

**Final Project Submission Instructions**

Each project will be carefully evaluated based on the Judging Criteria outlined below. The submission is complete only when all elements in the checklist are submitted in the requested formats. Please follow the instructions carefully; projects with missing or partial elements will be automatically disqualified. After completing this document, do not forget to email it to **ruhealthhack@gmail.com** before 4 PM on Sunday, October 27, 2024. Any submissions received after the deadline will be automatically disqualified from judging.

**Judging Criteria:​**

* **Innovation & Creativity:**How unique and original is the project?​
* **Technical Implementation:**Quality of code, cloud architecture, and data integration.​
* **Impact**: Relevance to healthcare and the potential real-world impact.​
* **Presentation**: Clarity of the project demo and documentation.​
* **Feasibility & Scalability:**Can this project be scaled up and deployed in a real healthcare setting?​

**Code Submission:**All code must be uploaded and submitted via a version control platform (e.g., GitHub) by the deadline. Any updates or versions submitted after the deadline timestamp will be automatically disqualified.

**Presentation/Demo:**Teams must submit a brief project presentation or demo in video format (maximum of 5 minutes) explaining the project concept, technology stack, and healthcare impact. Upload the video to YouTube and ensure that appropriate permissions are set for access by the RUHealthHack team.

**Working Prototype:** A functional prototype/software or proof of concept is encouraged. It should demonstrate how cloud technologies and healthcare data are being utilized.

**Documentation:**Please complete the following document with all the requested information and send it to **ruhealthhack@gmail.com** by 4 PM on Sunday, October 27, 2024.​

**Team #:** \_\_17\_\_\_\_\_\_\_\_

**Team Name:** Journey

**Team Lead Contact Information:**

**Project Theme**: Patient Education

**Project Title**: Journey Application – Patient Journey for Results

Code Submitted

* Link to Github repository: \_\_\_\_ [GitHub - romrom-20/RUHealthHack\_team17\_Journey](https://github.com/romrom-20/RUHealthHack_team17_Journey)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* Overview of the solution:

Our solution provides an accessible and easily digestible way for patients to understand their health data via a health tracking application. Information about health data in laboratory reports and doctor after visit summaries can be difficult for individuals to understand, when patients receive these documents they may be informed that some of their health metrics are outside the normal range. These abnormal values can cause undue stress and worry, while these abnormal values may not even be a cause for concern. Our application empowers patients with the knowledge to understand when and which health metrics may be a cause for concern and prompts them to seek additional advice from their doctor or if needed will connect the user with an appropriate specialist in their geographic location. The application is meant to assist patients along their healthcare journey, from their initial doctors visit, to assessing results of bloodwork and other diagnostic testing, to follow-up visits, and long-term disease maintenance. Our application aptly named “Journey” will provide a dashboard for patients to upload their health reports, like laboratory results (such as hemoglobin A1c, comprehensive metabolic panels, and lipid panels), as well as after visit summaries from their doctors. Once these documents are uploaded, our proprietary AI model will process the documents and inform users about health metrics of concern in their most recent health reports as well as from trends in routine and recurrent health reports. When a health metric such as hemoglobin A1c or LDL is detected as abnormal, information about what these metrics are and how they relate to a patients health will be presented to the user. Additionally, guidelines for how to improve these health metrics to aid in bringing the values back within a normal range are suggested based on national guidelines. This information will aid patients in having more productive and informed conversations with their doctors upon follow-up visits when lab tests are ordered. To increase the efficiency with which individuals receive specialized care when health metrics of concern are detected, the application can connect users with appropriate specialists to schedule care. Setting up appointments as early as possible can alleviate large costs associated with more intensive care from health issues that worsen as a patient waits to visit their primary care physician. We can thereby provide cost savings to both patients and health management organizations by connecting users to healthcare earlier on in their healthcare journey.

* Details on cloud architecture and tools used

The application is fully set up on AWS to ensure reliability, security, and efficient processing. It uses Amazon EC2 to run the Node.js backend, with an Elastic Load Balancer that helps manage traffic, automatically adjusting to handle more or less demand. For storing user documents and reports, we use Amazon S3, where all files are securely saved and encrypted, with rules in place to archive older files to save costs. To handle NLP processing, a local language model is hosted on Amazon SageMaker, keeping data processing entirely within AWS to ensure privacy. Static content, like reports and images, is delivered quickly to users around the world through Amazon CloudFront, which also protects against attacks with AWS WAF. We use Amazon CloudWatch to monitor how the application is running, and CloudTrail to track actions taken within the system for added security. IAM roles are set up to control access, so only necessary services and users can reach S3, EC2, or SageMaker. In this flow, users upload documents to S3, which are then processed by the language model, and results are stored back in S3, ready to be quickly delivered by CloudFront. This AWS setup makes the application scalable, secure, and ready to meet user needs efficiently.

* Steps for installation or deployment

1. **Clone the Repository**: Open a terminal and run:

git clone

cd

1. **Install Dependencies**: Make sure Node.js is installed, then run:

npm install

1. **Set Up Environment Variables**: Create a .env file in the project folder with values for environment variables (like AWS credentials if needed). Example:

S3\_BUCKET\_NAME=your-bucket-name

AWS\_ACCESS\_KEY\_ID=your-access-key

AWS\_SECRET\_ACCESS\_KEY=your-secret-key

OPENAI\_API\_KEY = your-key

1. **Run the Application**: Start the application with:

npm start

1. **Access the App**: Open a browser and go to http://localhost:3000 (or whichever port it uses).

Presentation uploaded to YouTube with appropriate settings

Demo

* Link to Canva Video: <https://www.canva.com/design/DAGUy26xGg4/upaGZoSr2mmezzL1Mtcvww/view?utm_content=DAGUy26xGg4&utm_campaign=designshare&utm_medium=link&utm_source=recording_view>
* Link to Canva presentation: <https://www.canva.com/design/DAGUy26xGg4/CCV4Dl_7mk65rfE1fmabtw/edit?utm_content=DAGUy26xGg4&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton>