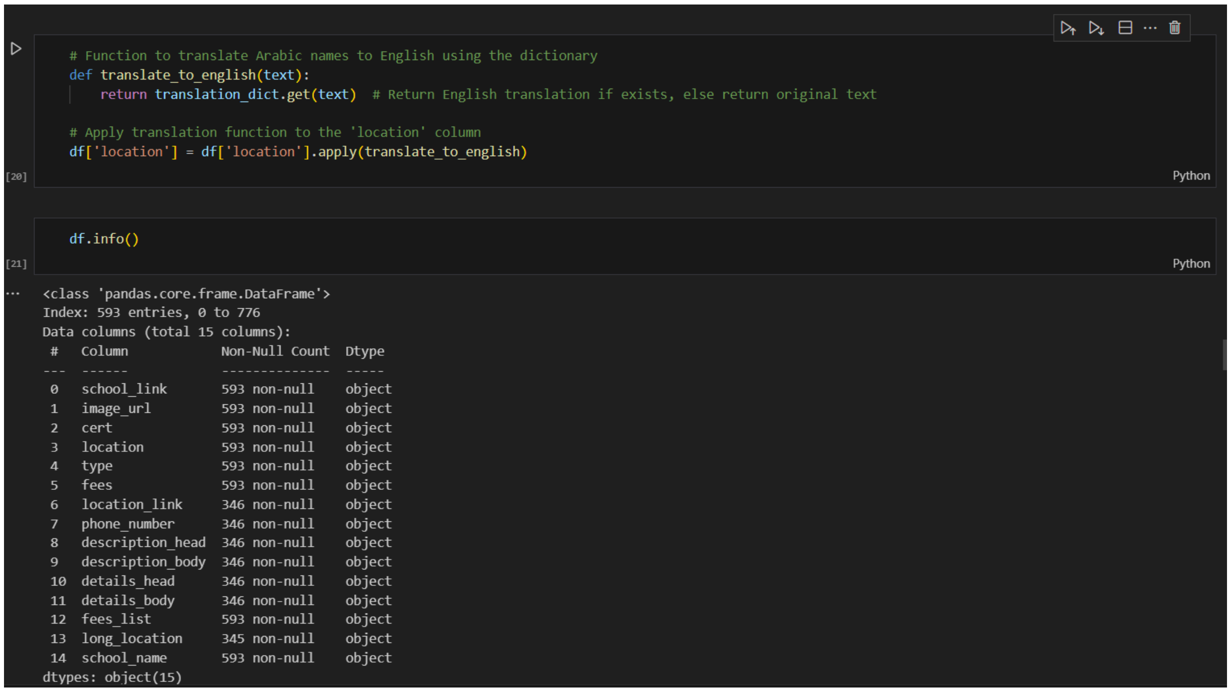
[18] Python

In this block of code, we dropped the nulls from the columns of features that we will be using later on in our recommender model part. Then we viewed all unique values of location column, which will be used later on to map all the Arabic locations to English ones.



```
translation_dict = {
    'العبور': 'Al Obour',
    'المريوطية': 'Al Maryoutia',
    'اكتوبر': 'October',
    'المقطم': 'Mokattam',
    'هليوبوليس الجديدة': 'New Heliopolis',
    'مدينة بدر': 'Badr City',
    'العاصمة الجديدة': 'New Capital',
    'مدينة نصر': 'Nasr City',
    'حدائق الزيتون': '''Hada'iq al-Zaytun''',
    'زهراء المعادي': 'Zahraa al Maadi',
    'طريق مصر الاسماعيلية الصحراوي': 'Cairo-Ismailia Desert Road',
    'مدينة الشروق': 'El Shorouk',
    'الشيخ زايد': 'Sheikh Zayed',
    'فيصل': 'Faisal',
    'شبرا مصر': 'Shubra Masr',
    'التجمع الأول': 'The 1st Settlement',
    'روض الفرج': 'Rod El Farag',
    'المعادي': 'Maadi',
    'الهرم': 'El Haram',
    'حدائق الأهرام': '''Hada'iq Al Ahram''',
    'التجمع الخامس': 'The 5th Settlement',
    'مصر الجديدة': 'Heliopolis',
    'العجوزة': 'Agouza',
    'مدينة السلام': 'Salam City',
    'بشتيل': 'Bashtil',
    'إمبابة': 'Imbaba',
    'الدقي': 'Dokki',
    'الرحاب': 'Al Rehab',
    'القطامية': 'Katameya',
    'حلوان': 'Helwan',
    'الزمالك': 'Zamalek',
    'كرداسة': 'Kerdasa',
    'مدينتي': 'Madinty',
    'شمال الجيزة': 'North Giza',
```

And here the mapping is illustrated.



```python
# Function to translate Arabic names to English using the dictionary
def translate_to_english(text):
    return translation_dict.get(text)  # Return English translation if exists, else return original text

# Apply translation function to the 'location' column
df['location'] = df['location'].apply(translate_to_english)
```

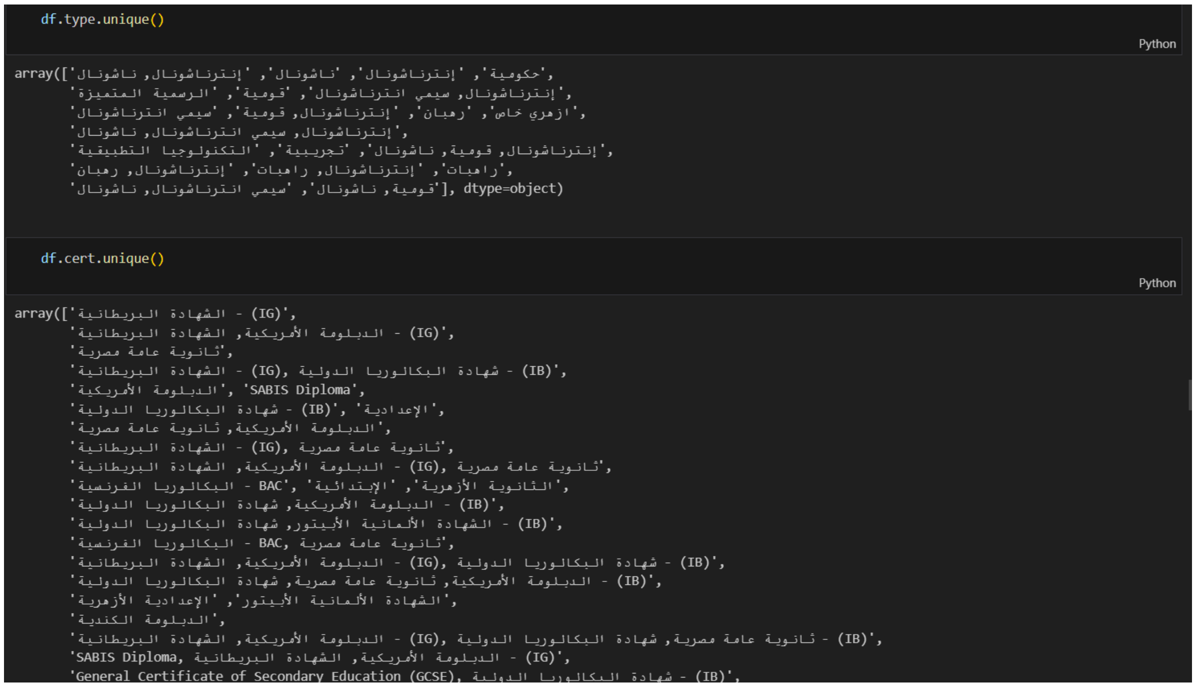
[20]

```python
df.info()
```

[21]

```
<class 'pandas.core.frame.DataFrame'>
Index: 593 entries, 0 to 776
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   school_link       593 non-null    object
 1   image_url         593 non-null    object
 2   cert              593 non-null    object
 3   location          593 non-null    object
 4   type              593 non-null    object
 5   fees              593 non-null    object
 6   location_link     346 non-null    object
 7   phone_number      346 non-null    object
 8   description_head  346 non-null    object
 9   description_body  346 non-null    object
 10  details_head      346 non-null    object
 11  details_body      346 non-null    object
 12  fees_list         593 non-null    object
 13  long_location     345 non-null    object
 14  school_name       593 non-null    object
dtypes: object(15)
```

Here we applied a function to make sure that if a school name is not in the mapping the school name will not be lost, it will remain as the original name. Then we applied our function to translate the names. As you can also see that the datatypes here are still objects which will all be changed later on.



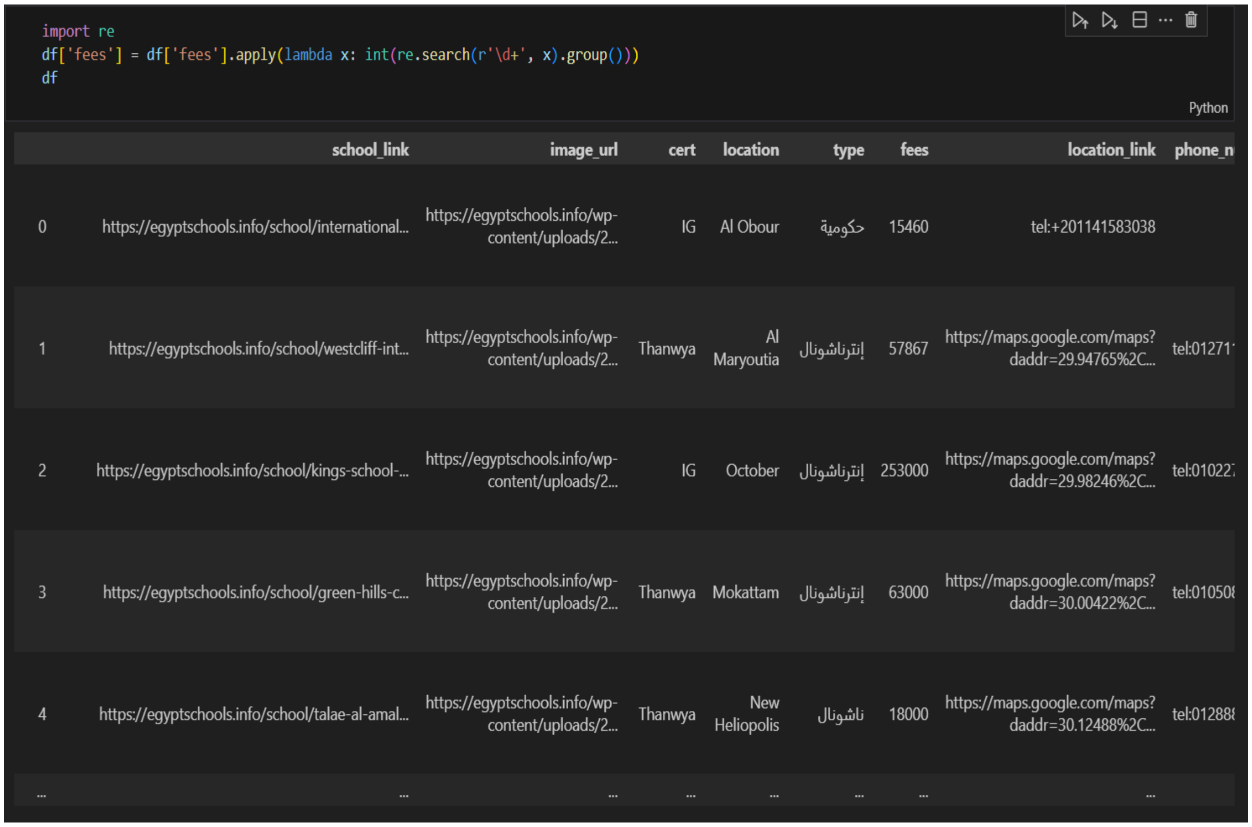
```python
df.type.unique()
```

```
array(['حكومية', 'إنترناشونال', 'ناشونال', 'إنترناشونال, ناشونال',
       'إنترناشونال, سيمي انترناشونال', 'قومية', 'الرسمية المتميزة',
       'ازهري خاص', 'رهبان', 'إنترناشونال, قومية', 'سيمي انترناشونال',
       'إنترناشونال, سيمي انترناشونال, ناشونال',
       'إنترناشونال, قومية, ناشونال', 'تجريبية', 'التكنولوجيا التطبيقية',
       'راهبات', 'إنترناشونال, راهبات', 'إنترناشونال, رهبان',
       'قومية, ناشونال', 'سيمي انترناشونال, ناشونال'], dtype=object)
```

```python
df.cert.unique()
```

```
array(['الشهادة البريطانية - (IG)',
       'الدبلومة الأمريكية, الشهادة البريطانية - (IG)',
       'ثانوية عامة مصرية',
       'الشهادة البريطانية - (IG), شهادة البكالوريا الدولية - (IB)',
       'الدبلومة الأمريكية', 'SABIS Diploma',
       'شهادة البكالوريا الدولية - (IB)', 'الإعدادية',
       'الدبلومة الأمريكية, ثانوية عامة مصرية',
       'الشهادة البريطانية - (IG), ثانوية عامة مصرية',
       'الدبلومة الأمريكية, الشهادة البريطانية - (IG), ثانوية عامة مصرية',
       'البكالوريا الفرنسية - BAC', 'الثانوية الأزهرية', 'الإبتدائية',
       'الدبلومة الأمريكية, شهادة البكالوريا الدولية - (IB)',
       'الشهادة الألمانية الأبيتور, شهادة البكالوريا الدولية - (IB)',
       'البكالوريا الفرنسية - BAC, ثانوية عامة مصرية',
       'الدبلومة الأمريكية, الشهادة البريطانية - (IG), شهادة البكالوريا الدولية - (IB)',
       'الدبلومة الأمريكية, ثانوية عامة مصرية, شهادة البكالوريا الدولية - (IB)',
       'الشهادة الألمانية الأبيتور', 'الإعدادية الأزهرية',
       'الدبلومة الكندية',
       'الدبلومة الأمريكية, الشهادة البريطانية - (IG), ثانوية عامة مصرية, شهادة البكالوريا الدولية - (IB)',
       'SABIS Diploma, الدبلومة الأمريكية, الشهادة البريطانية - (IG)',
       'General Certificate of Secondary Education (GCSE), شهادة البكالوريا الدولية - (IB)',
```

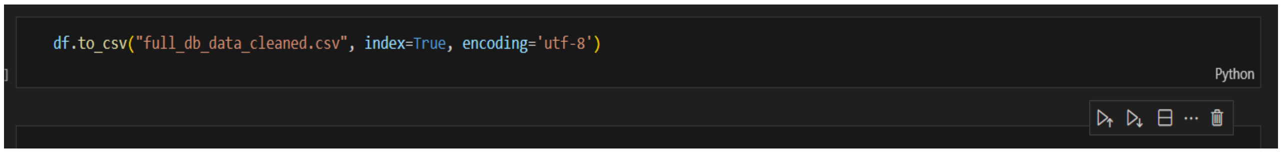
Repeating the mapping step with both "type" and "cert" features.



```python
import re
df['fees'] = df['fees'].apply(lambda x: int(re.search(r'\d+', x).group()))
df
```

	school_link	image_url	cert	location	type	fees	location_link	phone_n
0	https://egyptschools.info/school/international...	https://egyptschools.info/wp-content/uploads/2...	IG	Al Obour	حكومية	15460	tel:+201141583038	
1	https://egyptschools.info/school/westcliff-int...	https://egyptschools.info/wp-content/uploads/2...	Thanwya	Al Maryoutia	إنترناشونال	57867	https://maps.google.com/maps?daddr=29.94765%2C...	tel:01271
2	https://egyptschools.info/school/kings-school-...	https://egyptschools.info/wp-content/uploads/2...	IG	October	إنترناشونال	253000	https://maps.google.com/maps?daddr=29.98246%2C...	tel:01022
3	https://egyptschools.info/school/green-hills-c...	https://egyptschools.info/wp-content/uploads/2...	Thanwya	Mokattam	إنترناشونال	63000	https://maps.google.com/maps?daddr=30.00422%2C...	tel:01050
4	https://egyptschools.info/school/talae-al-amal...	https://egyptschools.info/wp-content/uploads/2...	Thanwya	New Heliopolis	ناشونال	18000	https://maps.google.com/maps?daddr=30.12488%2C...	tel:01288
...	...	...	...	...	...	...	...	

