YUGIXX - A WEB TOOL FOR TESTING REST APIs

U23CS492-FULLSTACK DEVELOPMENT PROJECT REPORT
SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE
DEGREE OF **BACHELOR OF ENGINEERING**
IN **COMPUTER SCIENCE AND ENGINEERING**
OF THE ANNA UNIVERSITY

U23CS492-FULLSTACK DEVELOPMENT PROJECT
April/May 2025

PROGRESS THROUGH KNOWLEDGE

PROJECT WORK

Submitted by

NAVEEN PRASANTH P-722823104103

BATCH
2023 – 2027

Under the Guidance of

Mr.B.Saravanan M.E., (Ph.D)

Assistant Professor

Computer Science And Engineering

Leadership & Excellence

Sri Eshwar College of Engineering
(An Autonomous Institution)
Kinathukadavu (Tk), Coimbatore - 641 202, Tamil Nadu
Approved by AICTE, New Delhi and Affiliated to Anna University, Chennai

NAAC NATIONAL ASSESSMENT AND ACCREDITATION COUNCIL
Accredited by NAAC with 'A' Grade
NBA NATIONAL BOARD of ACCREDITATION For CSE, ECE, EEE, MECH
nirf National Institutional Ranking Framework

BONAFIDE CERTIFICATE

Certified that this Report titled **“YUGIXX - A WEB TOOL FOR TESTING REST APIs”** is the bonafide work of

NAVEEN PRASANTH P 722823104103

who carried out the project work under my supervision.

--	--
SIGNATURE	**SIGNATURE**
Dr. R. Subha, M.E, Ph.D.	Mr.B.Saravanan M.E.,(Ph.D)
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Computer Science and Engineering,	Assistant Professor,
Sri Eshwar College of Engineering,	Computer Science and Engineering,
Coimbatore – 641 202.	Sri Eshwar College of Engineering,
	Coimbatore – 641 202.

Submitted for the **Autonomous Semester End Practical Full Stack Web Development** held on…………………..

INTERNAL EXAMINER	**EXTERNAL EXAMINER**

Sri Eshwar College of Engineering
Coimbatore | Tamilnadu
An Autonomous Institution
Affiliated to Anna University, Chennai

QUALITY POLICY

To establish a system of Quality Enhancement, which would on a continuous basis evaluate and enhance the quality of teaching – learning, research and extension activities of the institution, leading to improvements in all processes, enabling the institution to attain excellence.

INSTITUTE VISION

To be recognized as a premier institution, grooming students into globally acknowledged engineering professionals.

INSTITUTE MISSION

- Providing outcome and value-based engineering education
- Nurturing research and entrepreneurial culture
- Enabling students to be industry ready and fulfill their career aspirations
- Grooming students through behavioral and leadership training programs
- Making students socially responsible

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT VISION

To groom students into globally competent software professionals and meet the ever changing requirements of the industry.

DEPARTMENT MISSION

- Creating a quality academic environment with relevant IT infrastructure and empowering faculty and students with emerging technologies.
- Motivating staff and students to actively involve in lifelong learning and fostering research.
- Inculcating leadership and entrepreneurship skills in students.
- Generating opportunities for students to evolve as competent software professionals with societal consciousness.

PROGRAM EDUCATIONAL OBJECTIVES

PEO1: To prepare graduates for a career in software engineering

PEO2: To prepare students for higher studies, research, entrepreneurial and leadership roles by imparting the quality of lifelong learning

PEO3: To enable students to apply innovative solutions for real-life problems in computer science domain.

PROGRAM OUTCOMES

PO1: Engineering Knowledge: Apply knowledge of mathematics, natural science, computing, engineering fundamentals and an engineering specialization as specified in WK1 to WK4 respectively to develop to the solution of complex engineering problems.

PO2: Problem Analysis: Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development. (WK1 to WK4)

PO3: Design/Development of Solutions: Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required. (WK5)

PO4: Conduct Investigations of Complex Problems: Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions. (WK8).

PO5: Engineering Tool Usage: Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems. (WK2 and WK6)

PO6: The Engineer and The World: Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment. (WK1, WK5, and WK7).

PO7: Ethics: Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws. (WK9)

PO8: Individual and Collaborative Team work: Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.

PO9: Communication: Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and learning differences

PO10: Project Management and Finance: Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments.

PO11: Life-Long Learning: Recognize the need for, and have the preparation and ability for i) independent and life-long learning ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological change. (WK8)

PROGRAM SPECIFIC OUTCOMES

PSO1. Demonstrate knowledge in open source technologies

PSO2. Develop innovative solutions by adapting emerging technologies for industry oriented applications.

PSO3. Implement SDLC principles for project/product development.

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

We thank God for his grace and blessing that have been showered upon us to bring success for this project. The success we have accomplished in our project would not have been possible without the timely help and guidance rendered to us by many people. We take this opportunity to thank them all.

We wish to express our deep gratitude **Mr. R. Mohanram**, Chairman, Sri Eshwar College of Engineering, Kondampatti for being kindly permitted us to carry on the project. He has an abiding interest in our wellbeing.

We are grateful to **Dr. Sudha Mohanram M.E., Ph.D** Principal, Sri Eshwar College of Engineering, Kondampatti for having made all provision available to us.

We are thankful to **Mr. R. Rajaram**, Director, Sri Eshwar College of Engineering, Kondampatti for all his support to complete the project effectively.

We would like to express our heartfelt gratitude and humble thanks to Head of the Department **Dr. R. Subha M.E., Ph.D** for having facilitated us to complete the project successfully.

With the deep sense of gratitude, we extend our earnest and sincere thanks to our project guide **Mr. B. Saravanan M.E., (Ph.D),** Assistant Professor, Department of Computer Science and Engineering, for his kind suggestion and co-operation for doing the project. Finally, we would like to extend our thanks to our teaching and non-teaching staffs those who helped us in bringing out project successfully.

ABSTRACT

YUGIXX is a full-stack web application developed using the MERN stack (MongoDB, Express.js, React.js, and Node.js) designed to provide developers and testers with a user-friendly, browser-based tool for testing RESTful APIs. The application allows users to securely register and log in, offering personalized access to their saved API requests and testing history. Once authenticated, users can construct and send HTTP requests (GET, POST, PUT, DELETE, etc.) with full control over headers, query parameters, and JSON bodies. The core functionality of YUGIXX revolves around real-time request handling and response display, enabling users to inspect and debug API behavior quickly and efficiently. The frontend, built with React, offers a dynamic and responsive interface, while the backend, powered by Node.js and Express.js, handles request execution and data storage. MongoDB ensures reliable persistence of user data, including saved requests and session information. YUGIXX is designed to serve as a lightweight and accessible alternative to desktop tools like Postman, making it ideal for API-first development workflows. The platform is scalable and structured to support future enhancements such as automated testing scripts, token authentication (JWT, OAuth), environment variable support, and collaboration features..

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 OBJECTIVES

The objective of **YUGIXX** is to develop a secure, efficient, and user-friendly web application that enables users to test and debug RESTful APIs directly from the browser. The app aims to provide personalized user experiences through secure login and registration features, allowing users to create, edit, and manage API requests. It is designed to streamline API testing through features like customizable request methods, headers, and JSON payloads. The application also seeks to offer a responsive design for seamless access across devices and uses a scalable architecture built with the MERN stack (MongoDB, Express.js, React.js, Node.js) to ensure smooth performance, modularity, and future extensibility.

