IOTRIC

DEVELOPMENT PROPOSAL

PREPARED BY

Ved Prakash

CEO | IOTRIC

Thank you for meeting with us to discuss your company's software project requirements. Attached is our detailed development plan proposal for your consideration and approval. We know that every client is unique, so we strive to deliver an individual, innovative, and affordable proposal every time, following through with an outstanding delivery that is both on time and within budget.

About this Statement of Work

This is the Statement of Work ("SOW") to provide details necessary to ensure the successful delivery of a project and onboarding as a new or existing client. Please read the terms of this Statement of Work carefully, as you will select products and services, and agree to certain provisions that govern our relationship. When we accept it, this Statement of Work and all the accompanying or supplemental documents form the entire Agreement between us for this account.

The purpose is to ensure

The development plan is based on the specifications provided by the client via documentation. This document is not for providing a competitive analysis of the proposed solution market feasibility study of the proposed solution.

About App (m AI doc):

Internationally patented technology for the ultimate health app, featuring a blockchain and AI platform for Electronic Health Records (EHR) of connected users. It employs a multilayered structure of Deep Learning Algorithms for continuous and preventive monitoring of personal eHealth data.

The **m AI doc** app will analyze personal health data, match it with global medical information, and provide life-saving advice on the user's smartphone screen. It can also send real-time updates to personal medical actors, family members, and friends.

The ultimate goal is to achieve 99.9% correct diagnosis and personalized medical treatment almost instantly by combining personal health data with digital results of scans, genomic analysis, and laboratory data.

1. **Internationally patented technology:** The app boasts proprietary technology that has been patented internationally, indicating its uniqueness and potential superiority in the market.

2. **Blockchain platform for EHR:** The app utilizes blockchain technology as the foundation for storing Electronic Health Records (EHR) of connected users. Blockchain offers advantages such as enhanced security, transparency, and immutability, ensuring the integrity and privacy of health data.

3. **Deep Learning Algorithms (DLA) structure:** A multilayered structure of Deep Learning Algorithms will be built on top of the EHR platform to analyze incoming personal health data in real-time.

4. **Challenges in health monitoring devices:** Despite the abundance of health monitoring devices on the market, few are profitable, except for the Apple Watch. The goal is to create a preventive health monitoring solution based on universally available Electronic Health Records.

Key Points:

- Utilization of personal smartphone as a life-saving companion.
- Permanent monitoring of personal health situation.
- Alarming, diagnosis, and potential medical solutions integrated into the app.
- The multilayered structure of Deep Learning Algorithms for health/disease diagnosis.
- The target accuracy is of 99% for the diagnosis.
- Permanent monitoring of Electronic Health Records stored in private blockchain mega data platform.

System Components:

1. **User Interface (m AI doc App):**
    - Provides a user-friendly interface for users to interact with the application.
    - Allows users to input health data, view personalized insights, and receive notifications.
    - Supports authentication and user profile management functionalities.
2. **API Gateway:**
    - Acts as a single entry point for all client requests.

- Handles authentication, request routing, and load balancing.
- Provides security features such as rate limiting, encryption, and threat protection.

3. **Application Layer:**
    - Implements business logic and coordinates interactions between different components.
    - Validates user inputs, performs authorization checks, and enforces business rules.
    - Orchestrates the execution of microservices and backend processes.
4. **Microservices and Backend Components:**
    - **Authentication & Authorization Service:**
        - Manages user authentication, generates access tokens, and enforces access control policies.
        - Integrates with identity providers and user directories for user authentication.
    - **Data Layer:**
        - Stores and manages various types of data, including user profiles, electronic health records (EHR), health measurements, and medical records.
        - Utilizes relational and NoSQL databases for structured and unstructured data storage.
    - **Deep Learning Algorithms (DLA) Engine:**
        - Implements a multilayered structure of deep learning algorithms for real-time analysis of health data.
        - Processes incoming data from the Data Layer and generates personalized health insights and recommendations.
    - **Integration Services:**
        - Facilitates integration with external data sources such as medical devices, diagnostic tools, and research databases.
        - Provides connectors, APIs, and data pipelines for seamless data exchange and interoperability.
    - **Notification Service:**
        - Sends real-time notifications to users, personal medical actors, and authorized contacts.
        - Supports various communication channels such as push notifications, email, and SMS.
5. **Scalability and Performance:**
    - Utilizes containerization (e.g., Docker, Kubernetes) and microservices architecture for scalability and flexibility.

