Project name:

Client name and affiliation:

Team members:

Description of task

The Clinical Skills Expansion Project for MedSimAI aims to improve the training process for medical students by utilizing clinical reasoning, interdisciplinary scenarios, and physical examination simulations in one platform. As students learn most from practical examples, this project will provide students with opportunities to engage with structured experiences in simulated clinical visits to build their diagnostic reasoning, team-based decision making and physical examination skills. We will build this product initially around a Minimum Viable Product (MVP) and gradually increase its functionality.

The preliminary requirements revolve around three core features: Talk-Aloud Recording & Evaluation, Basic Integrated Case Flow, and Physical Exam Decision Tree. Our initial product will allow students to communicate their clinical reasoning, simulate the role of different users in multi-disciplinary scenarios and receive AI-driven feedback on physical exam simulations. We will build the MVP with a focus on scalability, accuracy, and integration into the existing MedSimAI framework.

The final deliverables for the MVP will include a working prototype with fully functional core components, an evaluation framework which will be used to assess reasoning and decision making of users, and a tracking system to observe students over time. This will be used to evaluate their progress and receive tailored feedback. Future phases will incorporate more advanced AI analysis, role variation, and interactive 3D physical exam models. We hope to make MedSimAI an adaptive learning platform.

Preliminary architecture

Existing System Interfaces

The Clinical Skills Expansion project will integrate with the following existing systems in MedSimAI:

1. *MedSimAI Frontend*

- Technology: React, Tailwind CSS
- Structure: The UI is modular and component-based, with key components such as Research.jsx and PerformanceSummary.jsx.
- Integration Points:
    - Implement new UI components for talk-aloud clinical reasoning, interdisciplinary case flows, and physical exam interactions.
    - Update the grading interface to incorporate new evaluation criteria.

2. *MedSimAI Backend*

- Technology: Flask/FastAPI, SQLAlchemy ORM for database interactions
- Current Features:
    - Stores structured JSON-based evaluation rubrics for grading criteria.
    - Implements a scoring system that evaluates clinical skills based on predefined performance criteria.
- Integration Points:
    - Extend backend APIs to store, process, and analyze talk-aloud recordings.
    - Add support for multi-role case interactions and physical exam simulations.
    - Implement real-time AI-driven feedback mechanisms.

3. *AI-Assisted Clinical Evaluation System*

- Purpose: Evaluates student performance using structured LLM-powered prompts based on predefined medical rubrics.
- Key Features:
    - Uses structured JSON-based evaluation criteria to assess key diagnostic questions.
    - Identifies whether students addressed required symptoms for conditions like URI, bronchiolitis, and croup.
    - Extracts direct quotes from conversations as supporting evidence.

    - Assigns structured, explainable feedback for grading.
- Integration Points:
    - Frontend: Displays AI-generated feedback in the grading interface.
    - Backend: Processes student transcripts using structured prompt-based evaluations.

Scope of Work: Self-Contained vs. Modifications

The project will require both standalone components and modifications to existing code:

- Modifications to Existing Code:
    - Extend backend APIs for grading and case logic.
    - Update frontend UI to display new clinical reasoning feedback.
    - Modify the evaluation system to support new structured grading criteria.
- Self-Contained Components:
    - Talk-Aloud Recording & Evaluation: Built as a standalone module with frontend, backend, and AI integration.
    - Physical Exam Decision Tree: Implemented separately for ease of integration into MedSimAI.

Technical Considerations

Frontend Stack

1. React: React is the preferred choice for frontend for modern application because of its component-based architecture. It is best suited for dynamic and modular interfaces such as ours. Given its virtual DOM, it can allow for fast updates when users interact with cases, submit responses and receive AI feedback
2. Tailwind CSS: Tailwind CSS has utility-first styling which allows for quick UI development while keeping our codebase clean. Given our requirements, where we have multiple UI elements such as cases, chat interactions, forms, tailwind CSS suits our case best.
3. D3.js or Chart.js: Data visualization is one of the most important parts of our application. We will use this to track student performance, and visualize trends for tailor feedback. D3.js and Chart.js provide us with highly customizable visualizations. We will alternate between the two given how much customization is required.

