**1. Introduction**

**Purpose:**

* The purpose of this document is to provide a detailed architectural overview of the telehealth application, aimed at delivering instant, secure, and user-friendly healthcare consultations remotely. This document will serve as a guide for developers, project managers, and stakeholders to understand the design and functionality of the system.

**Scope:**

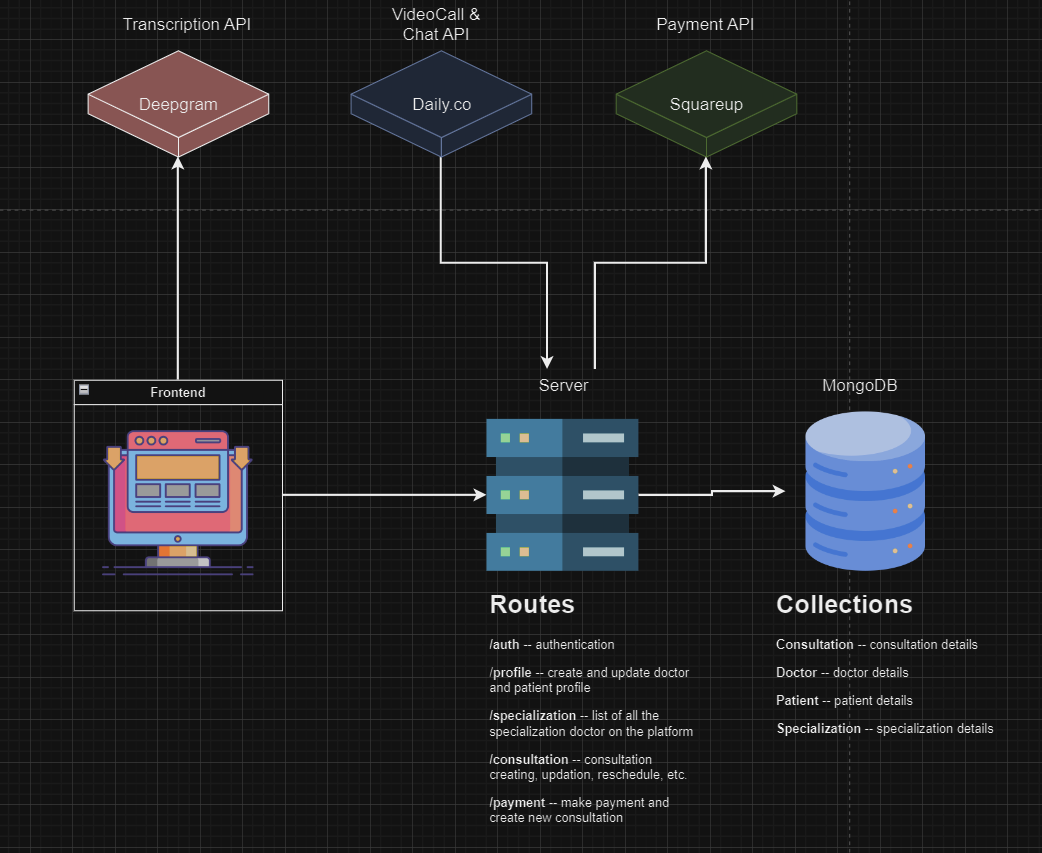
* **Overview of Features:** The document covers the website design, including how it handles user interactions, data processing, and integration with external services.
* **Objectives:** To ensure that the design meets all project requirements, including secure data storage, real-time video and chat functionality, payment processing, and transcription services.

**2. System Overview**

**Architecture Diagram:**

* **High-Level Diagram:**

url : <https://drive.google.com/file/d/1FuiJiXEiKH456a-c1KskdU8w0kxm4nFG/view?usp=sharing>

****

* **Components:**
  + **Client-Side (Front-End):** React application running in the user’s browser.
  + **Server-Side (Back-End):** Node.js server with Express.js handling API requests.
  + **Database:** MongoDB for storing user and consultation data.
  + **External Services:** Payment gateway (Squareup), transcription service (Deepgram), and Videocall service (Daily.co).

**Technology Stack:**

* **Front-End:**
  + **React:** A JavaScript library for building user interfaces. Used for creating the web app’s dynamic components and handling state management.
* **Back-End:**
  + **Node.js:** A runtime environment that allows JavaScript to run on the server-side.
  + **Express.js:** A web application framework for Node.js, used for building RESTful APIs and handling HTTP requests.
* **Database:**
  + **MongoDB:** A NoSQL database that stores data in a flexible, JSON-like format. Ideal for handling varied data types and structures.
* **Additional Services:**
  + **Payment Gateway:** Integrates with Squareup for handling financial transactions.
  + **Transcription Service:** Converts spoken language into text, useful for real-time transcription during consultations.
  + **Videocall and chat:** Enables real-time messaging during consultation between users and healthcare providers.