1.2 SCOPE OF THE PROJECT

The scope of the project includes the development of a full-stack web application that supports user authentication, a dashboard, and real-time notifications. Users can manage multiple api to test details such as API link, key, and value for Params and Headers, and JSON. The data is securely stored in MongoDB, ensuring persistence and reliability. The application currently functions as a test and management tool but is designed with scalability in mind, allowing future integration of additional features such as prescription uploads, consultation modules, testing history tracking, and mobile app compatibility. It primarily serves individuals who require assistance in adhering to their medication routines**.**

CHAPTER 2
SYSTEM ANALYSIS AND SPECIFICATION

2.1 PROBLEM DESCRIPTION

In today's rapidly evolving software development landscape, developers and QA engineers often face challenges when testing and debugging RESTful APIs. Traditional desktop-based API testing tools can be complex, require installation, and lack accessibility across devices. This creates inefficiencies, especially for teams working remotely or on cloud platforms. There is a growing need for a lightweight, web-based solution that enables users to easily construct, send, and analyze API requests from anywhere with an internet connection. The YUGIXX project aims to address this problem by delivering a secure, intuitive, and responsive web application that streamlines API testing, enhances productivity, and integrates seamlessly into modern development workflows.

2.2 FUNCTIONAL REQUIREMENT –

SOFTWARE AND HARDWARE REQUIREMENTS

- **Frontend:** React.js, HTML, CSS, JavaScript, React Router
- **Backend:** Node.js, Express.js, RESTful APIs
- **Database:** MongoDB (Local or Atlas)
- **Authentication:** bcrypt.js for password hashing, express-session or JWT
- **Notifications:** Browser Notifications (Web APIs)

2.3 NON-FUNCTIONAL REQUIREMENT

- **Usability**: The user interface should be simple, intuitive, and accessible across all devices.
- **Scalability**: The system should handle increasing users and features without performance issues.
- **Performance**: The app should load and respond quickly, even with multiple users

CHAPTER 3

SYSTEM DESIGN

USECASE DIAGRAM

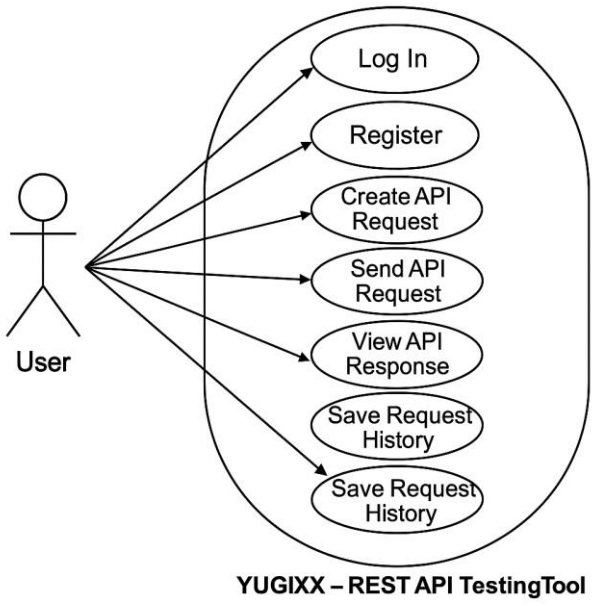


FIGURE 1 USECASE DIAGRAM

CLASS DIAGRAM

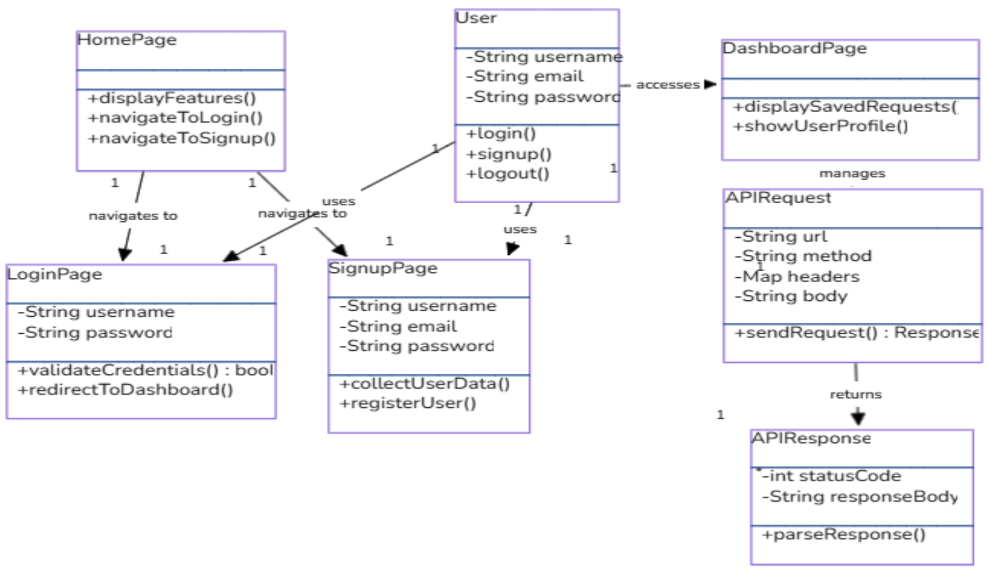


FIGURE 2 CLASS DIAGRAM

CHAPTER 4

MODULE DESCRIPTION

4.1 APP MODULE

This is the main React component that serves as the root of the application. It manages routing and renders other components such as Login, Register, and API Testing. It also links the CSS styles and controls the overall UI flow.

4.2 COMPONENT MODULE

4.2.1 Login Component

Responsible for user login functionality. It provides the login form, handles user input for email and password, and interacts with the backend to authenticate users**.**

4.2.2 Register Component

Handles new user registration. It collects user details such as username, email, and password, performs validation, and sends the data to the backend API for creating a new user account**.**

4.2.3 API Resquest Component

This core module allows users to create, view, edit, and delete their REST API test requests. It includes forms for inputting API details such as endpoint URL, HTTP method (GET, POST, PUT, DELETE), headers, parameters, and JSON body. The component displays a list of saved or past requests along with their responses for easy reference and reusability

CHAPTER 4

IMPLEMENTATION

COMPONENTS:

DASHBOARD.JS

```
import { useState, useEffect } from 'react';
import { useNavigate, useLocation } from 'react-router-dom';
import '../styles/Dashboard.css';
import {
  FiPlus,
  FiFolder,
  FiClock,
  FiSettings,
  FiTrendingUp,
  FiCheckCircle,
  FiXCircle,
  FiActivity,
  FiBarChart2,
  FiRefreshCw,
  FiExternalLink,
  FiChevronRight,
  FiZap,
  FiAlertCircle,
  FiPieChart,
  FiUsers,
  FiGlobe,
  FiLogOut
} from 'react-icons/fi';

const ApiTestForm = ({ onSubmit }) => {
  const [formData, setFormData] = useState({
    url: '',
    method: 'GET',
    headers: {},
    params: {},
    body: ''
  });

  const [newHeader, setNewHeader] = useState({ key: '', value: '' });
  const [newParam, setNewParam] = useState({ key: '', value: '' });
  const [loading, setLoading] = useState(false);
  const [response, setResponse] = useState(null);
  const [error, setError] = useState(null);

  const handleMethodChange = (e) => {
    setFormData({ ...formData, method: e.target.value });
  };

  const handleUrlChange = (e) => {
    setFormData({ ...formData, url: e.target.value });
  };

  const addHeader = () => {
    if (newHeader.key && newHeader.value) {
      setFormData({
        ...formData,
        headers: {
          ...formData.headers,
          [newHeader.key]: newHeader.value
        }
      });
      setNewHeader({ key: '', value: '' });
    }
  };

  const addParam = () => {
    if (newParam.key && newParam.value) {
      setFormData({
        ...formData,
        params: {
          ...formData.params,
          [newParam.key]: newParam.value
        }
      });
      setNewParam({ key: '', value: '' });
    }
  };

  const handleSubmit = async () => {
    setLoading(true);
    setError(null);
    setResponse(null);
    try {
      // Build URL with parameters
      const url = new URL(formData.url);
      Object.entries(formData.params).forEach(([key, value]) => {
        url.searchParams.append(key, value);
      });

      // Prepare request options
      const options = {
        method: formData.method,
        headers: {
          ...formData.headers,
          'Content-Type': 'application/json'
        }
      };

      // Add body for POST, PUT, PATCH methods
      if (['POST', 'PUT', 'PATCH'].includes(formData.method) && formData.body) {
        options.body = formData.body;
      }

      // Make the API call
      const response = await fetch(url.toString(), options);
      const data = await response.json();

      setResponse({
        status: response.status,
        statusText: response.statusText,
        headers: Object.fromEntries(response.headers.entries()),
        data
      });
    } catch (err) {
      setError(err.message);
    } finally {
      setLoading(false);
    }
  };

  return (
    <div className="api-test-form-container">
      <div className="api-test-form-content">
        <div className="form-group">
    <label>Method</label>
    <select value={formData.method} onChange={handleMethodChange}>
      <option value="GET">GET</option>
      <option value="POST">POST</option>
      <option value="PUT">PUT</option>
      <option value="DELETE">DELETE</option>
      <option value="PATCH">PATCH</option>
    </select>
  </div>

  <div className="form-group">
    <label>URL</label>
    <input
      type="text"
      value={formData.url}
      onChange={handleUrlChange}
      placeholder="Enter API endpoint URL"
    />
  </div>

  <div className="form-section">
    <h4>Headers</h4>
    <div className="key-value-inputs">
      <input
        type="text"
        placeholder="Header name"
        value={newHeader.key}
        onChange={(e) => setNewHeader({ ...newHeader, key: e.target.value })}
      />
      <input
        type="text"
        placeholder="Header value"
        value={newHeader.value}
        onChange={(e) => setNewHeader({ ...newHeader, value: e.target.value })}
      />
      <button onClick={addHeader} className="add-button">Add</button>
    </div>
    <div className="key-value-list">
      {Object.entries(formData.headers).map(([key, value]) => (
        <div key={key} className="key-value-item">
          <strong>{key}:</strong> {value}
        </div>
      ))}
```

```
          </div>
        </div>

        <div className="form-section">
              </div>
        </div>

        <div className="dashboard-card system-status">
          <div className="card-header">
            <h2>System Status</h2>
          </div>
          <div className="status-grid">
            <div className="status-item">
              <div className="status-icon uptime">
                <FiCheckCircle />
              </div>
              <div className="status-info">
                <h3>Uptime</h3>
                <p>{stats.uptime}%</p>
              </div>
            </div>
            <div className="status-item">
              <div className="status-icon error-rate">
                <FiAlertCircle />
              </div>
              <div className="status-info">
                <h3>Error Rate</h3>
                <p>{stats.errorRate}%</p>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  );
};

export default Dashboard;
```

DASHBOARD.CSS

```
.dashboard-container {
  display: flex;
  flex-direction: column;
  height: 100vh;
  background: linear-gradient(to right, #74ebd5, #acb6e5);
}

.header {
  background-color: #007bff;
  color: white;
  padding: 10px 20px;
  display: flex;
  justify-content: space-between;
  align-items: center;
}

.app-name {
  font-weight: bold;
  font-size: 20px;
}

.auth-buttons {
  display: flex;
  gap: 15px;
}

.auth-buttons button {
  padding: 12px 25px;
  font-size: 16px;
  color: white;
  background-color: #007bff;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  transition: background-color 0.3s ease, transform 0.2s ease;
}

.auth-buttons button:hover {
  background-color: #0056b3;
  transform: translateY(-2px);
}

.auth-buttons button:active {
  transform: translateY(2px);
```

```
}

.auth-buttons button:focus {
  outline: none;
}

.content {
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  flex-grow: 1;
  padding: 20px;
  text-align: center;
}

h1 {
  font-size: 28px;
  color: #333;
  margin-bottom: 20px;
}

p {
  font-size: 18px;
  color: #555;
  margin-bottom: 30px;
}

.dashboard-image {
  max-width: 100%;
  height: auto;
  margin-bottom: 30px;
  border-radius: 8px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

.get-started-btn {
  padding: 12px 30px;
  font-size: 18px;
  background-color: #007bff;
  .dashboard-container {
    display: flex;
    flex-direction: column;
    height: 100vh;
    background: linear-gradient(to right, #74ebd5, #acb6e5);
  }

  .header {
    background-color: #007bff;
    color: white;
    padding: 10px 20px;
    display: flex;
    justify-content: space-between;
    align-items: center;
  }

  .app-name {
    font-weight: bold;
    font-size: 20px;
  }

  .auth-buttons {
    display: flex;
    gap: 15px;
  }

  .auth-buttons button {
    padding: 12px 25px;
    font-size: 16px;
    color: white;
    background-color: #007bff;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s ease, transform 0.2s ease;
  }

  .auth-buttons button:hover {
    background-color: #0056b3;
    transform: translateY(-2px);
  }

  .auth-buttons button:active {
    transform: translateY(2px);
  }
```

```
.auth-buttons button:focus {
  outline: none;
}


.content {
  display: flex;
  flex-direction: row;
  justify-content: space-between;
  align-items: center;
  flex-grow: 1;
  padding: 20px;
  text-align: left;
}

.text-content {
  flex: 1;
  padding-right: 20px;
}

h1 {
  font-size: 28px;
  color: #333;
  margin-bottom: 20px;
}

p {
  font-size: 18px;
  color: #555;
  margin-bottom: 30px;
}


.dashboard-image {
  max-width: 100%;

  border-radius: 8px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}


.get-started-btn {
  padding: 12px 30px;
```

```
    font-size: 18px;
    background-color: #007bff;
    color: white;
    border: none;
    border-radius: 8px;
    cursor: pointer;
    transition: background-color 0.3s ease, transform 0.2s ease;
    margin-top: 20px;
    width: auto;
  }

  .get-started-btn:hover {
    background-color: #0056b3;
    transform: translateY(-2px);
  }

  .get-started-btn:active {
    transform: translateY(2px);
  }

  .get-started-btn:focus {
    outline: none;
  }


  .footer {
    color: white;
    text-align: center;
    padding: 10px 20px;
    margin-bottom: 30px;
  }

  .footer p {
    margin: 0;
    font-size: 14px;
  }

  color: white;
  border: none;
  border-radius: 8px;
  cursor: pointer;
  transition: background-color 0.3s ease, transform 0.2s ease;
}
```

```
.get-started-btn:hover {
  background-color: #0056b3;
  transform: translateY(-2px);
}

.get-started-btn:active {
  transform: translateY(2px);
}

.get-started-btn:focus {
  outline: none;
}

.footer {
  color: white;
  text-align: center;
  padding: 10px 20px;
  margin-bottom: 30px;
}

.footer p {
  margin: 0;
  font-size: 14px;
}
```