Backend Stack

1. Flask: Flask is a lightweight web framework that allows for thorough API development. It also works well with FastAPI which provides asynchronous capabilities and built in data validation. MedSimAI requires us to handle concurrent requests therefore, FastAPI is preferred for its high performance
2. PostgreSQL: We will use a relational database (PostgreSQL) for data storage. We will store case data, evaluation rubrics, and AI feedback. Postgres allows us to store JSONB fields which allows for flexible data storage. We will leverage ACID compliance to ensure data integrity.
3. JWT or OAuth: We will work with either JSON Web Tokens or OAuth for authentication. JWT is preferred because of its stateless authentication which allows for scalability but OAuth may be necessary when integrating universities in our platform. This will also provide secure single sign-on and handle FERPA regulations.
4. Jenkins: We will use Jenkins for continuous integration and development. This will reduce manual errors in deployment. It will also help with automated testing, security and container builds.

Infrastructure

1. AWS ECS: We will use ECS for hosting because of scalability and its container orchestration. We will be using Docker or another containerized application which will allow for better integration
2. S3: We will use S3 for data storage. This is cost-effective and highly available. We will store documents like transcripts, recordings, etc. here

Development methodology and outline plan

The team will follow an Agile development methodology with bi-weekly sprints, allowing for iterative development, continuous testing, and stakeholder feedback.

Development Approach

- **Sprint-based development**: Work will be divided into bi-weekly sprints, each focused on implementing a subset of features.
- **Continuous integration & deployment (CI/CD)**: New features will be tested and integrated in a staging environment before production deployment.
- **Regular stakeholder check-ins**: Frequent meetings with the Client to ensure feature alignment with user needs.
- **Code reviews & version control**: Code will be managed through GitHub using pull requests and peer reviews.

Testing Strategy

- **Unit testing**: Validating individual components (frontend and backend).
- **Integration testing**: Ensuring smooth communication between UI, backend APIs, and database.
- **End-to-end testing (E2E)**: Simulating real-world workflows to validate correctness.
- **User acceptance testing (UAT)**: Testing with medical students and faculty to refine usability.

Next Steps & Timeline

Sprint 1: Planning & Setup (2 Weeks)

- Get MedSimAI running locally for all team members.
- Familiarize with the codebase structure (frontend, backend, database, evaluation system).
- Identify key integration points for new features.
- Set up development workflows (Git, CI/CD pipelines, testing framework).

Sprint 2: Foundation & Backend Setup (4-6 Weeks)

- Design and implement database schema updates for new features.
- Create backend API endpoints for talk-aloud, case transitions, and physical exams.
- Ensure initial frontend-backend integration (basic data flow established).

Sprint 3: MVP Development: Talk-Aloud & Clinical Reasoning (2 Weeks)

- Implement audio recording and transcription pipeline.
- Develop GPT-based clinical reasoning evaluation.
- Display basic feedback on the frontend.

Sprint 4: MVP Development: Integrated Case Flow (2 Weeks)

- Implement role transitions (e.g., nurse → intern → doctor).
- Develop logic for tracking student decisions across roles.
- Ensure UI updates dynamically based on role transitions.

Sprint 5: MVP Development: Physical Examination Simulation (2 Weeks)

- Implement a step-by-step guided physical exam module.
- Develop decision tree-based AI evaluation for physical exams.
- Integrate structured feedback into the grading system.

Sprint 6:Refinement & User Testing (Ongoing)

- Run user testing sessions with MedSimAI stakeholders.
- Identify usability issues & feedback improvements.
- Optimize AI evaluation accuracy & frontend UX.

Coordination plan

Client Visibility and Communication

We will conduct regular bi-weekly live product demonstrations with the client during meetings at 7:00 pm Monday via zoom (we will make them weekly in February to speed up the onboarding process), where the client can also provide interactive feedback and test new features. In between meetings, recording of demos can be shared upon the client's request.

To keep the client an opportunity to track progress closely, we will email a brief weekly status report every Friday containing features completed this week, progress against sprint goals, risks requiring client attention, and resource needs or decisions required. Quick inquiries will be communicated in the Discord channel with the client. A Notion subpage with meeting minutes and action items, technical documentation, and user testing results will also be created. Weekly documentation updates every Thursday.

Team Communication and Collaboration

Sprint planning will occur bi-weekly on Mondays to prioritize the backlog, set goals aligned with MVP deadlines, break down tasks, assign owners, and estimate effort. Daily group chat standups at 11:00 am ensure visibility into progress, blockers, and task updates. A weekly technical deep-dive on Wednesdays will cover architecture reviews, code assessments, and technical issues. Also, we will evaluate achievements, discuss areas for improvement, and refine team working agreements.

For communication, discord serves as the primary channel with dedicated spaces for general updates, technical discussions, client interactions, and urgent issues. Formal communications and documentation are handled via email. GitHub will be used to support code collaboration. Issues will be posted on GitHub and discussed during scheduled meetings. Zoom is used for remote meetings.

