KAVEESHA HESHAN SENARATHNA

Matara, Sri Lanka | kaveesha.h927@gmail.com | +94 701452284 | GitHub | Portfolio | LinkedIn Profile

Professional Summary

Results-driven Software Engineering undergraduate at SLIIT with comprehensive expertise in full-stack development and emerging technologies. Proficient in building scalable web applications using the MERN stack, cross-platform mobile solutions with Flutter and implementing AI/ML capabilities in real-world projects. Experienced in IoT research and creating intuitive user experiences through Figma-based UI/UX design. Demonstrated ability to deliver user-focused digital solutions by combining strong technical foundations with problem-solving aptitude. Committed to continuous learning and applying research-driven approaches to develop innovative software solutions.

Education

BSc (Hons) in Information Technology (Specialized in Software Engineering)

SRI LANKA INSTITUTE OF INFORMATION TECHNOLOGY (SLIIT) | 2022 – 2026

School Education

RAHULA COLLEGE, MATARA | 2019 – 2021

Technical Skills

- **Backend:** Node.js, Express.js, Spring Boot, Java, PHP, RESTful APIs, Microservices, JWT, OAuth
- **Frontend:** React, Tailwind CSS, Material-UI, Kotlin, jQuery
- **Databases:** MongoDB, MySQL, Firebase
- **DevOps & Cloud:** Docker, CI/CD (GitHub Actions), Vercel
- **Testing:** Jest, Postman, React Testing Library, JUnit

Project Experience

1. Secure RESTful API for Personal Finance Tracker

- Developed a **full-stack income & expense management system** using **Node.js, Express.js, MongoDB**, and **React.js** frontend.
- Built a **responsive React.js UI** for **CRUD operations** on income/expense transactions, including **forms, validation, filtering, and search**.
- Designed and implemented **secure RESTful APIs** for **user authentication**, income tracking, and expense management.

- Integrated **JWT authentication** with **middleware** to enforce **secure access control** and protected routes (frontend + backend).
- Built **Mongoose data models** (**User, Income, Expense**) with **schema validation**, relationships, and optimized queries.
- Implemented **dashboard analytics** in React (summary cards/charts/tables) to visualize totals, categories, and trends.
- Added **PDF report generation** for exporting income/expense summaries and enabled **download/export** from the frontend.
- Applied **MVC architecture** on the backend to ensure clean, scalable, maintainable code and separation of concerns.
- Used **Git/GitHub version control**, followed best practices for **API security, error handling, and input validation**.

2. Healthcare Appointment & video consultation Platform

- Developed a **full-stack healthcare platform** with **Patient/Doctor/Admin portals** using **Node.js, Express.js, MongoDB,** and **React.js** frontend.
- Built a **responsive React.js UI** for **CRUD operations** on appointments, medical records, and consultations, including **forms, validation, filtering, and search**.
- Designed and implemented **secure RESTful APIs** for **user authentication, role-based access control (RBAC),** appointment scheduling, and telemedicine management
- ⍰ Integrated **JWT authentication** with **middleware** to enforce **secure access control** and protected routes (frontend + backend).
- ⍰ Built **Mongoose data models** (**User, Appointment, Doctor, Patient, Prescription, LabResult**) with **schema validation**, relationships, and optimized queries.
- ⍰ Implemented **video consultation features** including **video consultation integration** (Jitsi/Twilio), **waiting room functionality**, and **real-time notifications**.
- ⍰ Added **digital health records management** for storing/retrieving **prescriptions, lab results, medical history**, and enabled **file uploads** for documents.
- ⍰ Developed **admin dashboard analytics** in React (summary cards/charts/tables) to visualize **appointments, revenue, patient trends**, and user management.
- ⍰ Applied **MVC architecture** on the backend to ensure clean, scalable, maintainable code and separation of concerns.
- ⍰ Used **Git/GitHub version control**, followed best practices for **API security, error handling, input validation,** and **centralized Axios configuration**.