PROGRESS REPORT
AI-Powered Personal Health Record System - Medivise

TEAM MEMBER
Le Mai Thanh Son - V202301128
Nguyen Tien Nhan - V202301006
Nguyen Xuan Truong – V202300998

1 Introduction

1.1 Problem Statement

Traditional personal health record (PHR) systems mainly serve as passive data repositories, requiring patients and clinicians to manually review scattered information. This limits their practical value in early risk detection, personalized health management, and patient engagement[3]. At the same time, concerns around data security, accessibility, and trust hinder widespread adoption. There is a growing need for an intelligent, secure, and user-friendly solution that transforms raw medical records into actionable health insights, while safeguarding patient privacy.

1.2 Project Scope

The Medivise project aims to build an AI-powered Personal Health Record platform that empowers patients and clinicians with proactive, personalized healthcare support. The system is designed around four key modules:

- **Data Ingestion & Integration:** Supports structured data (lab results, prescriptions, vitals), unstructured clinical narratives, and patient-generated records through standardized uploads with OCR and metadata extraction.
- **Secure Storage & Access Control:** Implements encrypted databases, role-based access control, and audit logging to ensure privacy, trust, and compliance with healthcare security standards.
- **AI & Analytics Engine:** Uses NLP and predictive models to extract clinical insights, detect anomalies, forecast risks, and deliver guideline-aware recommendations tailored to individual patient profiles[1].
- **User Interaction Layer:** Provides intuitive web dashboards for patients and clinicians to upload, visualize, and query health records, receive AI-powered alerts, and securely communicate.

Given the 6-week development timeframe and the size of the team, the project scope has been carefully defined to ensure feasibility. Each module is developed and tested independently before integration, promoting modularity and minimizing development risks.

2 Update on Work Progress

2.1 Previously Completed Tasks

From the beginning to the last progress update (Project Description), the team has successfully completed several foundational tasks critical to the project's development:

- **Project Planning & Setup:** We began by setting up our project foundation: creating the GitHub repository, choosing our tech stack (React with Tailwind for the frontend, Flask for the AI microservice, PostgreSQL for storage, and a secure backend API layer), and dividing responsibilities among team members.
- **Requirement Analysis & Architecture Draft:** We analyzed requirements and drafted the system architecture, focusing on four main modules: data ingestion, secure storage, AI analytics, and user dashboards.
- **Database Design:** We designed and finalized the PostgreSQL schema, making sure to normalize tables for users, roles, records, prescriptions, and audit logs. We also added indexes on common query fields like IDs and timestamps.

- **Backend Foundation:** On the backend, we implemented core authentication using JWT and role-based access control (RBAC). We also built HTTPS endpoints and added support for secure search, filtering, and doctor–patient assignment mechanisms.
- **Frontend Initialization:** On the frontend, we created the initial structure of the dashboards, including login pages and placeholders for both patient and doctor views.

2.2 Recently Completed Tasks

- **Backend Development:** On the backend, we finished building APIs for patient and doctor record management. This included secure upload and retrieval of records, prescription handling, and filtering. We integrated encryption for sensitive data, TLS for secure transmission, and implemented audit logs for every access and modification. We also wrote unit tests and validated our APIs with Postman.
- **Database Integration:** We deployed our PostgreSQL schema in a live environment, tested it with sample data, and confirmed that data integrity and multi-user role access worked as expected. Query performance met our target of under 2 seconds for retrieval.
- **Frontend Progress:** On the frontend, we made significant progress on both patient and doctor dashboards. Users can now upload records, search and filter them, and send secure messages. We connected the frontend to backend APIs using Axios, enabling smooth data flow. All features work reliably on desktop browsers, and we plan to optimize the interface for mobile soon.
- **AI Microservice Development:** For the AI module, we developed a standalone Flask service that can analyze structured health data. It detects anomalies, sends medication reminders, and generates basic lifestyle recommendations. We tested it with sample datasets, and the early results are promising. The service is stable on its own, but integration into the full platform is still in progress.
- **Methodology Pipeline:** We also refined our data ingestion pipeline. Our system now supports multi-format uploads with OCR for scanned documents, extracts metadata, and transforms inputs into a unified JSON structure. Structured outputs are stored in PostgreSQL, while raw files are securely stored in object storage (MinIO/S3).

3 Methodology

3.1 Data Preprocessing

We designed our data pipeline to handle both structured and unstructured health data. Patients and doctors can upload records in multiple formats (PDF, images, scanned forms, and text). For image and scanned documents, we apply OCR to extract textual information, followed by metadata tagging (patient ID, date, record type). All uploaded inputs are transformed into a unified JSON format that allows consistent storage and processing.

- Structured data (e.g., lab results, prescriptions, vital signs) are parsed into relational tables in PostgreSQL.
- Unstructured data (e.g., doctor notes, discharge summaries) are preserved in object storage (MinIO/S3) while their extracted text is prepared for NLP-based analysis.
- Data cleaning includes standardizing medical terms, handling missing values, and validating timestamp formats.
- Sensitive identifiers are encrypted, and audit logs are automatically generated to track every access or modification.