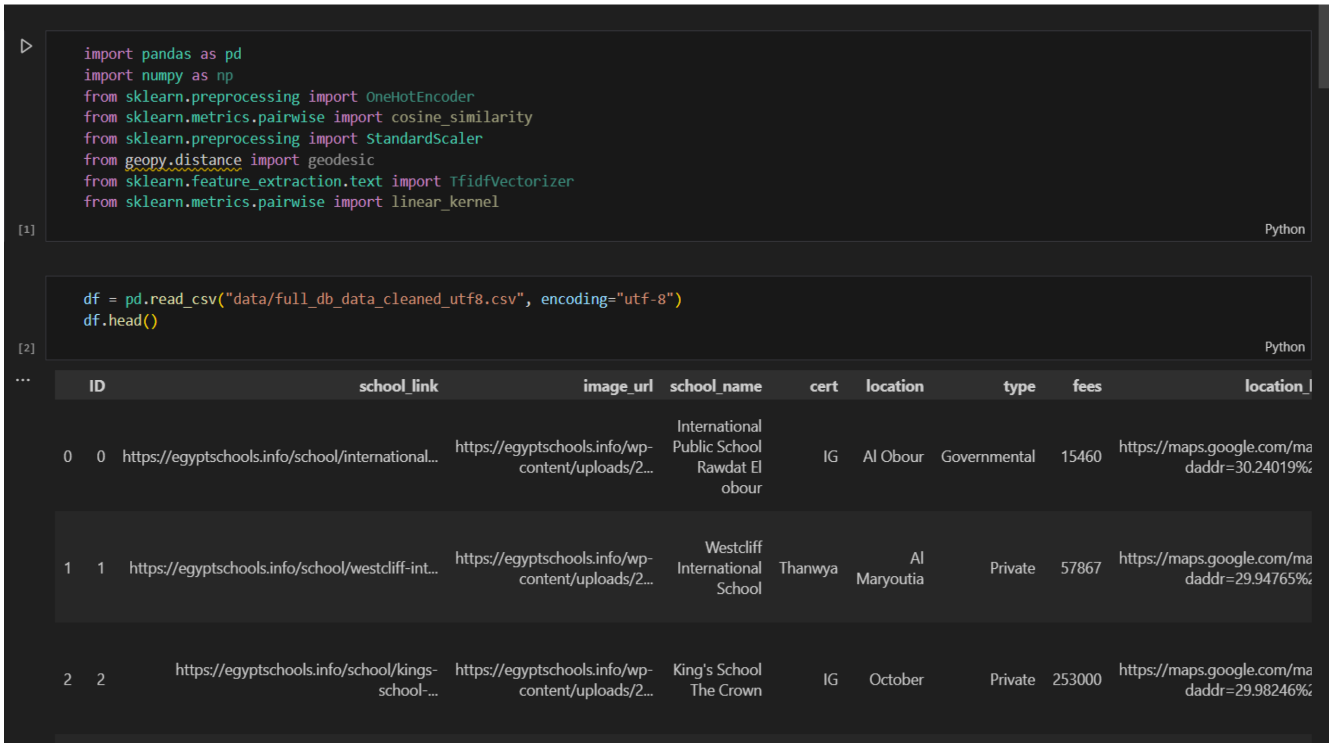
The purpose of this code is to parse the 'fees' column, extract the numeric value from each cell, and store the integer value back in the 'fees' column to make it easy to compare the schools prices, In addition, it will be used later in our recommender model.



```python
df.to_csv("full_db_data_cleaned.csv", index=True, encoding='utf-8')
```

In the final line in the preprocessing file, we saved our preprocessed data as csv file. and encoding='utf-8' to deal with Arabic texts.

4.1.16 Recommendation File



```
import pandas as pd
import numpy as np
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.preprocessing import StandardScaler
from geopy.distance import geodesic
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel
```

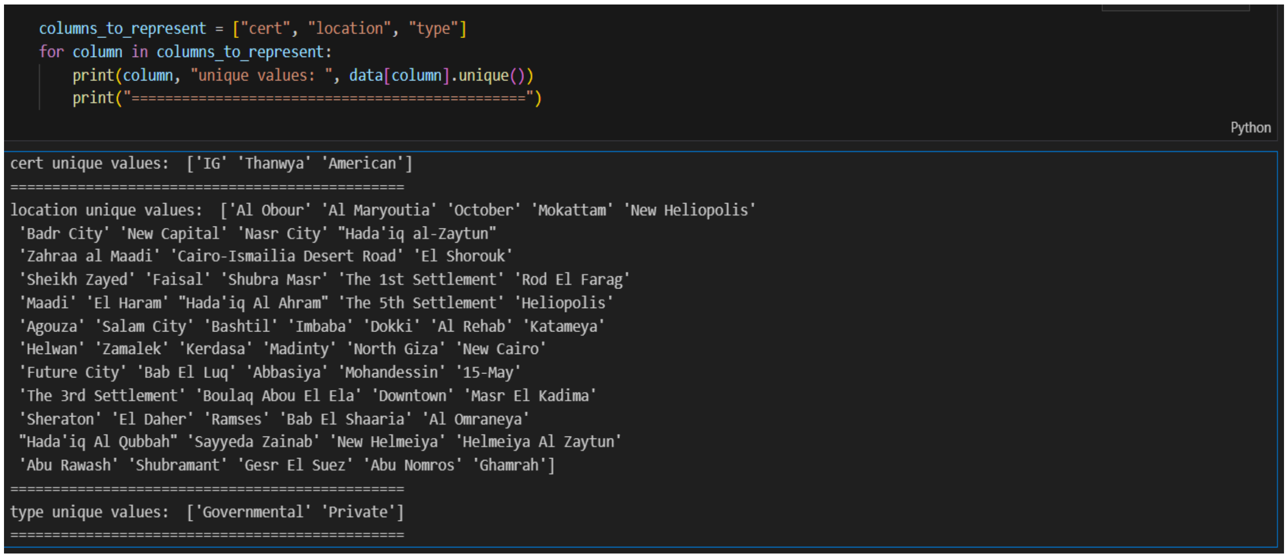
[1] Python

```
df = pd.read_csv("data/full_db_data_cleaned_utf8.csv", encoding="utf-8")
df.head()
```

[2] Python

	ID	school_link	image_url	school_name	cert	location	type	fees	location_l
0	0	https://egyptschools.info/school/international...	https://egyptschools.info/wp-content/uploads/2...	International Public School Rawdat El obour	IG	Al Obour	Governmental	15460	https://maps.google.com/ma daddr=30.24019%2
1	1	https://egyptschools.info/school/westcliff-int...	https://egyptschools.info/wp-content/uploads/2...	Westcliff International School	Thanwya	Al Maryoutia	Private	57867	https://maps.google.com/ma daddr=29.94765%2
2	2	https://egyptschools.info/school/kings-school-...	https://egyptschools.info/wp-content/uploads/2...	King's School The Crown	IG	October	Private	253000	https://maps.google.com/ma daddr=29.98246%2

Importing all python libraries that will be needed in the recommendation part. Then in the second cell we started reading our data.