- Implements caching mechanisms, load balancing, and horizontal scaling to handle increased user traffic and data volumes.
- Monitors system performance, resource utilization, and application health using monitoring tools and logging frameworks.

6. **Security and Compliance:**
    - Enforces security measures such as encryption, data masking, and access controls to protect sensitive health data.
    - Implements compliance with healthcare regulations such as HIPAA, GDPR, and OWASP guidelines.
    - Conducts regular security audits, vulnerability assessments, and penetration testing to identify and mitigate security risks.
7. **Monitoring and Logging:**
    - Implements logging frameworks (e.g., ELK stack) to capture and analyze system logs, errors, and events.
    - Monitors system metrics, application performance, and user interactions using monitoring tools (e.g., Prometheus, Grafana).
    - Alerts administrators and operators about critical issues, anomalies, and security threats in real time.
8. **External Interfaces and Integrations:**
    - Integrates with external systems and services such as electronic medical records (EMR) systems, telemedicine platforms, and wearable health devices.
    - Implements APIs, webhooks, and data formats (e.g., HL7, FHIR) for seamless data exchange and interoperability.
    - Ensures compatibility and compliance with industry standards and interoperability frameworks.

System API Design:
Here's a high-level overview of the API system design.

1. **RESTful API Design:**
    - Define a set of RESTful endpoints for accessing different functionalities of the application, such as user authentication, health data retrieval, analysis, and notifications.
    - Use HTTP methods (GET, POST, PUT, DELETE) to perform operations on resources and ensure consistency and predictability in API interactions.

2. **Authentication and Authorization:**
    - Implement secure authentication mechanisms such as OAuth 2.0 or JSON Web Tokens (JWT) to verify the identity of users and protect sensitive data.
    - Enforce role-based access control (RBAC) to restrict access to certain API endpoints based on user roles and permissions.

3. **Data Formats:**
    - Data formats for request and response payloads as JSON or XML, to facilitate interoperability and ease of integration with client applications.
    - Use standard data structures and conventions to ensure consistency and compatibility across different API endpoints.
4. **Error Handling:**
    - Define standardized error responses and status codes to communicate error conditions and provide meaningful error messages to API consumers.
    - Include error-handling logic in API endpoints to gracefully handle exceptions, validate input data, and prevent potential security vulnerabilities.
5. **Versioning:**
    - Plan for versioning of API endpoints to accommodate future changes and updates to the application without breaking existing client integrations.
    - Use semantic versioning (e.g., v1, v2) to indicate backward-compatible and backward-incompatible changes to the API.
6. **Rate Limiting and Throttling:**
    - Implement rate limiting and throttling mechanisms to prevent abuse and ensure fair usage of API resources.
    - Define usage quotas, request limits, and rate limits based on user roles, subscription plans, or API usage tiers.
7. **Documentation**:
    - Create comprehensive documentation for the API, including detailed descriptions of endpoints, request parameters, response payloads, authentication methods, and error handling.
    - Use of tools like OpenAPI (formerly Swagger) or API Blueprint to generate interactive API documentation and client SDKs for different programming languages.
8. **Testing and Monitoring:**
    - Develop automated tests for API endpoints to verify functionality, performance, and reliability under different scenarios.

- Implement monitoring and logging mechanisms to track API usage, detect anomalies, and troubleshoot issues in real-time.

9. **Scalability and Performance:**
    - Design the API system to be horizontally scalable to accommodate growing user traffic and increasing data volumes.
    - Use caching, load balancing, and microservices architecture to distribute workload efficiently and improve overall system performance.
10. **Security and Compliance:**
    - Ensure compliance with security standards and regulations such as HIPAA, GDPR, and OWASP (Open Web Application Security Project) guidelines.
    - Implement encryption, data masking, and other security measures to protect sensitive health information and prevent unauthorized access or data breaches.