LOGIN.JS

```
import { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import axios from 'axios';

const Login = ({ setIsAuthenticated }) => {
  const [formData, setFormData] = useState({
    email: '',
    password: ''
  });
  const [error, setError] = useState('');
  const navigate = useNavigate();

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

const handleSubmit = async (e) => {
  e.preventDefault();
  setError(''); // Clear any previous errors
  try {
    console.log('Attempting login with:', formData);
    const response = await axios.post('http://localhost:5000/api/login', formData, {
      withCredentials: true,
      headers: {
        'Content-Type': 'application/json'
      }
    });
    console.log('Login response:', response.data);
    if (response.data.token) {
      localStorage.setItem('token', response.data.token);
      setIsAuthenticated(true);
      navigate('/dashboard');
    } else {
      setError('No token received from server');
    }
  } catch (err) {
    console.error('Login error details:', err);
    if (err.response) {
      // The request was made and the server responded with a status code
      // that falls out of the range of 2xx
      setError(err.response.data.message || 'Login failed. Please check your credentials.');
    } else if (err.request) {
      // The request was made but no response was received
      setError('No response from server. Please check if the server is running.');
    } else {
      // Something happened in setting up the request that triggered an Error
      setError('An error occurred while setting up the request.');
    }
  }
};

return (
  <div className="auth-container">
    <div className="auth-form">
      <h2>Login to YUGIXX</h2>
      {error && <div className="error-message">{error}</div>}
```

```
        <form onSubmit={handleSubmit}>
          <div className="form-group">
            <label htmlFor="email">Email</label>
            <input
              type="email"
              id="email"
              name="email"
              value={formData.email}
              onChange={handleChange}
              required
            />
          </div>
          <div className="form-group">
            <label htmlFor="password">Password</label>
            <input
              type="password"
              id="password"
              name="password"
              value={formData.password}
              onChange={handleChange}
              required
            />
          </div>
          <button type="submit" className="btn btn-primary">Login</button>
        </form>
        <p className="auth-link">
          Don't have an account? <a href="/signup">Sign up</a>
        </p>
      </div>
    </div>
  );
};

export default Login;
```

REGISTER.JS

```
import { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import axios from 'axios';

const Signup = () => {
```

```
const [formData, setFormData] = useState({
  name: '',
  email: '',
  password: ''
});
const [error, setError] = useState('');
const [loading, setLoading] = useState(false);
const navigate = useNavigate();

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

const validateForm = () => {
  if (!formData.name.trim()) {
    setError('Name is required');
    return false;
  }
  if (!formData.email.trim()) {
    setError('Email is required');
    return false;
  }
  if (!formData.password) {
    setError('Password is required');
    return false;
  }
  if (formData.password.length < 6) {
    setError('Password must be at least 6 characters long');
    return false;
  }
  return true;
};

const handleSubmit = async (e) => {
  e.preventDefault();
  setError('');

  if (!validateForm()) {
    return;
  }

  setLoading(true);
  try {
    console.log('Attempting signup with:', { ...formData, password: '***' });
    const response = await axios.post('http://localhost:5000/api/signup', formData, {
      headers: {
        'Content-Type': 'application/json'
      }
    });
    console.log('Signup response:', response.data);

    if (response.data.message === 'User created successfully') {
      navigate('/login');
    } else {
      setError('Unexpected response from server');
    }
  } catch (err) {
    console.error('Signup error:', err);
    if (err.response) {
      setError(err.response.data.message || 'Signup failed. Please try again.');
    } else if (err.request) {
      setError('No response from server. Please check if the server is running.');
    } else {
      setError('An error occurred while setting up the request.');
    }
  } finally {
    setLoading(false);
  }
};

return (
  <div className="auth-container">
    <div className="auth-form">
      <h2>Create Your YUGIXX Account</h2>
      {error && <div className="error-message">{error}</div>}
      <form onSubmit={handleSubmit}>
        <div className="form-group">
          <label htmlFor="name">Name</label>
          <input
            type="text"
            id="name"
            name="name"
            value={formData.name}
            onChange={handleChange}
            required
            minLength="2"
          />
        </div>
        <div className="form-group">
          <label htmlFor="email">Email</label>
          <input
            type="email"
            id="email"
            name="email"
            value={formData.email}
            onChange={handleChange}
            required
          />
        </div>
        <div className="form-group">
          <label htmlFor="password">Password</label>
          <input
            type="password"
            id="password"
            name="password"
            value={formData.password}
            onChange={handleChange}
            required
            minLength="6"
          />
        </div>
        <button
          type="submit"
          className="btn btn-primary"
          disabled={loading}
        >
          {loading ? 'Signing up...' : 'Sign Up'}
        </button>
      </form>
      <p className="auth-link">
        Already have an account? <a href="/login">Login</a>
      </p>
    </div>
  </div>
 );
};

export default Signup;
```

HOME.CSS

```
.home {
  min-height: 100vh;
  background: linear-gradient(135deg, #f8f9fa 0%, #e9ecef 100%);
}

/* Hero Section */
.hero-section {
  padding: 80px 20px;
  background: linear-gradient(135deg, #ffffff 0%, #f8f9fa 100%);
  position: relative;
  overflow: hidden;
}
```

```
.hero-section::before {
  content: '';
  position: absolute;
  top: 0;
  left: 0;
  right: 0;
  height: 100%;
  background: url('data:image/svg+xml,<svg xmlns="http://www.w3.org/2000/svg"
viewBox="0 0 100 100"><circle cx="50" cy="50" r="40" fill="none" stroke="%234a90e2"
stroke-width="0.5" opacity="0.1"/></svg>') center/cover;
  opacity: 0.1;
}

.hero-content {
  max-width: 1200px;
  margin: 0 auto;
  display: grid;
  grid-template-columns: 1fr 1fr;
  gap: 40px;
  align-items: center;
}

.hero-text {
  position: relative;
  z-index: 1;
}

.hero-title {
  font-size: 3.5rem;
  font-weight: 800;
  line-height: 1.2;
  margin-bottom: 20px;
  color: #2c3e50;
}

.gradient-text {
  background: linear-gradient(135deg, #4a90e2 0%, #357abd 100%);
  -webkit-background-clip: text;
  -webkit-text-fill-color: transparent;
  background-clip: text;
}

.hero-subtitle {
  font-size: 1.2rem;
  color: #6c757d;
  margin-bottom: 30px;
  line-height: 1.6;
}

.hero-cta {
  display: flex;
  gap: 20px;
}

.cta-button {
  display: inline-flex;
  align-items: center;
  justify-content: center;
  padding: 12px 24px;
  border-radius: 8px;
  font-weight: 600;
  font-size: 1rem;
  transition: all 0.3s ease;
  text-decoration: none;
}

.cta-button.primary {
  background: linear-gradient(135deg, #4a90e2 0%, #357abd 100%);
  color: white;
  box-shadow: 0 4px 6px rgba(74, 144, 226, 0.2);
}

.cta-button.primary:hover {
  transform: translateY(-2px);
  box-shadow: 0 6px 12px rgba(74, 144, 226, 0.3);
}

.cta-button.secondary {
  background: white;
  color: #4a90e2;
  border: 2px solid #4a90e2;
}

.cta-button.secondary:hover {
  background: #f8f9fa;
  transform: translateY(-2px);
}

.hero-stats {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  gap: 20px;
  background: white;
  padding: 30px;
  border-radius: 16px;
  box-shadow: 0 10px 30px rgba(0, 0, 0, 0.05);
}

.stat-item {
  text-align: center;
  padding: 20px;
  border-radius: 12px;
  background: #f8f9fa;
  transition: all 0.3s ease;
}

.stat-item:hover {
  transform: translateY(-5px);
  box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1);
}

.stat-item svg {
  font-size: 2rem;
  color: #4a90e2;
  margin-bottom: 10px;
```

REQUEST BUILDING.JS

```
import { useState } from 'react';
import axios from 'axios';
import '../styles/RequestBuilder.css';

const RequestBuilder = () => {
  const [request, setRequest] = useState({
    url: '',
    method: 'GET',
    headers: [{ key: '', value: '' }],
    params: [{ key: '', value: '' }],
    body: '',
    authType: 'none',
    authToken: '',
    apiKey: '',
  });

  const [response, setResponse] = useState(null);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState('');

  const handleInputChange = (e) => {
    const { name, value } = e.target;
    setRequest(prev => ({ ...prev, [name]: value }));
  };

  const handleKeyValueChange = (type, index, field, value) => {
    setRequest(prev => ({
      ...prev,
      [type]: prev[type].map((item, i) =>
        i === index ? { ...item, [field]: value } : item
      )
    }));
  };

  const addKeyValuePair = (type) => {
    setRequest(prev => ({
      ...prev,
      [type]: [...prev[type], { key: '', value: '' }]
    }));
  };

  const removeKeyValuePair = (type, index) => {
    setRequest(prev => ({
      ...prev,
      [type]: prev[type].filter((_, i) => i !== index)
    }));
  };

            {request.authType === 'apiKey' && (
              <input
                type="text"
                name="apiKey"
                value={request.apiKey}
                onChange={handleInputChange}
                placeholder="Enter API Key"
              />
            )}
          </div>
        </div>
      </div>
    </form>

    {error && <div className="error-message">{error}</div>}

    {response && (
      <div className="response-section">
        <div className="response-header">
          <span className={`status-code ${response.status >= 200 && response.status < 300 ?
'success' : 'error'}`}>
            {response.status} {response.statusText}
          </span>
          <span className="response-time">Time: {response.time}ms</span>
        </div>

        <div className="response-tabs">
          <div className="tab-section">
            <h3>Response Body</h3>
            <pre>{JSON.stringify(response.data, null, 2)}</pre>
          </div>

          <div className="tab-section">
            <h3>Response Headers</h3>
```

```
              <pre>{JSON.stringify(response.headers, null, 2)}</pre>
            </div>
          </div>
        </div>
      )}
    </div>
  );
};

export default RequestBuilder;
```

MEDICINEREMINDER.CSS

```
.request-builder {
  padding: 2rem;
  max-width: 1200px;
  margin: 0 auto;
}

.request-section {
  background: white;
  border-radius: 8px;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  margin-bottom: 2rem;
}

.url-section {
  display: flex;
  gap: 1rem;
  padding: 1rem;
  border-bottom: 1px solid #eee;
}

.url-section select {
  padding: 0.5rem;
  border: 1px solid #ddd;
  border-radius: 4px;
  min-width: 100px;
}

.url-section input {
  flex: 1;
  padding: 0.5rem;
  border: 1px solid #ddd;
  border-radius: 4px;
}

.url-section button {
  padding: 0.5rem 1rem;
  background: #1a73e8;
  color: white;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}

.url-section button:disabled {
  background: #ccc;
  cursor: not-allowed;
}

.tabs {
  padding: 1rem;
}

.tab-section {
  margin-bottom: 1.5rem;
}

.tab-section h3 {
  margin-bottom: 1rem;
  color: #333;
}

.key-value-pair {
  display: flex;
  gap: 1rem;
  margin-bottom: 0.5rem;
}

.key-value-pair input {
  flex: 1;
  padding: 0.5rem;
  border: 1px solid #ddd;
  border-radius: 4px;
}

.key-value-pair button {
  padding: 0.5rem;
  background: #f1f3f4;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}
```

```
.tab-section button {
  padding: 0.5rem 1rem;
  background: #f1f3f4;
  border: none;
  border-radius: 4px;
  cursor: pointer;
  margin-top: 0.5rem;
}

textarea {
  width: 100%;
  padding: 0.5rem;
  border: 1px solid #ddd;
  border-radius: 4px;
  font-family: monospace;
}

.response-section {
  background: white;
  border-radius: 8px;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  padding: 1rem;
}

.response-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 1rem;
  padding-bottom: 1rem;
  border-bottom: 1px solid #eee;
}

.status-code {
  padding: 0.25rem 0.5rem;
  border-radius: 4px;
  font-weight: bold;
}

.status-code.success {
  background: #e6f4ea;
  color: #137333;
}

.status-code.error {
  background: #fce8e6;
  color: #c5221f;
}
```

```
.response-time {
  color: #666;
}

.response-tabs {
  display: grid;
  grid-template-columns: 1fr;
  gap: 1rem;
}

pre {
  background: #f8f9fa;
  padding: 1rem;
  border-radius: 4px;
  overflow-x: auto;
  }
```

APP.JS

```
import { BrowserRouter as Router, Routes, Route, Navigate } from 'react-router-dom';
import { useState } from 'react';
import Navbar from './components/Navbar';
import Home from './pages/Home';
import Login from './pages/Login';
import Signup from './pages/Signup';
import Dashboard from './pages/Dashboard';
import RequestBuilder from './pages/RequestBuilder';
import Collections from './pages/Collections';
import TestSuites from './pages/TestSuites';
import History from './pages/History';
import Settings from './pages/Settings';
import Analytics from './pages/Analytics';
import Monitoring from './pages/Monitoring';
import './App.css';

function App() {
  const [isAuthenticated, setIsAuthenticated] = useState(false);

  return (
    <Router>
      <div className="app">
        <Navbar isAuthenticated={isAuthenticated} setIsAuthenticated={setIsAuthenticated} />
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/login" element={<Login setIsAuthenticated={setIsAuthenticated} />} />
          <Route path="/signup" element={<Signup />} />
          <Route
            path="/dashboard"
            element={
              isAuthenticated ?
                <Dashboard /> :
```

```
      <Navigate to="/login" replace />
  }
/>
<Route
  path="/builder"
  element={
    isAuthenticated ?
      <RequestBuilder /> :
      <Navigate to="/login" replace />
  }
/>
<Route
  path="/collections"
  element={
    isAuthenticated ?
      <Collections /> :
      <Navigate to="/login" replace />
  }
/>
<Route
  path="/test-suites"
  element={
    isAuthenticated ?
      <TestSuites /> :
      <Navigate to="/login" replace />
  }
/>
<Route
  path="/history"
  element={
    isAuthenticated ?
      <History /> :
      <Navigate to="/login" replace />
  }
/>
<Route
  path="/settings"
  element={
    isAuthenticated ?
      <Settings /> :
      <Navigate to="/login" replace />
  }
/>
<Route
  path="/analytics"
  element={
    isAuthenticated ?
      <Analytics /> :
      <Navigate to="/login" replace />
  }
```

```
        />
        <Route
          path="/monitoring"
          element={
            isAuthenticated ?
              <Monitoring /> :
              <Navigate to="/login" replace />
          }
        />
      </Routes>
    </div>
  </Router>
 );
}

export default App;
```