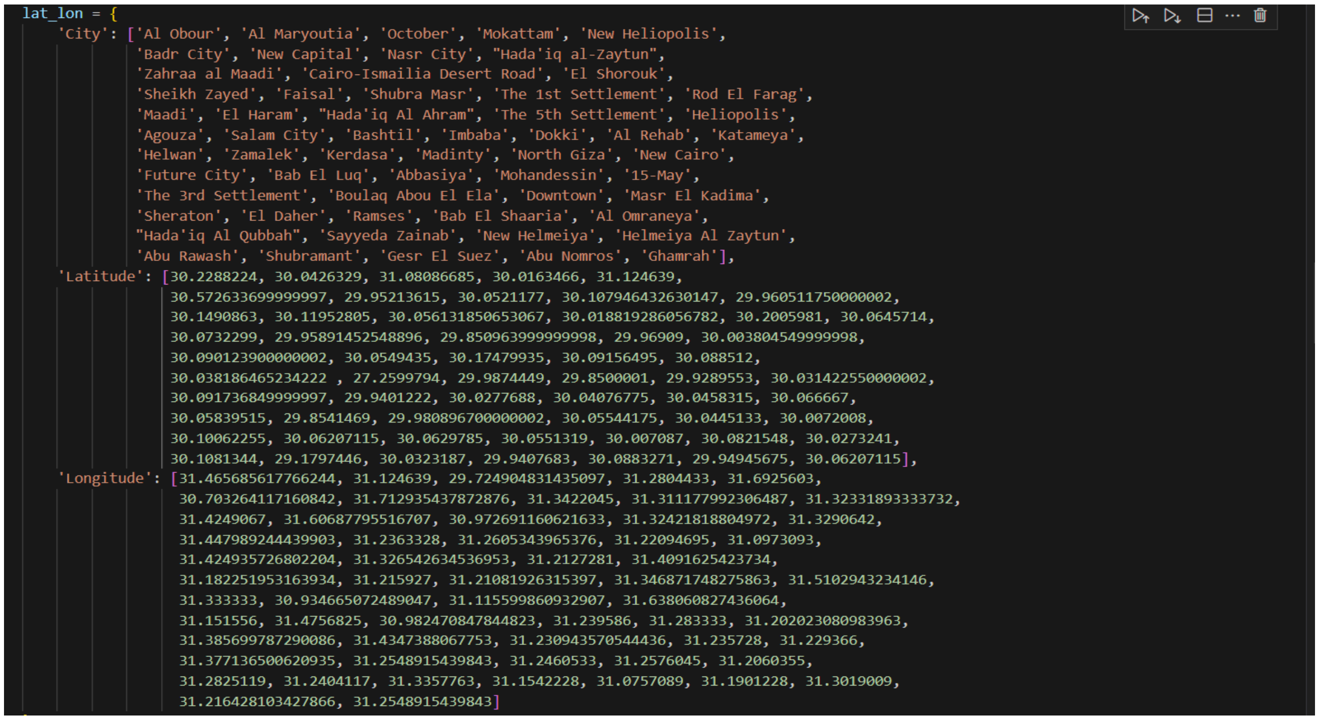


```python
columns_to_represent = ["cert", "location", "type"]
for column in columns_to_represent:
    print(column, "unique values: ", data[column].unique())
    print("==============================================")
```

Python

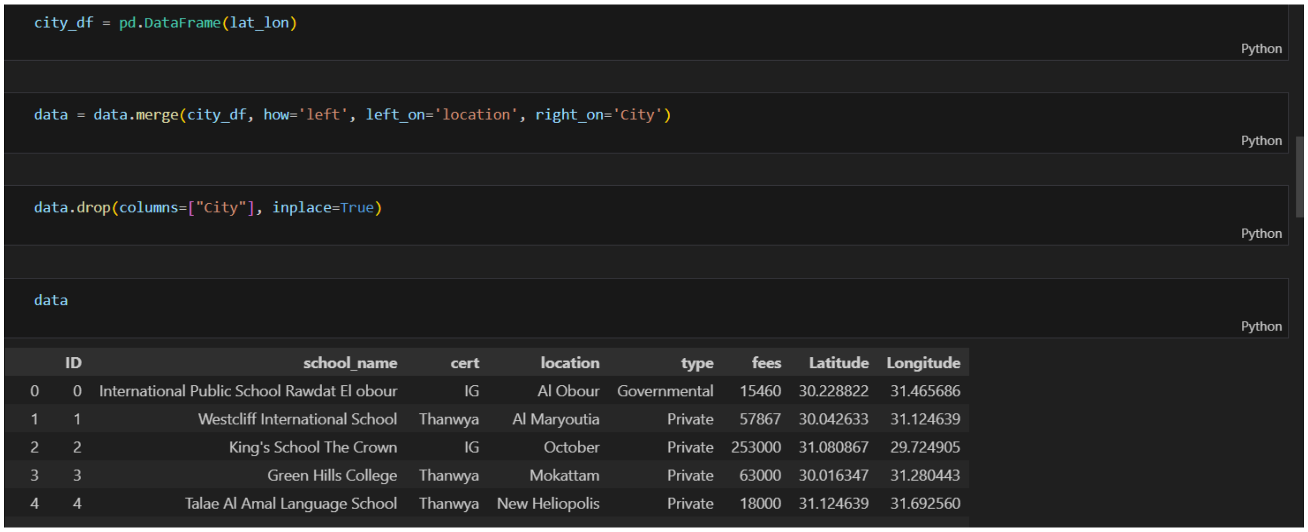
```
cert unique values:  ['IG' 'Thanwya' 'American']
==============================================
location unique values:  ['Al Obour' 'Al Maryoutia' 'October' 'Mokattam' 'New Heliopolis'
 'Badr City' 'New Capital' 'Nasr City' "Hada'iq al-Zaytun"
 'Zahraa al Maadi' 'Cairo-Ismailia Desert Road' 'El Shorouk'
 'Sheikh Zayed' 'Faisal' 'Shubra Masr' 'The 1st Settlement' 'Rod El Farag'
 'Maadi' 'El Haram' "Hada'iq Al Ahram" 'The 5th Settlement' 'Heliopolis'
 'Agouza' 'Salam City' 'Bashtil' 'Imbaba' 'Dokki' 'Al Rehab' 'Katameya'
 'Helwan' 'Zamalek' 'Kerdasa' 'Madinty' 'North Giza' 'New Cairo'
 'Future City' 'Bab El Luq' 'Abbasiya' 'Mohandessin' '15-May'
 'The 3rd Settlement' 'Boulaq Abou El Ela' 'Downtown' 'Masr El Kadima'
 'Sheraton' 'El Daher' 'Ramses' 'Bab El Shaaria' 'Al Omraneya'
 "Hada'iq Al Qubbah" 'Sayyeda Zainab' 'New Helmeiya' 'Helmeiya Al Zaytun'
 'Abu Rawash' 'Shubramant' 'Gesr El Suez' 'Abu Nomros' 'Ghamrah']
==============================================
type unique values:  ['Governmental' 'Private']
==============================================
```

