The SJSU Wellness Assistant: AI-Powered Telehealth Platform

Anudeep Goud Kotagiri, Kaushikq Ravindran, Komal Venugopal Vattumilli, Parth Joshi
Computer Engineering Department
San José State University (SJSU)
San José, CA, USA

***Abstract*—This report details *The SJSU Wellness Assistant*, an advanced AI-powered telehealth platform tailored for San Jose State University students. The platform enhances healthcare accessibility, efficiency, and user experience through integrated technologies including a secure MERN stack, Firebase for authentication, OpenAI-based Retrieval-Augmented Generation (RAG) for automated medical summarization, multilingual chatbot support, and Zoom for live consultations with transcription capabilities. By prioritizing HIPAA compliance, scalable backend services, and a responsive Web UI, the system addresses key pain points in remote healthcare delivery. The solution encompasses comprehensive requirements, user personas, job shadowing insights, storyboard and wireframes, high-level architecture design, data flow diagrams, and a sequence of workflows. HTML5 enhancements, RESTful interfaces, robust client-side design, rigorous testing, Selenium automation, cross-browser compatibility, JS libraries documentation, design patterns, pagination strategies, localization features, SEO optimizations, and performance profiling form integral parts of this platform. This cohesive approach ensures a seamless and secure experience for patients, doctors, and administrators, ultimately empowering the university community with reliable, intelligent, and accessible telehealth services.**

Acknowledgments

The team acknowledges the valuable insights from SJSU students, healthcare providers, and administrative staff who participated in the VOC studies, persona development, and job shadowing sessions. Their feedback and collaboration were instrumental in shaping a platform that meets real-world needs. Additionally, the guidance from academic mentors and technical support personnel ensured that best practices, compliance standards, and cutting-edge tools were effectively integrated.

Contents

Acknowledgments 1

I Project Report (300) 1
- I-A Project Description (10) . . . 1
- I-B Requirements (10) . . . 2
- I-C Web UI Requirement Principles – VOC, Personas, Job Shadowing (15) . . . 2
- I-D Web UI Design Principles – Storyboard, Wireframes (30) . . . 2
- I-E High Level Architecture Design (10) . . 3
- I-F Data Flow Diagram & Component Level Design (5) . . . 4
- I-G Sequence or Workflow (5) . . . 4
- I-H HTML5 Features Used (20) . . . 4
- I-I Interfaces – RESTFul & Server Side Design (10) . . . 4
- I-J Client Side Design (20) . . . 4
- I-K Testing (UI or Stress test) (25) . . . 4
- I-L Automation (Selenium *) (25) . . . 5
- I-M Cross Browser Compatibility (20) . . . 5
- I-N Java Script Libraries – documentation (10) 5
- I-O Design Patterns Used (10) . . . 5
- I-P Pagination (15) . . . 5
- I-Q Localization (Bonus, Bonus, Bonus) (30) 5
- I-R SEO (10) . . . 5
- I-S Profiling (20) . . . 5

I. Project Report (300)

This section provides a comprehensive aggregation of all project components. The following subsections align with the structured checklist provided.

A. Project Description (10)

Project Description: The SJSU Wellness Assistant is a cutting-edge AI-powered platform designed to enhance healthcare accessibility and efficiency for students at San Jose State University. By integrating advanced technologies such as a MERN stack (MongoDB, Express, React, Node.js) along with Firebase for authentication, the system ensures robust development and secure access management. OpenAI's Retrieval-Augmented Generation (RAG) capabilities enrich the chatbot's intelligence, providing automated medical summarization and enabling multilingual support for a diverse user population.

At its core, the platform delivers a user-centric design compliant with HIPAA standards, ensuring data confidentiality and integrity. The integration with Zoom for live consultations allows patients and doctors to communicate in real-time, while transcription features ensure accurate record-keeping for follow-up care. The platform's intuitive patient portal enables seamless login/registration, a symptom checker powered by AI, a chat interface, and follow-up scheduling. On the doctor's side, a dedicated portal supports patient management, consultation dashboards, and prescription generation. This robust

ecosystem streamlines appointments, offers sophisticated AI-driven insights post-consultation, and presents comprehensive consultation histories for both patients and doctors.

The platform thus stands as a holistic solution: from quick symptom assessments to secure consultations, automated summarizations, prescription handling, localized language support, and integrated testing and optimization. Overall, the SJSU Wellness Assistant redefines remote healthcare interactions for the SJSU community, ensuring that students receive timely, informative, and personalized medical support.