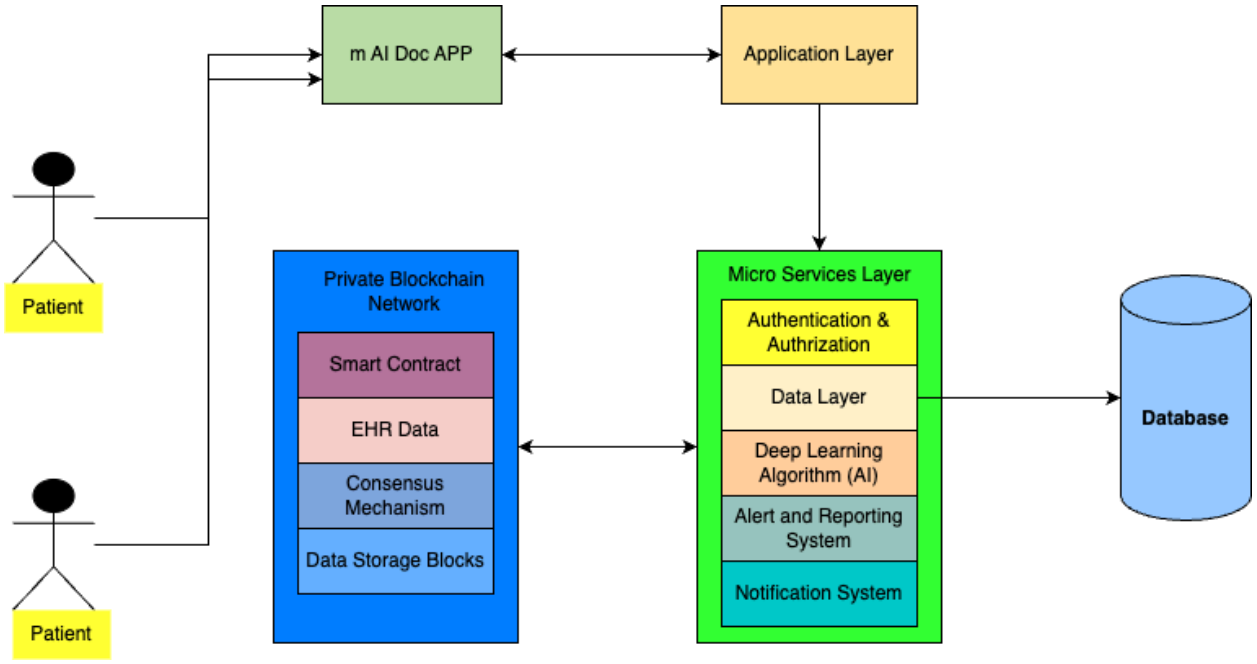
Private Blockchain For EHR

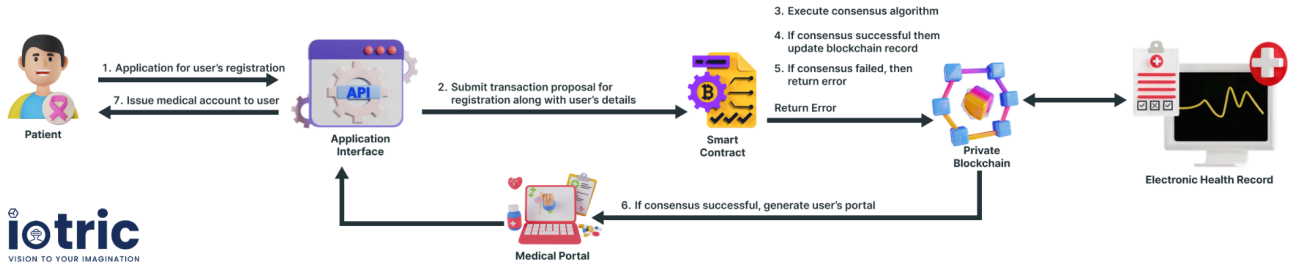
The purpose of private blockchain for EHR is to get several advantages such as enhanced security, improved data integrity, and streamlined access control.

Implementation:

- **Private Blockchain:** Deploy a permissioned blockchain network using Hyperledger Fabric (Or any other Private Chain), where member organizations serve as nodes.
- **Smart Contracts:** Develop smart contracts to govern the creation, access, and update of patient records, ensuring data integrity and privacy.
- **APIs:** Implement APIs for creating, retrieving, updating, and searching patient records, facilitating interoperability between member EHR systems.
- **Security**: Implement encryption, authentication, and access control mechanisms to protect patient data and ensure compliance with regulatory requirements.

Top-Level System Design





Technology Stack:

Mobile App: React-Native or Native Android (Kotlin) and IOS (Swift)
Web Dashboard: React.js
Backend : Node.js, Python
Private Blockchain: Hyper Ledger?