Viewing unique values for these columns to decide if we are going to replace them manually or using one hot encode for example.



```python
lat_lon = {
    'City': ['Al Obour', 'Al Maryoutia', 'October', 'Mokattam', 'New Heliopolis',
             'Badr City', 'New Capital', 'Nasr City', "Hada'iq al-Zaytun",
             'Zahraa al Maadi', 'Cairo-Ismailia Desert Road', 'El Shorouk',
             'Sheikh Zayed', 'Faisal', 'Shubra Masr', 'The 1st Settlement', 'Rod El Farag',
             'Maadi', 'El Haram', "Hada'iq Al Ahram", 'The 5th Settlement', 'Heliopolis',
             'Agouza', 'Salam City', 'Bashtil', 'Imbaba', 'Dokki', 'Al Rehab', 'Katameya',
             'Helwan', 'Zamalek', 'Kerdasa', 'Madinty', 'North Giza', 'New Cairo',
             'Future City', 'Bab El Luq', 'Abbasiya', 'Mohandessin', '15-May',
             'The 3rd Settlement', 'Boulaq Abou El Ela', 'Downtown', 'Masr El Kadima',
             'Sheraton', 'El Daher', 'Ramses', 'Bab El Shaaria', 'Al Omraneya',
             "Hada'iq Al Qubbah", 'Sayyeda Zainab', 'New Helmeiya', 'Helmeiya Al Zaytun',
             'Abu Rawash', 'Shubramant', 'Gesr El Suez', 'Abu Nomros', 'Ghamrah'],
    'Latitude': [30.2288224, 30.0426329, 31.08086685, 30.0163466, 31.124639,
                 30.572633699999997, 29.95213615, 30.0521177, 30.107946432630147, 29.960511750000002,
                 30.1490863, 30.11952805, 30.056131850653067, 30.018819286056782, 30.2005981, 30.0645714,
                 30.0732299, 29.95891452548896, 29.850963999999998, 29.96909, 30.003804549999998,
                 30.090123900000002, 30.0549435, 30.17479935, 30.09156495, 30.088512,
                 30.038186465234222 , 27.2599794, 29.9874449, 29.8500001, 29.9289553, 30.031422550000002,
                 30.091736849999997, 29.9401222, 30.0277688, 30.04076775, 30.0458315, 30.066667,
                 30.05839515, 29.8541469, 29.980896700000002, 30.05544175, 30.0445133, 30.0072008,
                 30.10062255, 30.06207115, 30.0629785, 30.0551319, 30.007087, 30.0821548, 30.0273241,
                 30.1081344, 29.1797446, 30.0323187, 29.9407683, 30.0883271, 29.94945675, 30.06207115],
    'Longitude': [31.465685617766244, 31.124639, 29.724904831435097, 31.2804433, 31.6925603,
                  30.703264117160842, 31.712935437872876, 31.3422045, 31.311177992306487, 31.32331893333732,
                  31.4249067, 31.60687795516707, 30.972691160621633, 31.32421818804972, 31.3290642,
                  31.447989244439903, 31.2363328, 31.2605343965376, 31.22094695, 31.0973093,
                  31.424935726802204, 31.326542634536953, 31.2127281, 31.4091625423734,
                  31.182251953163934, 31.215927, 31.21081926315397, 31.346871748275863, 31.5102943234146,
                  31.333333, 30.934665072489047, 31.115599860932907, 31.638060827436064,
                  31.151556, 31.4756825, 30.982470847844823, 31.239586, 31.283333, 31.202023080983963,
                  31.385699787290086, 31.4347388067753, 31.230943570544436, 31.235728, 31.229366,
                  31.377136500620935, 31.2548915439843, 31.2460533, 31.2576045, 31.2060355,
                  31.2825119, 31.2404117, 31.3357763, 31.1542228, 31.0757089, 31.1901228, 31.3019009,
                  31.216428103427866, 31.2548915439843]
```

Using "lat_lon" to represent the latitude and longitude of each school(school location on maps).



```python
city_df = pd.DataFrame(lat_lon)
```

Python

```python
data = data.merge(city_df, how='left', left_on='location', right_on='City')
```

Python

```python
data.drop(columns=["City"], inplace=True)
```

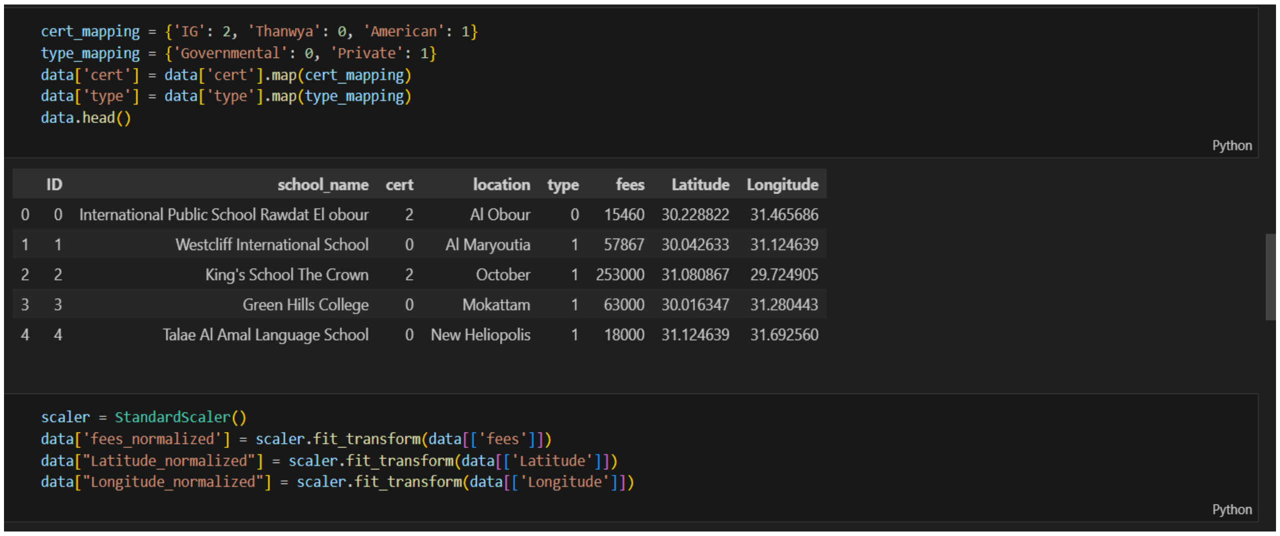
Python

```python
data
```

Python

	ID	school_name	cert	location	type	fees	Latitude	Longitude
0	0	International Public School Rawdat El obour	IG	Al Obour	Governmental	15460	30.228822	31.465686
1	1	Westcliff International School	Thanwya	Al Maryoutia	Private	57867	30.042633	31.124639
2	2	King's School The Crown	IG	October	Private	253000	31.080867	29.724905
3	3	Green Hills College	Thanwya	Mokattam	Private	63000	30.016347	31.280443
4	4	Talae Al Amal Language School	Thanwya	New Heliopolis	Private	18000	31.124639	31.692560

This line of code is merging the data Data Frame with the city_df Data Frame based on the 'location' column in data and the 'City' column in city_df. The resulting Data Frame will have all the columns from both the original Data Frames.



```python
cert_mapping = {'IG': 2, 'Thanwya': 0, 'American': 1}
type_mapping = {'Governmental': 0, 'Private': 1}
data['cert'] = data['cert'].map(cert_mapping)
data['type'] = data['type'].map(type_mapping)
data.head()
```