B. Requirements (10)

Functional Requirements: - **Patient Registration:** Secure registration with Firebase Authentication, encryption of sensitive info in MongoDB. - **Login and Authentication:** User login with session management via Firebase Auth. Role-based access for patients/doctors. - **Symptom Checker:** AI-driven symptom input, suggests conditions or referrals. - **Consultation Management:** Appointment booking, doctors can manage schedules. - **Live Consultations:** Zoom integration for real-time video, transcription in MongoDB. - **Automated Medical Summarization:** NER and RAG-based summarization post-consultation. - **Multilingual Chatbot:** Multiple language support, contextually accurate responses. - **Prescription and Follow-Up Management:** Doctors can generate digital prescriptions, schedule follow-ups, and notify patients. - **Data History and Retrieval:** Patients can access past consultation history and summaries. - **Admin Dashboard:** Monitor performance and manage user data securely.

Non-Functional Requirements: - **Performance:** 95% responses ¡ 1s, low latency for real-time features. - **Scalability:** Support 10,000 concurrent users, dynamic scaling. - **Security:** HTTPS, end-to-end encryption, HIPAA compliance. - **Reliability:** 99.9% uptime, auto-recovery mechanisms. - **Localization:** UI and chatbot in at least 5 languages. - **Accessibility:** WCAG 2.1 compliance. - **Testing and Validation:** Selenium-based E2E tests, API performance tests (JMeter).

System Constraints: - Firebase for authentication. - MERN stack for frontend/backend. - MongoDB for structured data. - OpenAI for language processing and summarization. - Zoom API for live consultations and transcription.

C. Web UI Requirement Principles – VOC, Personas, Job Shadowing (15)

Voice of the Customer (VOC): Feedback from surveys, interviews, and contextual inquiries revealed the need for remote healthcare access with real-time consultations, secure data management, intuitive dashboards, and multilingual support. These insights guided feature prioritization for patients, doctors, and administrators, ensuring user-friendly appointment handling, easy data retrieval, and accessible healthcare information.

Personas: 1. *Student User:* - Goals: Secure registration, easy appointment booking, access to medical summaries. - Challenges: Limited time, complex medical terms, quick retrieval of past consultations. - Usage: Frequently checks upcoming appointments, interacts with AI chatbot, reviews summaries.

2. *Healthcare Provider:* - Goals: Manage patient volume, access concise summaries, generate prescriptions, follow-ups. - Challenges: Efficient data retrieval, effective communication during live consultations. - Usage: Reviews patient histories, conducts video sessions, leverages AI-driven summaries.

3. *Administrator:* - Goals: Maintain system stability, monitor data security, ensure uptime and performance. - Challenges: Quick issue resolution, secure workflows, multilingual consistency. - Usage: Monitors system health, enforces compliance, ensures platform localization.

Job Shadowing: Observing real interactions at SJSU Wellness Center revealed usability hurdles and needs: - Simplified navigation with intuitive dashboards. - Streamlined appointment scheduling, direct access to meeting links. - Enhanced chatbot for multilingual support and context-sensitive assistance. - Integrated prescription workflows to easily create, retrieve, and remind patients.

These findings influenced the platform's design, emphasizing smooth user flows, quick data access, responsive UI elements, and culturally inclusive localization.

D. Web UI Design Principles – Storyboard, Wireframes (30)

Wireframes: Wireframes provided the skeletal layout of key pages: - Patient Portal: Registration forms, appointment calendars, chat interfaces, upcoming consultation lists. - Doctor Dashboard: Patient lists, summarized medical data, quick prescription generation forms. - Admin Panel: Performance metrics, user management tools, language configuration screens.

Storyboard: The storyboard depicted a student logging in, using a symptom checker, booking an appointment, interacting with the chatbot in their preferred language, joining a Zoom consultation via a single click, and finally reviewing the AI-generated medical summary and prescription details afterward.