Project Tracking and Management

We use Notion for tracking requirements and managing user stories with detailed classification based on priority, status, and dependencies. Weekly backlog refinement ensures requirements stay updated, acceptance criteria are clarified, and client feedback is recorded.

A risk register in Notiont tracks potential issues that are categorized into technical, schedule, resource, and quality risks. The team reviews top risks weekly, updates mitigation strategies, and escalates critical concerns. Issue management is split between GitHub for technical bugs and

Notion sprint roadmap for project tracking, with priority levels ensuring timely resolution. Daily standups address high-priority issues, while bi-weekly summaries are shared with the client.

Risk analysis

Time risk

This project is designed to be completed in one semester. During the development of the project, additional complexities may be introduced, preventing the project from being delivered on time. Some examples include:

1. Miscommunicated requirements
2. Underestimation of implementation time

Resource & Technology risk

During development, we may encounter situations where we do not have necessary resources or technological expertise to continue on some tasks.

First of all, as an AI-driven project, the computational units are vital. Despite the completion of model training, model inference relies on stable GPU infrastructure, which we as students do not have reliable access to. This could lead to undiscovered problems during testing or deployment of the project, and may delay development if the issue remains unsolved.

Next, to accommodate many professional functionalities such as 3D modeling or voice translation that we do not have an expertise in, we may resort to open source APIs or libraries for our project. This introduces potential risks of library failure or server failure on the developers end of the libraries/APIs, which may block our project for an unpredicted amount of time. Additionally, library updates could also trigger potential failures.

Lastly, as a project that targets primarily medical students, we as developers may need to acquire necessary medical-knowledge in order to serve the users best. However, these training could take extra time and may need medical experts' assistance, and their availability could become a potential risk that could block the project.

Data security risk

The platform may store some sensitive data such as patient information that may be used for model training or scenario simulations. These data must be kept confidential according to relevant laws and regulations. We may encounter challenges during development due to unfamiliarity with the law.

Human resources

The size of the developers group is small, and each member has other commitments, so task distribution could be a challenge to the team.

Mitigation strategy:

In order to mitigate time risks, we plan to conduct pre-scheduled sprint meetings with our stakeholders to enhance communication of requirements and timelines. Additionally, we would divide tasks into concrete milestones and more specific tickets to avoid large differences between expected time and actual development time.

Regarding resources and technology, we plan to take advantage of AWS, Azure and other cloud platforms to quickly obtain scalable solutions without having to invest in hardware. We plan to prioritize libraries and APIs with large community support, such as PyTorch. Lastly, we will communicate our needs for domain-knowledge early with our clients, who may have connections with medical students available for us to consult.

To address data security risks, we plan to use PostgreSQL as our database, which provides data security. Additionally, we will consult the client about the source, access permissions and relevant regulations before we make any data visible to users.

To address human resources risks, we plan to have additional weekly developers' meetings in addition to meeting with the client, distributing specific, detailed task tickets to members and tracking progress on assigned tasks.

Overall, though this project is exposed to some risks, we have plans to mitigate these risks and are prepared to proceed.

Discussion of business considerations

Feasibility analysis

Resource analysis

There are a few constraints, specifically time and domain knowledge. There is a little less than three months to complete this project, and it will take some time to onboard onto the codebase, along with addressing with other blockers that may show up as this project advances. However, the team contains the necessary skills to complete this project and meet the client's requirements before the deadline. Given the team's skillset and time dedicated to the project, there is sufficient time to onboard and implement the client's MVPs successfully.

Technical Analysis

The following analysis evaluates the feasibility of implementing the client's MVP features based on the team's technical skills:

1. *Talk-aloud for clinical reasoning*

   The primary features in this product will be the ability to transcribe students' dialogue, utilize machine learning to return feedback, and store common mistakes. The current system already supports dialogue transcription and evaluation, making it feasible to expand with additional functionality. The team has the technical skills in frontend and backend to create another interface that provides this feature, making this feasible.
2. *Basic integrated case flow*

   Along with the requirements from the feature above, the team will also require a state management system to track user decisions and transition between roles. Since the roles are predetermined (nurse, intern, doctor, etc.), the complexity of the task is reduced. This new requirement is within the team's capabilities, so this feature is feasible.
3. *Physical exam decision tree*

   This feature includes a step-by-step guided physical exam where students select and interpret examination findings. AI generates feedback based on students' decisions. Since the module follows a structured progression rather than a complex branching system

(which may be explored after MVP features are completed), the team has the skills to successfully implement this feature.

Feasibility Conclusion

Given the available resources, technical constraints, and project scope, the MVP features are **feasible** within the given timeframe. The project should proceed as planned.