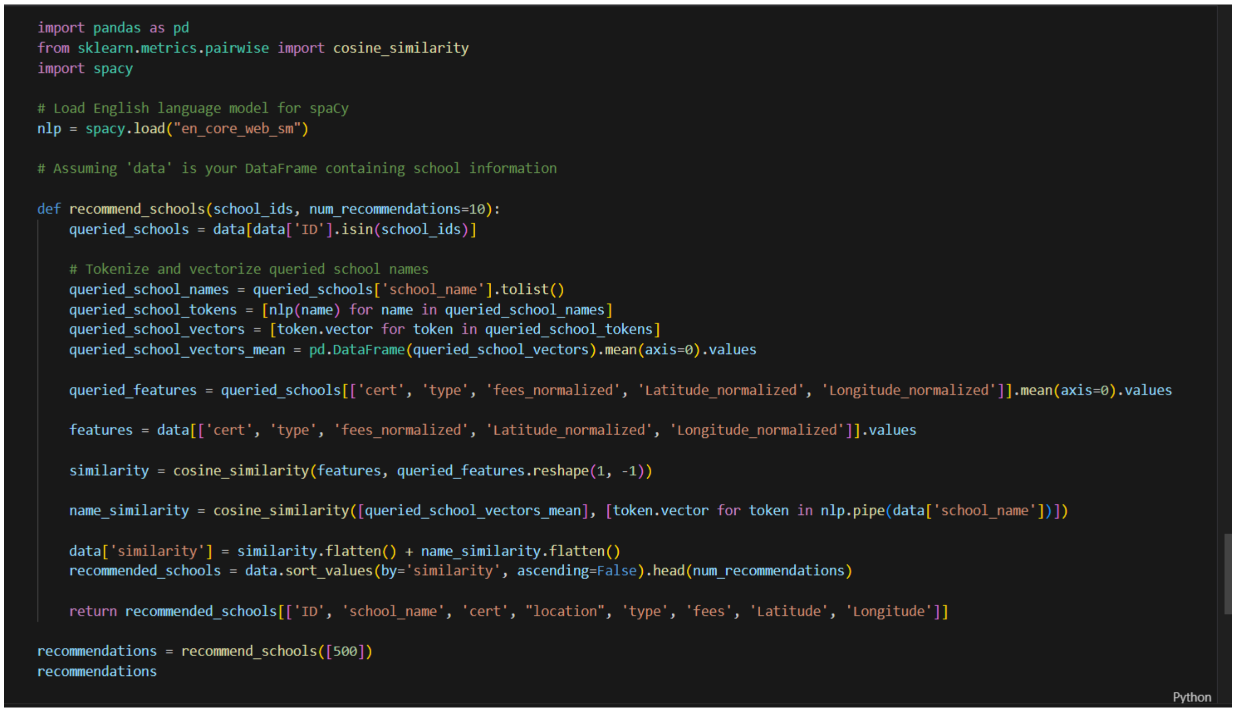
Python

	ID	school_name	cert	location	type	fees	Latitude	Longitude
0	0	International Public School Rawdat El obour	2	Al Obour	0	15460	30.228822	31.465686
1	1	Westcliff International School	0	Al Maryoutia	1	57867	30.042633	31.124639
2	2	King's School The Crown	2	October	1	253000	31.080867	29.724905
3	3	Green Hills College	0	Mokattam	1	63000	30.016347	31.280443
4	4	Talae Al Amal Language School	0	New Heliopolis	1	18000	31.124639	31.692560

```python
scaler = StandardScaler()
data['fees_normalized'] = scaler.fit_transform(data[['fees']])
data["Latitude_normalized"] = scaler.fit_transform(data[['Latitude']])
data["Longitude_normalized"] = scaler.fit_transform(data[['Longitude']])
```

Python

Here we decided to do manual mapping for "cert" and "type" features for they have limited number of values.
Then we used the Standard Scaler to scale the values of the numeric features to give them approximately equal values.

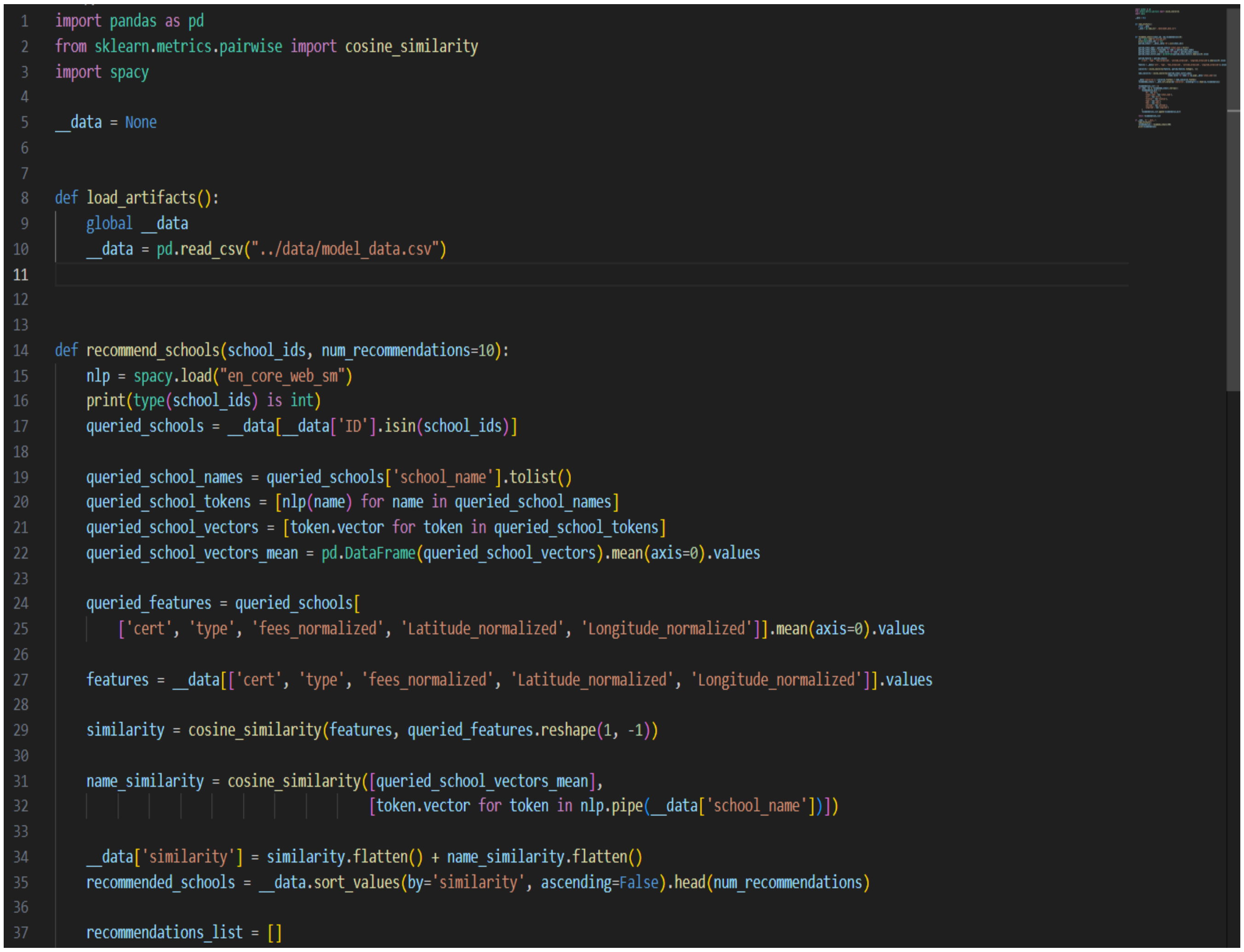


```python
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity
import spacy

# Load English language model for spaCy
nlp = spacy.load("en_core_web_sm")

# Assuming 'data' is your DataFrame containing school information

def recommend_schools(school_ids, num_recommendations=10):
    queried_schools = data[data['ID'].isin(school_ids)]

    # Tokenize and vectorize queried school names
    queried_school_names = queried_schools['school_name'].tolist()
    queried_school_tokens = [nlp(name) for name in queried_school_names]
    queried_school_vectors = [token.vector for token in queried_school_tokens]
    queried_school_vectors_mean = pd.DataFrame(queried_school_vectors).mean(axis=0).values

    queried_features = queried_schools[['cert', 'type', 'fees_normalized', 'Latitude_normalized', 'Longitude_normalized']].mean(axis=0).values

    features = data[['cert', 'type', 'fees_normalized', 'Latitude_normalized', 'Longitude_normalized']].values

    similarity = cosine_similarity(features, queried_features.reshape(1, -1))

    name_similarity = cosine_similarity([queried_school_vectors_mean], [token.vector for token in nlp.pipe(data['school_name'])])

    data['similarity'] = similarity.flatten() + name_similarity.flatten()
    recommended_schools = data.sort_values(by='similarity', ascending=False).head(num_recommendations)

    return recommended_schools[['ID', 'school_name', 'cert', "location", 'type', 'fees', 'Latitude', 'Longitude']]

recommendations = recommend_schools([500])
recommendations
```

Python

Here we are importing our similarity model and spacy for the NLP part.
This code illustrates our recommendation model, where the user can input a list of school IDs, and the system will recommend similar schools based on both their features and their names.

4.1.17 Utils File



```python
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity
import spacy

__data = None


def load_artifacts():
    global __data
    __data = pd.read_csv("../data/model_data.csv")



def recommend_schools(school_ids, num_recommendations=10):
    nlp = spacy.load("en_core_web_sm")
    print(type(school_ids) is int)
    queried_schools = __data[__data['ID'].isin(school_ids)]

    queried_school_names = queried_schools['school_name'].tolist()
    queried_school_tokens = [nlp(name) for name in queried_school_names]
    queried_school_vectors = [token.vector for token in queried_school_tokens]
    queried_school_vectors_mean = pd.DataFrame(queried_school_vectors).mean(axis=0).values

    queried_features = queried_schools[
        ['cert', 'type', 'fees_normalized', 'Latitude_normalized', 'Longitude_normalized']].mean(axis=0).values

    features = __data[['cert', 'type', 'fees_normalized', 'Latitude_normalized', 'Longitude_normalized']].values

    similarity = cosine_similarity(features, queried_features.reshape(1, -1))

    name_similarity = cosine_similarity([queried_school_vectors_mean],
                                        [token.vector for token in nlp.pipe(__data['school_name'])])

    __data['similarity'] = similarity.flatten() + name_similarity.flatten()
    recommended_schools = __data.sort_values(by='similarity', ascending=False).head(num_recommendations)

    recommendations_list = []
```

The purpose of this part of the code is to provide a way to load the necessary data for the school recommendation system. By keeping the data in a global variable, other parts of the code can easily access the loaded data without having to read the file every time it's needed.