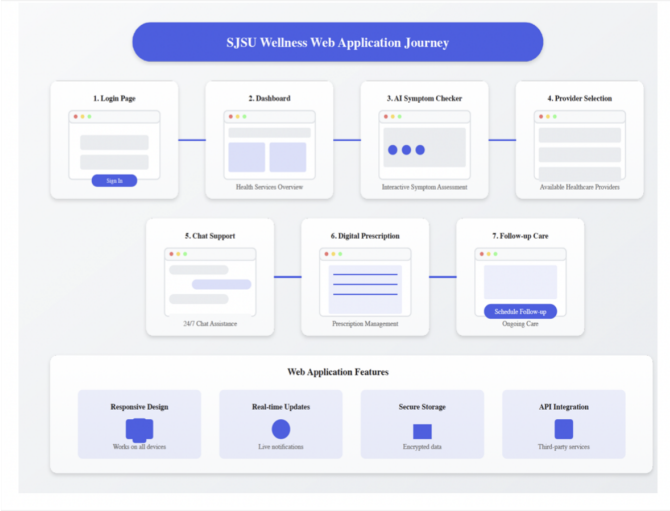

Fig. 1: Storyboard

These initial design artifacts ensured that the UI supports the required functionalities, guiding developers and stakeholders

with a clear visual and functional blueprint. The inclusion of collapsible sections, responsive layouts, dynamic data rendering, and culturally adaptable design elements ensured a smooth user experience across various devices and languages.

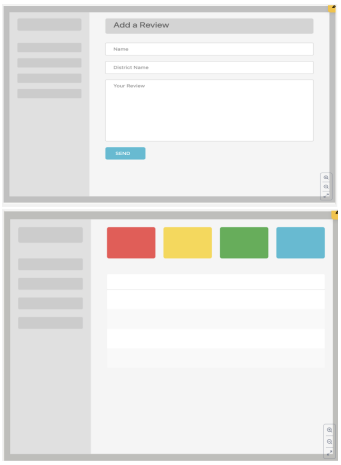

Fig. 2: Review & Dashboard Mockup

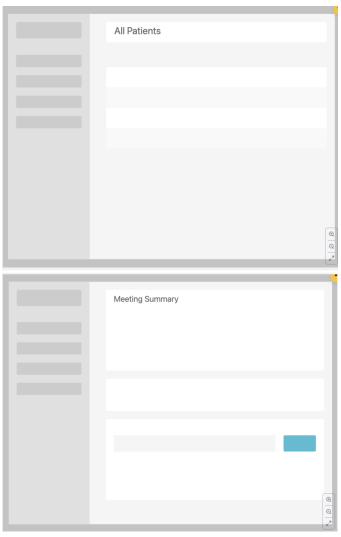

Fig. 3: Patients & Meeting Summary Mockup

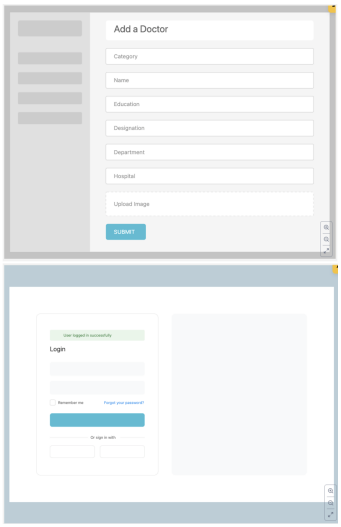

Fig. 4: Login & Add Doctor Page Mockup

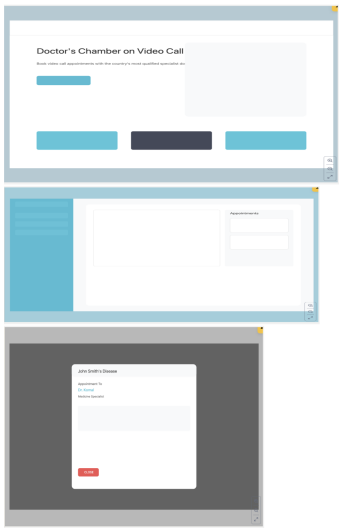

Fig. 5: Appointment, Home & Disease Page Mockups

E. High Level Architecture Design (10)

Architecture Overview: The platform's architecture comprises: - **Users Layer:** Patients, Healthcare Providers, Administrators. - **Frontend (React):** Responsive UI, authentication forms, data visualization, Zoom integration. - **API Gateway (Firebase):** Authentication, real-time sync, routing requests. - **Backend Services (MERN + AI):** Core functions (registration, upload, summarization), Zoom services (transcription), RAG system (OpenAI integration). - **Data Layer (MongoDB):** Central data store for user profiles, medical data, consultation records, transcripts.

Key capabilities: Security (HIPAA), real-time video consultations (Zoom), NER + RAG for AI-driven insights, scalable microservices architecture ensuring performance and reliability.