3.2 Model and Algorithm Selection

Our AI pipeline is built as a standalone Flask microservice that currently focuses on analyzing structured health data. It applies rule-based anomaly detection to flag irregular patterns in vital signs and lab results, while also generating reminders for medication schedules and basic lifestyle recommendations. Looking ahead, we plan to extend this engine with two additional capabilities:

- **NLP & Clinical Concept Extraction:** Using BioBERT embeddings to process unstructured text (doctor notes, summaries), enabling semantic search and concept recognition[2]. This will allow patients and doctors to query health records more naturally.
- **Hybrid Recommender System:** Combining rule-based reasoning (e.g., guideline-driven alerts) with statistical models to deliver personalized health recommendations and risk predictions.

This modular approach allows us to continuously improve AI functions without disrupting the main platform.

3.3 Evaluation Metrics

We will evaluate our system at multiple levels:

- **Model-level performance:**
  - As identifying high-risk patients is the top priority, we will focus on maximizing Recall for the High class in the three-class prediction problem (Low, Moderate, High).
  - While Recall is emphasized for High-risk cases, we will also monitor Precision and F1 to ensure a balance between false positives and false negatives.
  - Fairness and bias checks across demographic subgroups.
- **System-level performance:**
  - Response time benchmarks: < 2 seconds for record retrieval and < 3 seconds for AI insights.
  - End-to-end workflow validation (upload $\rightarrow$ storage $\rightarrow$ AI $\rightarrow$ dashboard visualization).
  - Load testing to assess scalability under concurrent users.
- **User experience and usability:**
  - Surveys and feedback sessions with test users to assess dashboard accessibility, clarity of AI recommendations, and trust in data security.
  - Qualitative evaluation of explainability — ensuring patients and clinicians understand why a particular alert or recommendation was generated.

4 Preliminary Results and System Walkthrough

4.1 Current System Functionality

At this stage, we have implemented and tested several key components of the Medivise platform:

- **Backend Services:** Secure APIs now support patient and doctor record management, including uploads, prescriptions, and filtering. Authentication is handled with JWT, and role-based access control ensures that only authorized users can view or modify records. All data transmissions are encrypted via TLS, and audit logs are recorded for accountability.
- **Database and Storage:** The PostgreSQL schema has been deployed with relational tables for users, roles, health records, prescriptions, and audit logs. Queries return results in under two seconds during testing. Raw files are stored in MinIO/S3, while structured data is normalized and indexed for efficient retrieval.
- **Frontend Dashboards:** Patients can upload medical records, search and filter their history, and receive health reminders. Doctors can access their assigned patients' records and prescriptions. Messaging and secure communication channels are functional, and all features have been tested on desktop browsers.
- **AI Microservice:** The Flask-based AI module currently analyzes structured health data. It can detect anomalies in vital signs, generate medication reminders, and provide simple lifestyle suggestions. Initial testing with sample datasets shows promising accuracy, though full integration into the platform is still ongoing.

4.2 Demonstrated Workflow

A typical user flow in our system currently looks like this:

1. A patient logs into their dashboard and uploads a scanned prescription.
2. The system applies OCR to extract text and metadata, then converts the information into a standardized JSON format.
3. Structured information (e.g., medication names, dosage, timestamps) is stored in PostgreSQL, while the original file is archived in object storage.
4. The AI microservice analyzes dosage schedules and generates a reminder.
5. The reminder is displayed in the patient dashboard, while the doctor dashboard updates with the new record.

4.3 Early Results

- **Performance:** API queries and record retrieval meet the current target of < 2 seconds. The AI service responds within 3 seconds for anomaly detection and reminders.
- **Reliability:** Role-based access control and encryption mechanisms work consistently across tested scenarios.
- **Usability:** The dashboards have been tested on laptops and PCs, with smooth user interactions. Mobile responsiveness remains a task for future development.
- **AI Insights:** Early outputs show that the AI module correctly identifies irregular values and produces relevant reminders, although more evaluation with clinical data is required.

5 Project's Timeline and Milestones

We organized our project into a six-week development schedule, with clear milestones for each phase. The table below summarizes the planned tasks, achievements, and current status.

Week	Milestones	Status
Week 1	Finalize requirements; draft architecture design; set up GitHub repository; assign roles.	Completed
Week 2	Implement initial PostgreSQL schema; set up backend authentication (JWT) and RBAC; create frontend skeleton with login and dashboards.	Completed
Week 3	Expand backend APIs for patient/doctor record management; integrate PostgreSQL with live environment; add audit logging; connect frontend to backend (Axios).	Completed
Week 4	Develop AI microservice for structured data analysis (anomaly detection, reminders, lifestyle tips); refine data ingestion pipeline with OCR and JSON transformation.	Completed (AI running standalone; integration pending)
Week 5	Integrate AI microservice with backend and frontend; implement semantic search with BioBERT embeddings; optimize dashboards for mobile devices.	In Progress
Week 6	Perform end-to-end testing; evaluate performance (latency, accuracy); refine security (2FA, key rotation); prepare final deployment and documentation.	Upcoming

Progress Summary

- We have successfully completed the **first four weeks of milestones**, including backend, database, frontend dashboards, and an operational AI microservice.
- Now, we are coming to **Week 5**, focusing on integration and advanced features like semantic search.
- In **Week 6**, our priority will be system testing, security hardening, and final deployment preparation.