APP.CSS

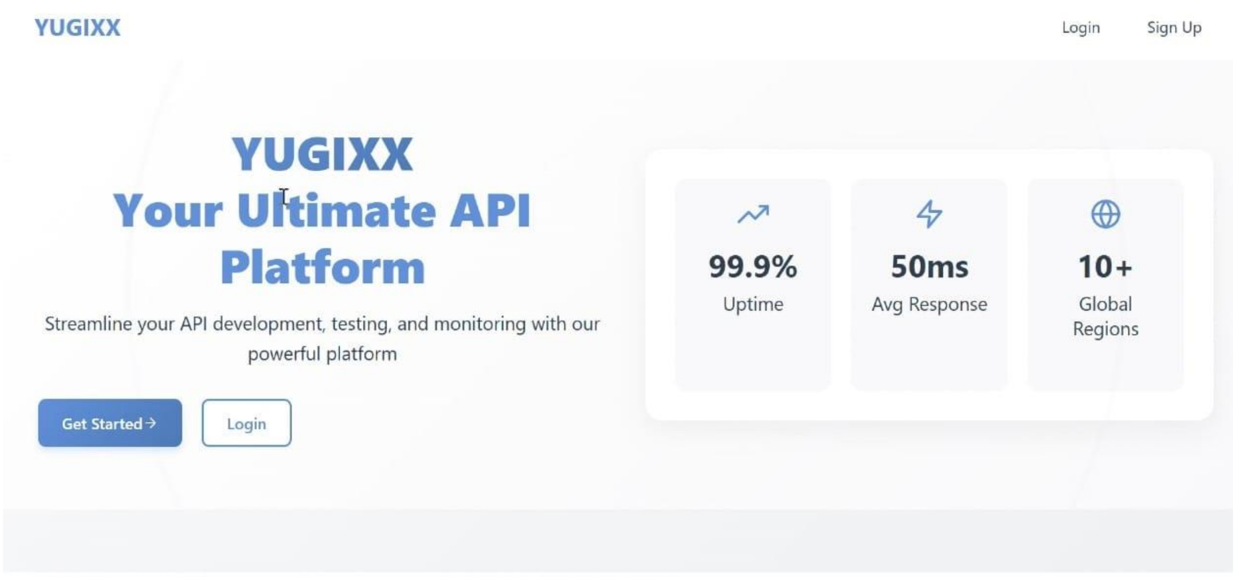
```
/* Global Styles */
:root {
  --primary-color: #3b82f6; /* Softer blue for better accessibility */
  --secondary-color: #1f2937; /* Darker gray for contrast */
  --accent-color: #10b981; /* Green for success/accent */
  --error-color: #ef4444; /* Softer red for errors */
  --background-color: #f9fafb; /* Lighter background for modern look */
  --text-color: #111827; /* Darker text for readability */
  --card-bg: #ffffff; /* White for cards */
  --shadow-color: rgba(0, 0, 0, 0.05);
  --border-color: #e5e7eb;
  --hover-transition: all 0.3s ease-in-out;
}

* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body {
  font-family: 'Inter', -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, sans-serif;
  background-color: var(--background-color);
  color: var(--text-color);
  line-height: 1.6;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

/* Navbar Styles */
.navbar {
  background: linear-gradient(90deg, var(--card-bg), #f1f5f9);
  padding: 1.25rem 2rem;
  box-shadow: 0 4px 12px var(--shadow-color);
  display: flex;
  justify-content: space-between;
  align-items: center;
  position: sticky;
  top: 0;
  z-index: 1000;
}

.navbar-brand a {
  font-size: 1.75rem;
  font-weight: 700;
  color: var(--primary-color);
  text-decoration: none;
  letter-spacing: -0.025em;
}

.navbar-links {
  display: flex;
  gap: 1.5rem;
  align-items: center;
}

.navbar-links a {
  color: var(--text-color);
  text-decoration: none;
  padding: 0.5rem 1rem;
  border-radius: 6px;
  font-weight: 500;
  transition: var(--hover-transition);
}

.navbar-links a:hover {
  background-color: var(--primary-color);
  color: white;
  transform: translateY(-1px);
}

.navbar-links button {
  background: linear-gradient(135deg, var(--primary-color), #2563eb);
  color: white;
  border: none;
  padding: 0.5rem 1.25rem;
  border-radius: 6px;
  font-weight: 500;
  cursor: pointer;
  transition: var(--hover-transition);
  box-shadow: 0 2px 8px rgba(59, 130, 246, 0.2);
}
```

```
.navbar-links button:hover {
  background: linear-gradient(135deg, #2563eb, var(--primary-color));
  transform: translateY(-2px);
  box-shadow: 0 4px 12px rgba(59, 130, 246, 0.3);
}

/* Home Page Styles */
.home-container {
  max-width: 1280px;
  margin: 0 auto;
  padding: 3rem 1.5rem;
}

.hero-section {
  text-align: center;
  padding: 5rem 0;
  background: linear-gradient(180deg, rgba(59, 130, 246, 0.05), transparent);
  border-radius: 12px;
}

.hero-section h1 {
  font-size: 3.5rem;
  font-weight: 800;
  margin-bottom: 1.25rem;
  color: var(--primary-color);
  letter-spacing: -0.05em;
}

.hero-section p {
  font-size: 1.25rem;
  margin-bottom: 2.5rem;
  color: var(--secondary-color);
  max-width: 600px;
  margin-left: auto;
  margin-right: auto;
}

.cta-buttons {
  display: flex;
  gap: 1.25rem;
  justify-content: center;
  flex-wrap: wrap;
}

.features-section {
  margin-top: 5rem;
}

.features-section h2 {
```

```css
  text-align: center;
  margin-bottom: 3rem;
  font-size: 2.25rem;
  font-weight: 700;
  color: var(--text-color);
}

.features-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(320px, 1fr));
  gap: 2rem;
}

.feature-card {
  background: var(--card-bg);
  padding: 2rem;
  border-radius: 12px;
  box-shadow: 0 4px 12px var(--shadow-color);
  transition: var(--hover-transition);
}

.feature-card:hover {
  transform: translateY(-4px);
  box-shadow: 0 8px 24px var(--shadow-color);
}

.feature-card h3 {
  color: var(--primary-color);
  margin-bottom: 1rem;
  font-size: 1.5rem;
  font-weight: 600;
}

/* Auth Pages Styles */
.auth-container {
  display: flex;
  justify-content: center;
  align-items: center;
  min-height: calc(100vh - 80px);
  padding: 2rem;
  background: linear-gradient(180deg, var(--background-color), #e5e7eb);
}
```

CHAPTER 6

RESULTS AND DISCUSSIONS



YUGIXX
Login
Sign Up
YUGIXX
Your Ultimate API Platform
Streamline your API development, testing, and monitoring with our powerful platform
Get Started →
Login
99.9%
Uptime
50ms
Avg Response
10+
Global Regions