**3. Front-End Architecture**

**UI/UX Design:**

* **Components:**
  + **Appointment Scheduler:** A React component that allows users to select a time slot, specialty, and provider for their consultation.
  + **Consultation Interface:** A video call interface using Daily.co API video service, integrated with components to display consultation details.
  + **Payment Form:** A secure form component for entering payment details, integrated with a payment gateway.
  + **Dashboard:** a dashboard for doctor and patient to see their upcoming and past consultations.

**Features:**

* **Instant Health Consultation:** Allows users to schedule and join video consultations with healthcare providers. The React application handles user inputs, API calls to fetch available slots, and displays the video interface.
* **Patient Information Capture:** Forms collect user data such as name, contact details, and medical history. The information is sent to the back-end for processing and storage.
* **Chat Integration:** Real-time chat functionality is implemented using daily.co API. Messages are sent and received in real-time during consultations.
* **Transcription Service Integration:** Use APIs from transcription services to process and display text in real-time during video consultations. The transcribed text can be shown alongside the video feed.

**Security Measures:**

* **Data Encryption:** Ensure that all data sent between the client and server is encrypted using HTTPS. Sensitive data, such as payment information, should be handled securely.
* **Authentication:** Use secure methods for user authentication, such as JWT (JSON Web Tokens), to protect user sessions and data.

**4. Back-End Architecture**

**API Design:**

* **Endpoints:**

**/auth**-- authentication

/**profile** -- create and update doctor and patient profile

**/specialization** -- list of all the specialization doctor on the platform

**/consultation**-- consultation creating, updating, reschedule, etc.

**/payment**-- make payment and create new consultation

* **Data Handling:**
  + **Request Validation:** Ensure all incoming data is validated and sanitized to prevent security issues and ensure data integrity.
  + **Response Formatting:** Return data in a consistent format, such as JSON, for easy integration with the front-end.

**Business Logic:**

* **Consultation Scheduling:** Implement logic for managing consultation, provider scheduling, and patient bookings. Ensure that providers are notified of new bookings and that patients receive confirmation.
* **Payment Processing:** Integrate with a payment gateway to handle transactions securely. Implement logic for verifying payments, handling payment errors, and updating consultation statuses based on payment success.
* **Transcription Service:** Overcome challenges of difference in dialect/ accent in remote consultation by adding transcription feature.
* **Remote Consultation and messaging:** Integrated video calling with chat functionality provide information from within the application during Telehealth session

**Authentication & Authorization:**

* **User Authentication:** Implement JWT-based authentication to securely manage user sessions. Ensure that tokens are stored securely and verified on each request.
* **Authorization:** Define roles and permissions to control access to different parts of the application. For example, patients should have access to their own consultations and payment information, while healthcare providers should have access to their schedules and patient details.

**Security Measures:**

* **Data Encryption:** Encrypt sensitive data stored in the database and transmitted over the network.
* **Input Validation:** Validate and sanitize all user inputs to prevent injection attacks and other security vulnerabilities.

**5. Database Design**

**Schema Design:**

* **Consultation --** The Consultation schema stores details about each patient-doctor consultation, including patient and doctor references, consultation date, symptoms, diagnosis, prescription, consultation status, meeting URL, and payment transaction details. It also tracks timestamps for when the consultation was created and updated.
* **Doctor --** The Doctor schema stores information about healthcare providers, including their name, email, hashed password for authentication, and their medical specialization. It also tracks the date when the doctor's record was created.
* **Patient --** The Patient schema stores personal information about patients, including their name, email, password (hashed for security), gender, and encrypted health information (like past medical conditions, current medications, allergies, and family medical history). The health data is encrypted for privacy, and it also tracks when the patient record was created.
* **Specialization --** The Specialization schema stores information about the various medical specialties available in the telehealth application. It includes the type of specialization (e.g., cardiology, dermatology), consultation fees associated with that specialty, and tracks when the specialization record was created.

**Data Access:**

**Mongoose Models:** Define Mongoose schemas and models to interact with MongoDB. These models handle data validation, querying, and manipulation.

* **CRUD Operations:** Implement create, read, update, and delete operations for each collection, ensuring that data is managed effectively and securely.

**Figma Link** : <https://www.figma.com/design/isYNkywg2WFa4hrnwjMprc/Codia-AI-Design%3A-Screenshot-to-Editable-Figma-Design-(Community)?node-id=0-1&t=Sr6FQ2gXBopYIlpO-1>