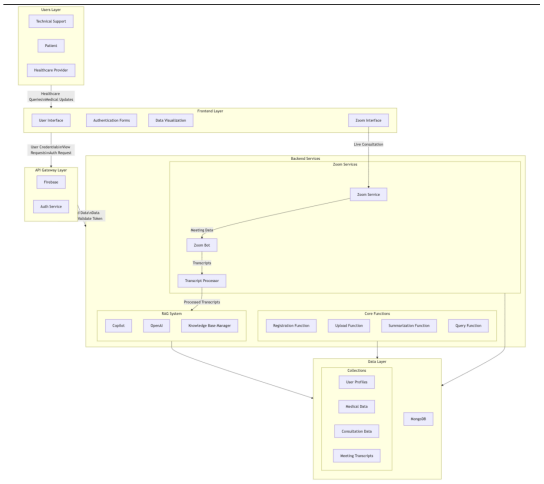

Fig. 6: High-Level Architecture Diagram (Placeholder)

F. Data Flow Diagram & Component Level Design (5)

Data Flow Diagram (DFD): The DFD illustrates: - Users requesting authentication via Firebase. - Patients booking appointments via the frontend, invoking backend APIs. - Zoom services feeding transcripts to the AI pipeline. - The RAG system retrieving data from MongoDB, returning summarized insights. - Final data rendered in the UI for patients and doctors.

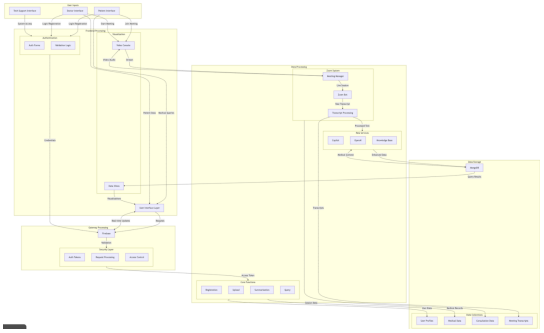

Fig. 7: Data Flow Diagram (Placeholder)

Component-level design refines this further into distinct modules: authentication handlers, NER processors, RAG retrievers, and UI components interacting through well-defined APIs.

G. Sequence or Workflow (5)

A typical workflow: 1. Patient logs in (Firebase Auth). 2. Schedules an appointment (Frontend → Backend → MongoDB). 3. Joins consultation (Zoom → Real-time transcription). 4. Post-consultation transcript processed by NER and RAG (Backend + OpenAI). 5. Summaries stored in MongoDB, rendered in Patient/Doctor portals. 6. Patient reviews summaries, doctor updates prescriptions, system notifies follow-ups.

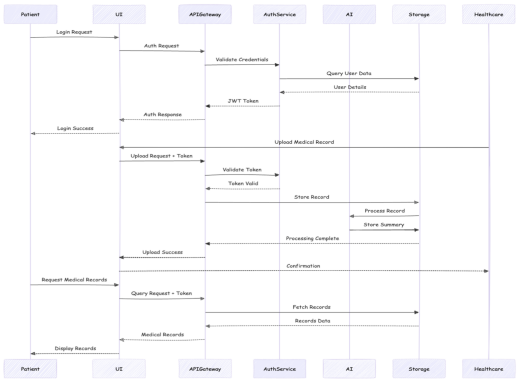

Fig. 8: Sequence Diagram (Placeholder)

H. HTML5 Features Used (20)

Semantic Structure: Use of modern HTML5 semantic elements (`<div>`, `<span>`, `<h4>`, `<h6>`, `<p>`) and proper class naming (e.g., `doctor-description`, `doctor-name`) improves readability and accessibility.

Rich Media Support: Images with `alt` attributes ensure accessibility. Responsive images and base64 image support for optimized loading.

Interactive Elements: Collapsible content sections, dynamic state-based rendering, and event handling for smooth user interactions.

I. Interfaces – RESTFul & Server Side Design (10)

RESTful APIs: Data-driven component rendering, secure endpoints for appointments, consultations, prescriptions. The server-side Node.js/Express backend handles CRUD operations, integrates with Firebase for auth, and interfaces with Zoom and OpenAI APIs.

Server-Side Features: - Authentication and token validation. - Public asset serving for static files. - Role-based access control. - Data handling with secure endpoints for patient histories, transcripts, and summaries.

J. Client Side Design (20)

Component Architecture: React-based modular components, each responsible for a specific UI element (appointment forms, consultation dashboards, profile views).