FIGURE 3 Home Page

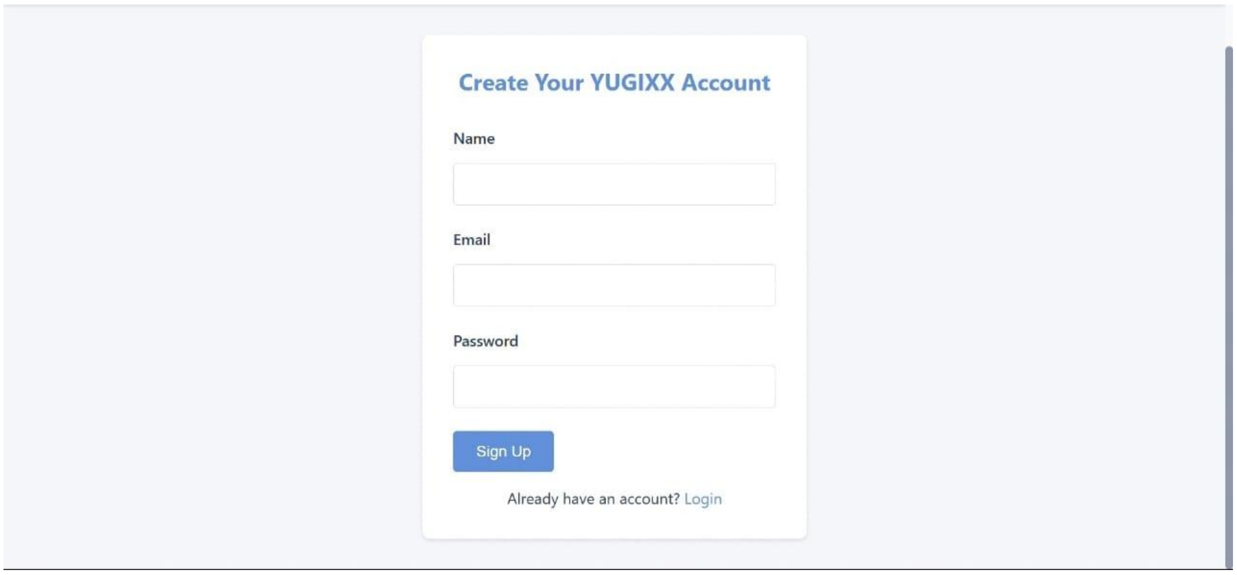


FIGURE 4 Sign Up Page

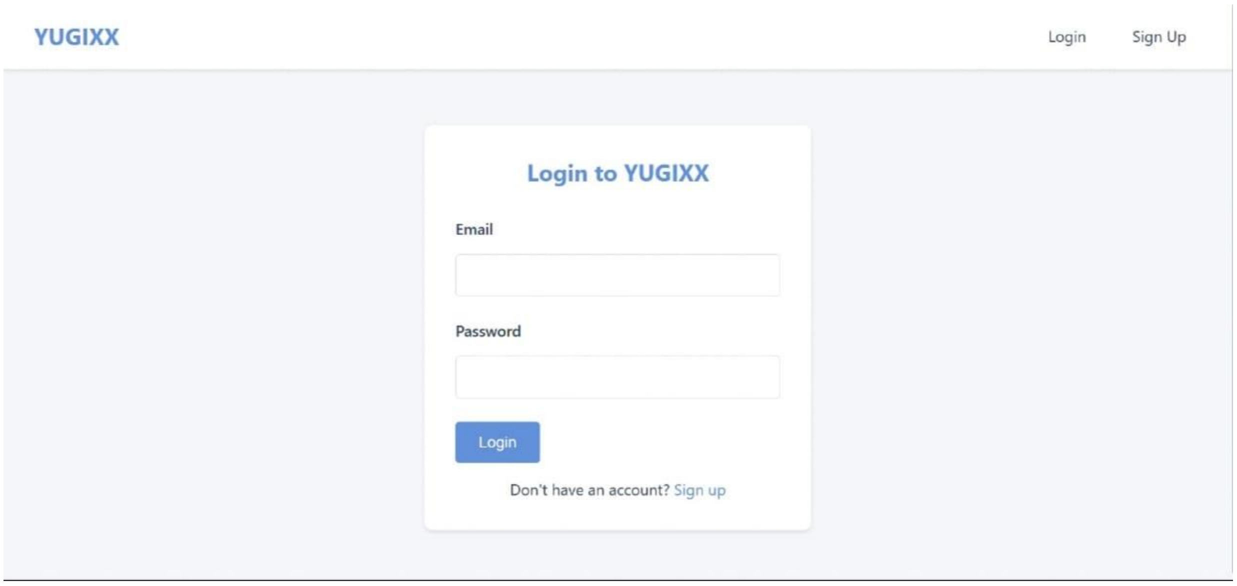


FIGURE 5 Login Page

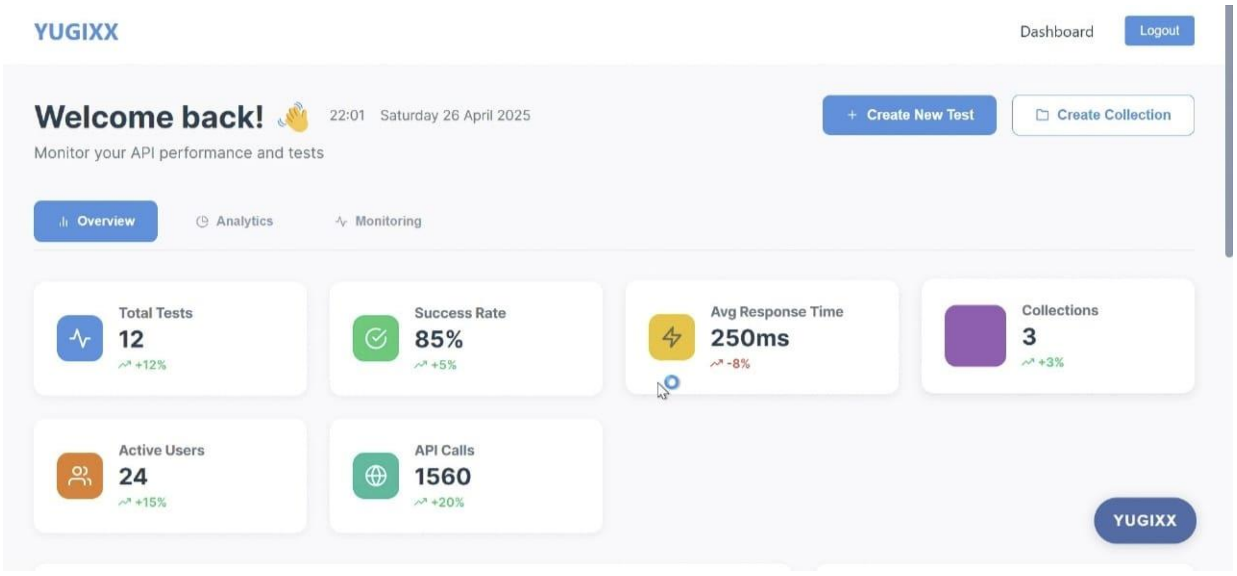


FIGURE 6 Dashboard Page

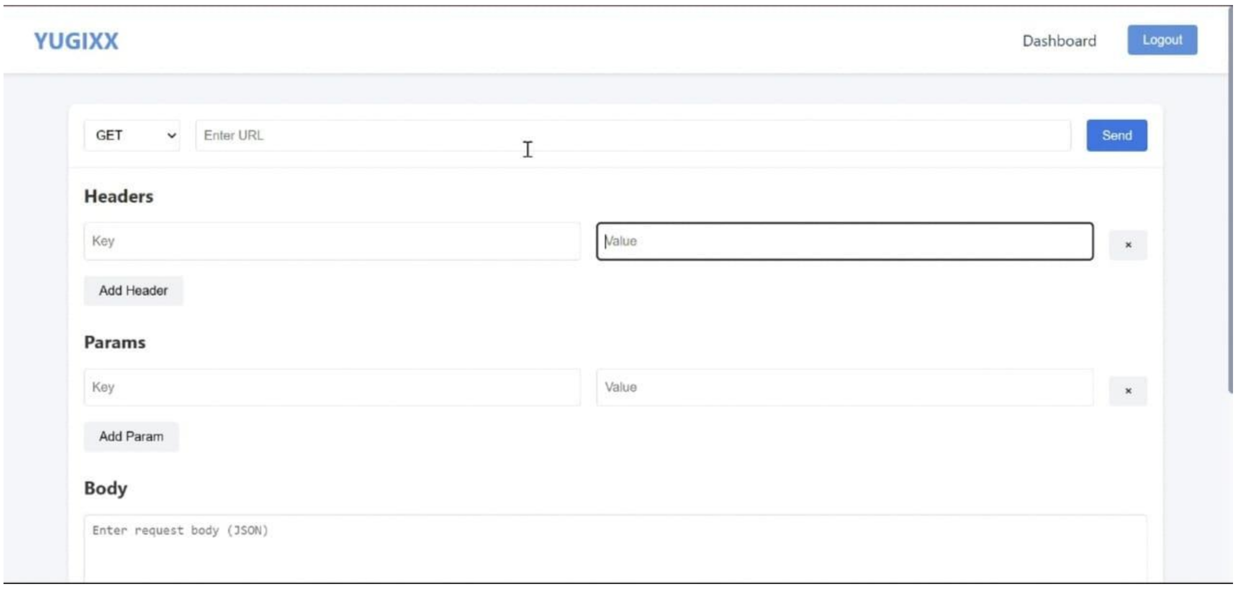
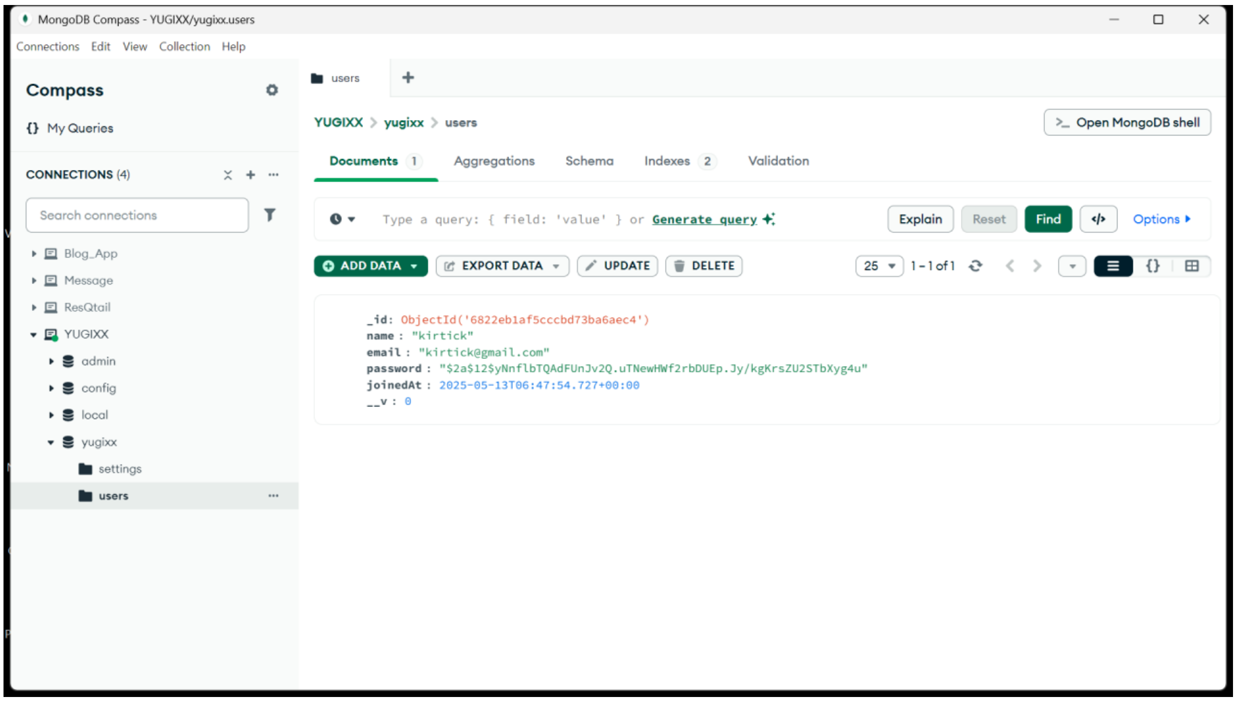


FIGURE 5 API Testing Page



```
_id: ObjectId('6822eb1af5cccbd73ba6aec4')
name : "kirtick"
email : "kirtick@gmail.com"
password : "$2a$12$yNnflbTQAdFUnJv2Q.uTNewHWf2rbDUEp.Jy/kgKrsZU2STbXyg4u"
joinedAt : 2025-05-13T06:47:54.727+00:00
__v : 0
```

FIGURE 6 MongoDB Connection Page

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

The YUGIXX platform, developed using the MERN stack (MongoDB, Express.js, React.js, Node.js), provides a robust and user-friendly solution for testing REST APIs directly from the browser. By combining React.js for an interactive frontend, Node.js and Express.js for efficient backend processing, and MongoDB for secure and scalable data storage, the application ensures high performance and reliability. Core features such as customizable HTTP requests, support for all major request methods, user authentication, and request history management make YUGIXX a practical alternative to desktop-based API testing tools.The Medicine Reminder App developed using the MERN stack (MongoDB, Express.js, React.js, Node.js) offers a robust solution for managing medication schedules. By leveraging MongoDB for data storage, Express.js and Node.js for backend operations, and React.js for a dynamic frontend, the app ensures scalability and responsiveness. Features like customizable