6 Team Contributions

Our project is driven by three members, each taking responsibility for different system components while supporting one another during integration and testing.

Team Member	Role	Contributions
Nguyen Tien Nhan	AI & Security Engineer	He developed the Flask-based AI microservice for structured data analysis, including anomaly detection, medication reminders, and lifestyle suggestions. He worked on the data ingestion pipeline with OCR and JSON transformation. He also implemented data security measures such as TLS for data in transit, AES-256 encryption for data at rest, and strict role-based access control.
Le Mai Thanh Son	Frontend Developer & UI Lead	He built the patient and doctor dashboards in React with Tailwind, implementing record upload, search/filter, and secure messaging features. He integrated the frontend with backend APIs using Axios, validated smooth interactions on desktop browsers, and started optimizing the interface for mobile responsiveness.
Nguyen Xuan Truong	Backend & Database Developer	He designed and deployed the PostgreSQL schema for users, roles, health records, prescriptions, and audit logs, ensuring normalization and indexing for fast queries. He also developed secure backend APIs with JWT authentication and RBAC, integrated audit logging, and connected the backend with the database. He tested functionality thoroughly using unit tests and Postman.

Each member contributed to complementary modules: one focused on backend and database stability, one on frontend usability, and one on AI and security. This distribution allowed the team to make consistent progress across all layers of the system while maintaining clear ownership of tasks.

7 Difficulties

During the development of Medivise, we encountered several challenges that affected both our technical progress and workflow. These difficulties are summarized below:

- **Integration of the AI Module:** Although the AI microservice works well in isolation, integrating it into the main platform has been challenging. Defining consistent input/output formats, handling asynchronous requests, and ensuring smooth communication between the backend and AI service remain ongoing issues.
- **Processing Unstructured Data:** While structured data ingestion has been successful, working with unstructured clinical notes is more complex. Implementing OCR at scale and preparing data for NLP models such as BioBERT requires additional preprocessing steps, storage design, and optimization for retrieval speed.
- **Security and Compliance:** Ensuring strong data security is a priority, but it introduces difficulties. Implementing encryption, access control, and audit logs works in testing; however, we still face challenges in key management, token refresh strategies, and preparing for advanced requirements such as two-factor authentication and secure key rotation.
- **Testing and Evaluation:** Most of our current tests use synthetic or sample datasets. Without access to real-world clinical data, it is difficult to fully evaluate model accuracy, fairness, and bias.

We also need more comprehensive end-to-end and load testing to validate latency goals under concurrent users.

- **User Interface Responsiveness:** Our frontend dashboards are stable on desktop, but adapting them for mobile responsiveness has proven more difficult than expected. Given that many patients may primarily use mobile devices, this remains a critical challenge to solve before deployment.

8 Conclusions

So far, we have successfully built the core foundations of the Medivise platform. Our backend services, database schema, and frontend dashboards are stable and functioning as expected. The AI microservice for structured data analysis is operational, demonstrating the ability to detect anomalies, generate reminders, and provide lifestyle suggestions. Together, these results show that our system can already deliver meaningful value for both patients and clinicians.

The project is promising in several ways. First, our modular design makes it easier to scale and integrate new features such as semantic search with BioBERT and hybrid recommendation engines. Second, our security-first approach ensures that data privacy and compliance are central to every stage of the system. Finally, the usability of the dashboards and early testing of performance (retrieval within 2 seconds and AI responses within 3 seconds) indicate that the platform can meet practical requirements for everyday use.

Looking forward, our focus will be on integrating the AI microservice with the main platform, optimizing the dashboards for mobile devices, and conducting comprehensive end-to-end testing. We will also prioritize strengthening security features, refining the semantic search pipeline, and evaluating the AI models with more representative datasets. With these next steps, we expect Medivise to evolve into a secure, intelligent, and user-friendly personal health record system capable of transforming passive data into actionable healthcare insights.

References

[1] Chayakrit Krittanawong, Albert J. Rogers, Kipp W. Johnson, Zhen Wang, Mintu P. Turakhia, Jonathan L. Halperin, and Sanjiv M. Narayan. Integration of novel monitoring devices with machine learning technology for scalable cardiovascular management. *Nature Reviews Cardiology*, 18(2): 75–91, Oct 2020. doi: https://doi.org/10.1038/s41569-020-00445-9.

[2] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4), Sep 2019. doi: https://doi.org/10.1093/bioinformatics/btz682. URL https://academic.oup.com/bioinformatics/advance-article/doi/10.1093/bioinformatics/btz682/5566506.

[3] P. C. Tang, J. S. Ash, D. W. Bates, J. M. Overhage, and D. Z. Sands. Personal health records: Definitions, benefits, and strategies for overcoming barriers to adoption. *Journal of the American Medical Informatics Association*, 13(2):121–126, Mar 2006. doi: https://doi.org/10.1197/jamia.m2025. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1447551/.