State Management: React hooks (`useState`, `useEffect`) for local state management. Dynamic rendering for real-time updates.

UI/UX Features: Responsive layouts (CSS, possibly Tailwind), collapsible sections, conditional rendering, image optimization, localization for multilingual UI.

K. Testing (UI or Stress test) (25)

Testing Frameworks: - Jest for unit tests of React components. - Selenium and Cypress for E2E UI testing. - JMeter for load and stress testing.

Test Scenarios: - Validation of form inputs, appointment scheduling flow. - Performance metrics under 10,000 concurrent users. - Response times, error rates, throughput documented and optimized.

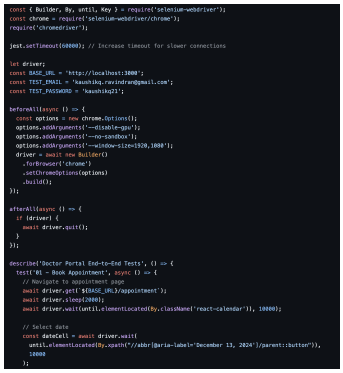

Fig. 9: Selenium UI Tests

Fig. 10: Performance Test

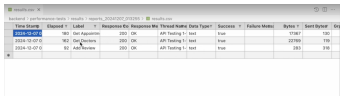

Fig. 11: Test Results

*L. Automation (Selenium *) (25)*

End-to-End Automation: Selenium WebDriver tests cover: - Appointment booking: calendar interaction, form submission. - Login flow: session management checks. - Data consistency: verifying appointment data across pages. - Payment processes (if integrated).

Test Infrastructure: Screenshot capture on failures, custom waits, error handling, automated test execution in CI/CD pipeline.

M. Cross Browser Compatibility (20)

Browser Support: Optimized for Chrome; tests extended to other browsers. Custom browser profiles, viewport management, cross-browser element interaction strategies ensure consistent rendering across devices.

Testing Methodology: Visual regression tests (screenshot comparisons), responsive checks on various screen sizes, documenting browser-specific considerations.

N. Java Script Libraries – documentation (10)

Core Stack: - React.js: Component-based architecture. - Bootstrap 4/CSS frameworks: Responsive styling. - Firebase: Authentication and real-time data. - OpenAI: Language processing for summaries. - Zoom SDK: Live consultation integration. - Stripe (if payment involved): Secure payment handling. - React Calendar, Socket.IO: Scheduling and real-time updates.

Each library integration is documented, detailing installation, usage patterns, configuration, and custom wrapper components.

O. Design Patterns Used (10)

Architectural Patterns: - Component-based architecture in React. - Page Object Model (POM) in Selenium tests. - Observer pattern for state management. - Factory pattern for flexible WebDriver initialization. - Singleton pattern for shared utilities.

P. Pagination (15)

Implementation Details: - Appointment listings use pagination for efficient navigation. - Cursor-based pagination for MongoDB queries. - Client-side caching improves performance. - Responsive pagination controls ensure accessibility. - Adjustable page sizes adapt to user preferences.

Q. Localization (Bonus, Bonus, Bonus) (30)

Language Support: - Multilingual chatbot (English, Telugu, Hindi, plus 3 more languages). - Language detection and dynamic UI adaptation. - Translated UI elements, culturally appropriate date/time formats. - RTL support for relevant languages. - Ensures inclusive access for the diverse SJSU community.

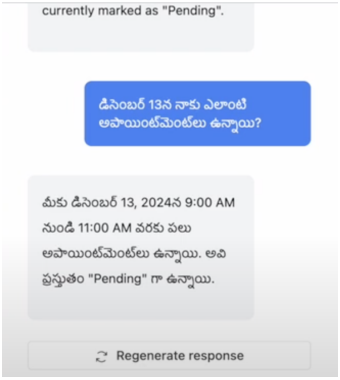

Fig. 12: Localization

R. SEO (10)

Optimization Strategies: - Semantic HTML structure for search engine visibility. - Dynamic meta tags, descriptive alt attributes. - Mobile-first responsive design. - Structured data markup (JSON-LD) to improve discoverability.

S. Profiling (20)

Performance Monitoring: - Database query optimization and indexing. - Memory usage tracking in Node.js backend. - Client-side performance metrics via Lighthouse audits. - Automated performance reporting and continuous monitoring to ensure stable, responsive user experience under load.