reminders, user authentication, and a user-friendly interface enhance medication adherence and user engagement. To further enhance the functionality and user experience, several future upgrades can be integrated. These include environment variable management for streamlined testing across development stages, team collaboration features for shared access and request organization, and automation tools for testing scripts and workflows. Additionally, implementing advanced response visualizations, integrating API documentation tools (like Swagger), and supporting multi-user roles can significantly extend the tool’s professional capabilities. Features such as saving test suites, exporting results, and cloud storage integration will further solidify YUGIXX as a powerful, scalable, and secure API testing platform for developers and QA professionals alike.

REFERENCES

[1] https://www.youtube.com/watch?v=KeC6l1wIYKc
[2] https://www.postman.com/
[3] https://www.sciencedirect.com/science/article/pii/S1877050921006015
[4]
https://www.researchgate.net/publication/359134793_Automated_REST_API_Testing_Framework
[5] https://www.youtube.com/watch?v=ecT42O6I_WI
[6] Corradini, F., Polini, A., & Re, B. (2021). “REST API testing: A systematic literature review.” *Journal of Systems and Software*, Vol. 176, pp. 110925.
[7] Ahmad, A., & Shafqat, N. (2022). “Security and performance testing of RESTful web services.” *International Journal of Computer Applications*, Vol. 184, No. 23, pp. 12–19.
[8] OpenAPI Initiative. (2023). *OpenAPI Specification Documentation*. Available: https://swagger.io/specification/