4.3 Technologies

React :

React is a JavaScript library for building user interfaces, particularly single-page applications where you need a fast and interactive user experience. React allows developers to create large web applications that can update and render efficiently in response to data changes. It does this through a concept called the virtual DOM, which minimizes the number of direct DOM manipulations required.

CSS:

CSS (Cascading Style Sheets) is a style sheet language used to describe the presentation of a document written in HTML or XML. In your React application, CSS is used to style components and layout the application.

Fetch API:

The Fetch API is a modern interface that allows you to make HTTP requests to servers from web browsers. It provides a more powerful and flexible feature set than older techniques like XMLHttpRequest.

Asynchronous Requests:

The Fetch API uses promises, allowing you to make requests asynchronously and handle responses or errors easily.
Usage: With the Fetch API, you can send GET, POST, PUT, DELETE requests, and more. It supports sending JSON data to and receiving JSON data from the server.

Node.js:

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. It allows you to run JavaScript on the server side, enabling the development of scalable and high-performance applications.
Event-Driven Architecture: Node.js is designed to handle asynchronous I/O operations, making it ideal for real-time applications.

Single-Threaded: Node.js operates on a single-threaded event loop, which can handle multiple connections simultaneously without creating new threads for each connection.

Express:

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.
Express provides a straightforward way to handle different routes in your application. It supports various HTTP methods and URL patterns.
Express middleware functions are used to handle requests, responses, and any processing in between. This includes tasks like parsing JSON request bodies, handling authentication, and more.

Visual Studio code IDE:

It is a free source-code editor made by Microsoft its Features include
support for debugging, syntax highlighting, intelligent code completion,
code refactoring and embedded Git, and it supports flutter.

JSON for Data Interchange:

JSON (JavaScript Object Notation) is a lightweight data interchange format that's easy for humans to read and write, and easy for machines to parse and generate. JSON is the standard format for data exchange in web applications.

Python:

Python is an object-oriented, high-level programming language with integrated dynamic semantics primarily for web and app development.

Chapter 5: Testing

5.1 Unit Testing

Unit testing forms the foundational level of our testing process. It involves testing individual components or units of our software to verify their functionality in isolation. This meticulous testing ensures that each unit of code performs as intended, contributing to the overall system's stability.

5.1.1 User Registration and Profile Creation

- Description: Verify that the user registration and profile creation process functions correctly.

- Test Steps:

1. Navigate to the registration page and fill in the required information.

2. Submit the registration form and verify that the user is successfully registered.

3. Log in with the registered credentials and ensure that the user's profile is displayed correctly.

4. Verify that all the entered information is accurately saved in the user's profile.

5. Test the validation of the registration form by leaving required fields blank or providing invalid input.

6. Ensure that appropriate error messages are displayed for missing or invalid input.

5.1.2 School Profile Management for Administrators

- Description: Ensure that administrators can manage school profiles effectively.

- Test Steps:

1. Log in as an administrator and access the school profile management section.

2. Add a new school profile by providing all the necessary details.

3. Save the profile and verify that the new school is correctly added to the system.

4. Edit an existing school profile and update the information.

5. Confirm that the changes are reflected in the updated school profile.

6. Test the deletion of a school profile and ensure that it is successfully removed from the system.

7. Verify that only authorized administrators have access to the school profile management functionality.

5.1.3 Advanced Search and Filtering Options

- Description: Verify that the search and filtering options allow users to find schools based on specific criteria.

- Test Steps:

1. Use the search functionality to find schools based on location, academic programs, or facilities.

2. Verify that the search results accurately match the specified criteria.

3. Apply different filters (e.g., proximity, fees, educational approach) and ensure that the displayed schools meet the selected criteria.

4. Combine multiple search and filter options to perform complex searches and validate the accuracy of the results.

5. Test the performance of the search and filtering functionalities with a large number of schools and varying search criteria.

5.1.4 Side-by-Side School Comparison Feature

- Description: Ensure that users can compare multiple schools side by side.

- Test Steps:

1. Select two or more schools for comparison.

2. Verify that the selected schools are displayed in a comparison view.

3. Check that the relevant information, such as location, facilities, academic programs, and fees, is presented for each school.

4. Ensure that users can easily navigate between the compared schools and view the detailed information.

5. Test the system's ability to handle a large number of compared schools without performance issues.

5.1.5 Personalized Recommendation Engine

- Description: Validate the accuracy and relevance of the personalized school recommendation feature.

- Test Steps:

1. Create user profiles with specific preferences, such as location, academic programs, and extracurricular activities.

2. Verify that the system generates personalized school recommendations based on the user’s liked schools.

3. Confirm that the recommended schools align with the specified preferences and requirements.

4. Test the recommendation engine's ability to adapt and update recommendations when user preferences are modified.

5. Validate the system's response time for generating personalized recommendations.

5.2 Integrated Testing:

Once the individual units have been thoroughly tested, we proceed to integrated testing. This phase involves combining these units to evaluate their interactions and ensure seamless integration. By assessing the system as a whole, we can identify and address any potential issues that may arise due to component interactions.

5.2.1 User Journey Testing

- Description: Test the end-to-end user journey, ensuring seamless data flow and functionality across system components.

- Test Steps:

1. Complete the user journey from registration to profile creation, school search, comparison, and recommendation.

2. Verify that data is correctly passed between different system components.

3. Test the integration of visual resources, user feedback, and rating systems within the user journey.

4. Ensure that the system maintains the user's session and displays relevant information consistently throughout the journey.

5. Validate the overall performance and responsiveness of the user journey.

5.2.2 Integration with External Data Sources

- Description: Validate the integration with external data sources to ensure accurate and up-to-date information.

- Test Steps:

1. Test the integration with external data sources (e.g., educational institutions, government databases).

2. Verify that the system correctly retrieves and updates data from the external sources.

3. Test different scenarios, such as when external data sources are temporarily unavailable or when there are inconsistencies in the retrieved data.

4. Validate the system's ability to handle and gracefully recover from external data source failures.

5.2.3 Compatibility Testing

Test Steps:

1. Test the system's compatibility with different web browsers (e.g., Chrome, Firefox, Safari, Internet Explorer).

2. Verify that the system's layout, functionality, and visual resources are consistent across different browsers.

3. Test the system's compatibility with different devices and various operating systems.

4. Validate that the system's responsiveness and user experience are consistent across different devices and screen sizes.

5. Test the accessibility features, such as keyboard navigation and screen reader compatibility, to ensure inclusivity for users with disabilities.

5.3 Additional Testing:

In addition to unit and integrated testing, we conduct various additional testing procedures to ensure comprehensive coverage. These may include regression testing, performance testing, security testing, and user acceptance testing. These additional tests help us identify any functional, performance, or security issues and make necessary improvements to deliver a robust and reliable software system.

5.3.1 Performance Testing

- Description: Verify the system's performance under different loads and concurrent user scenarios.

- Test Steps:

1. Simulate a high number of concurrent users accessing the system.

2. Measure the response times and ensure that the system remains responsive and performs well under the load.

3. Test the system's scalability by gradually increasing the number of schools and users and verifying that the performance remains stable.

4. Validate the system's resource utilization and identify any bottlenecks or performance issues.

5.3.2 Security Testing

- Description: Validate the system's security measures to protect user data.

- Test Steps:

1. Verify that user data is securely stored and transmitted using encryption techniques.

2. Test the system's authentication and authorization mechanisms to ensure only authorized users can access sensitive data.

3. Attempt unauthorized access to the system and verify that appropriate security measures are in place to prevent it.

4. Test the system for common security vulnerabilities, such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).

5. Validate that the system has appropriate measures in place to mitigate or prevent security vulnerabilities.

5.3.3 Usability Testing

- Description: Evaluate the system's interface, navigation, and user experience.

- Test Steps:

1. Conduct usability testing sessions with representative users to evaluate the system's interface, navigation, and overall user experience.

2. Collect user feedback on the system's ease of use, intuitiveness, and clarity of information presentation.

3. Identify any usability issues or areas for improvement based on user feedback and observations.

4. Test the system's accessibility features, including keyboard navigation and screen reader compatibility, to ensure inclusivity for users with disabilities.

5. Validate the system's responsiveness and performance in providing a seamless and user-friendly experience.

Chapter 6: Results and Discussion

6.1 Results

6.1.1 Expected Result

The objective of this project was to develop a comprehensive school search and recommendation system that would accurately match users with schools based on their preferences and requirements. The expected result was a fully functional system that would allow users to register, create profiles, search for schools using various criteria, compare schools side by side, and receive personalized recommendations. The system was designed to provide highly accurate and relevant recommendations, ensuring a seamless and user-friendly experience.

6.1.2 Actual Results

The actual results achieved from the project closely align with the expected results. The developed school search and recommendation system successfully enables users to register and create profiles, providing a platform for personalized school exploration. Users can search for schools based on location, academic programs, facilities, and other relevant criteria. The system accurately retrieves schools that meet the specified criteria and presents them to the users.

The side-by-side school comparison feature allows users to select multiple schools and view detailed information side by side. The system accurately presents the selected schools' data, including information on location, facilities, academic programs, and fees. Users can easily navigate between the compared schools and make informed decisions based on the presented information.

The personalized recommendation engine generates tailored school recommendations based on user profiles and preferences. By considering factors such as location, academic programs, extracurricular activities, and user feedback, the system provides accurate and relevant recommendations. Users can modify their preferences, and the recommendation engine adapts accordingly, ensuring dynamic and up-to-date suggestions.

Overall, the actual results demonstrate that the developed system successfully achieves its intended functionality. Users can register, create profiles, search for schools, compare them side by side, and receive personalized recommendations. The system performs these tasks accurately and efficiently, providing users with a valuable tool for school exploration and decision-making.

6.2 Discussion

While the actual results align closely with the expected results, it is important to discuss the differences between the two and identify potential areas for improvement.

One aspect to consider is the precision of the personalized recommendations. While the system aims to provide highly accurate and relevant school suggestions, certain limitations may affect the precision of the results. These limitations may arise from incomplete or outdated data sources, variations in user preferences and priorities, and the inherent challenge of capturing the full spectrum of a user's requirements. Conducting further research and enhancing data sources can help improve the precision of the recommendations.

Another point of discussion is the system's performance and scalability. While the actual results indicate that the system performs tasks accurately and efficiently, it is important to evaluate its performance under varying loads and concurrent user scenarios. Conducting performance testing with a high number of concurrent users can help identify any performance bottlenecks and ensure the system can scale effectively without compromising its responsiveness and stability.

Usability testing and user feedback are crucial for identifying areas for improvement in the system's interface, navigation, and overall user experience. The discussion should include any usability issues identified during testing sessions and propose potential enhancements to streamline the user interface, improve information presentation, and enhance the overall user experience.

In summary, the actual results achieved closely align with the expected results of the project. The system successfully enables users to register, create profiles, search for schools, compare them side by side, and receive personalized recommendations. However, there may be minor differences and limitations that can be addressed through further research and improvements in data sources, performance scalability, and user experience. The discussion provides insights into these differences and offers recommendations for future enhancements to bridge the gap between the expected and actual results.

Chapter 7: Conclusion

In conclusion, this project has successfully achieved its objectives in developing a comprehensive school search and recommendation system. The system's functionality, including user registration, profile creation, school search based on various criteria, side-by-side school comparison, and personalized recommendations, has been implemented successfully. The actual results closely align with the expected results, showcasing the system's accuracy and efficiency in providing relevant school suggestions to users. By allowing users to make informed decisions when selecting schools, the project has fulfilled its goal of delivering a user-friendly platform for school exploration and decision-making.

Moving forward, there are recommendations to further enhance the project if provided with the right resources. Firstly, expanding and improving the data sources utilized by the system would significantly enhance the accuracy and precision of the personalized recommendations. Incorporating up-to-date data from educational institutions, government databases, and reliable sources would ensure the system remains current and dependable. Additionally, integrating advanced machine learning algorithms and artificial intelligence techniques can further enhance the recommendation engine. By leveraging user feedback, behavior patterns, and academic outcomes, the system can continuously adapt and improve its suggestions, providing users with even more personalized and relevant school recommendations.

Furthermore, investing in performance optimization and scalability is crucial for the long-term success of the project. Conducting thorough performance testing and optimizing the system's architecture would ensure its responsiveness and stability, even under high loads and concurrent user scenarios. This would enable the system to accommodate a larger user base and future growth without compromising its performance.

Chapter 8: Future Work

Looking ahead, there are several areas that can be explored for future work to improve the project. Firstly, incorporating more advanced data analytics and predictive modeling techniques can enhance the accuracy and effectiveness of the personalized recommendations. By analyzing a broader range of variables, such as user demographics, socioeconomic factors, and academic performance, the system can provide more tailored and insightful recommendations to users, helping them find the most suitable schools for their specific needs.

Moreover, expanding the system's functionality to include additional features can enhance the user experience. For instance, integrating user reviews, ratings, and testimonials would provide users with a more comprehensive and holistic view of the schools. This would allow users to consider not only objective criteria but also subjective experiences and opinions, aiding their decision-making process.

Additionally, incorporating social media platforms and leveraging user-generated content can enhance the system's engagement and community aspect. Allowing users to share their experiences, ask questions, and connect with others interested in the same schools would create a vibrant and interactive ecosystem within the system. This would foster a sense of community among users and provide valuable insights and support throughout the school selection process.

Lastly, continuous improvement and maintenance of the system are essential. Regular updates and enhancements should be implemented to keep up with changing user needs, technological advancements, and evolving educational landscapes. Gathering user feedback through surveys, conducting usability testing, and closely monitoring system performance are vital activities to ensure the system remains relevant, user-friendly, and efficient.

In conclusion, the future work for this project involves enhancing the recommendation engine, expanding the system's functionality, incorporating user-generated content, and ensuring continuous improvement and maintenance. With the right resources and strategic investments, the project can further solidify its position as a valuable tool for school exploration and decision-making, providing users with an exceptional experience and helping them make well-informed choices about their education.

Bibliography

The following is a comprehensive list of the materials consulted during the writing of this report. These sources have provided valuable information and insights relevant to the project:

Websites:
1. Ministry of Education and Technical Education, Egypt. (https://www.moe.gov.eg/) - The official website of the Ministry of Education in Egypt, providing information on the education system, policies, and initiatives.

2. Egyptian Knowledge Bank. (https://www.ekb.eg/) - A comprehensive digital library and educational platform in Egypt, offering access to a wide range of educational resources, including textbooks, research papers, and online courses.

3. Schools in Egypt. (https://www.schoolsinegypt.com/) - An online directory of schools in Egypt, providing information on various educational institutions across different cities and regions of the country.

4. International Schools in Egypt. (https://www.internationalschoolsinegypt.com/) - A platform dedicated to international schools in Egypt, offering information on curriculum, admissions, and facilities for expatriate families and students seeking an international education.

5. School Rankings Egypt. (https://www.schoolrankings-eg.com/) - A website providing rankings and profiles of schools in Egypt, based on factors such as academic performance, facilities, and extracurricular offerings.

6. Cairo American College. (https://www.cacegypt.org/) - A leading American international school in Cairo, Egypt, offering an American curriculum and providing educational opportunities for students from preschool to high school.

7. British International School, Cairo. (https://www.bisc.edu.eg /) - A prestigious British international school located in Cairo, Egypt, providing a British curriculum and a wide range of academic and extracurricular programs.

8. Deutsche Schule der Borromäerinnen, Cairo. (https://www.dscairo.de/) - The German School of the Borromean Sisters in Cairo, Egypt, offering a German curriculum and educational opportunities for German-speaking students.