Progress Report

- Increment 1 -
Group #2

Authors:

Daivion Brooks
Brooks Berry
Jason Lee
Nadav Peper
Nandagopal Nair

1) **Team Members**

- *Daivion Brooks* - ***FSU ID: dlb22e / GitHub ID: dbrooks11***
- *Brooks Berry* – ***FSU ID: blb16m / GitHub ID: brooksberry10***
- *Jason Lee* – ***FSU ID: jsl23e / GitHub ID: JasonLee4489***
- *Nadav Peper* – ***FSU ID: np22bd / GitHub ID: nadavp5***
- *Nandagopal Nair* - ***FSU ID: nn22e / GitHub ID: nanduf***

2) **Project Title and Description**

Title*: Med Assist App*
Description: The purpose of this app is to act as a daily assistant for medical patients, offering a set of features designed to optimize medical practices. It allows the user to add daily symptoms and treatment trackers, graphs for trend, and a find a doctor feature.

3) **Accomplishments and overall project status during this increment**

Accomplishments:

- *Established the core structure of the application using React (frontend) and Flask (backend).*
- *Implemented user authentication features, including:*
    - *Registration API route for new users.*
    - *Login route with support for either email + password or username + password.*
    - *Refresh route to handle revoked or expired JWT access tokens.*
- *Built additional backend routes for handling medical data:*
    - *User profile retrieval (individual and all information).*
    - *Symptom tracking: add and retrieve symptoms.*
    - *Treatment tracking: add and retrieve treatments*
- *Developed the initial frontend interface, including:*
    - *A working home page with navigation structure.*
    - *User-friendly design with a consistent theme for the app.*
- *Established the database schema to provide support for application modules.*
    - *Enabled database migrations*
- *Implemented treatments CRUD*
- *Registered the treatments blueprint to avoid circular imports*

Overall Project Status: *The Med Assist App is in a strong foundational stage. Core backend functionality for user management and medical data tracking is in place, and we are in the process of finalizing the database schema and data models to align with project requirements. The frontend provides a clean starting point for user interaction. Our next steps will focus on expanding CRUD functionality (update/delete symptoms and*

treatments), enhancing frontend-backend integration, improving UI/UX, and adding more advanced features such as data visualization (graphs/trends) and a potential "find a doctor" functionality.

We are on track with our goals for the first increment and have a solid base to build upon

for future iterations.

4) Challenges, changes in the plan and scope of the project and things that went wrong during this increment

Challenges:

Daivion Brooks *-One of the main challenges for me during this increment was related to backend development and learning new technologies. Since Flask and Python were new to me, there was a steep learning curve in understanding how to properly structure routes, connect to the database, and securely handle user data. Initially, the backend used basic POST and GET methods that stored all user information directly in the database, including plain-text passwords. After further research, I saw this as extremely insecure, which required refactoring. I then implemented proper password hashing and improved how user data was stored. Another challenge was the form-handling process. At first, Flask-WTF was used to build forms; however, because our frontend is built with React, I saw this approach was incompatible. After realizing this, I had to scrap all the Flask-WTF forms I worked on and refactor the backend to send and receive JSON data instead. To make this work smoothly, Marshmallow was added for data validation and serialization. These challenges were addressed through research, trial and error, and restructuring the code to align better with secure practices.*

Nadav Peper *- For me, one of the main challenges was adjusting to a different project structure than I'm used to. In past projects, I've worked mostly with Next.js, where both the backend and frontend are integrated into a single framework. With this project, we're using Python and Flask for the backend and React for the frontend, which was new to me. Learning how to structure the two separately and get them to communicate smoothly took some time. On the frontend side, I struggled with creating a clean and modern design from scratch. At first, I was piecing things together manually, and it felt messy and time consuming. I discovered the HeroUI library, which really helped with that. Using the components from that library made the app look more modern and consistent, while also speeding up development. Another issue I faced was handling JWT tokens coming from the backend. I had to learn how to properly store and use these tokens in the frontend so that users stayed authenticated across sessions and requests. At first, I ran into problems where tokens weren't being refreshed or validated correctly, causing unexpected logouts and broken API calls. However, after debugging for a while, I fixed those issues and it all works now.*

Nandagopal Nair *– My biggest challenge was onboarding to an existing Flask codebase and adding Treatments CRUD without touching teammates' code. I initially hit a circular import when wiring the treatments blueprint; I fixed it by importing/ registering the blueprint inside create_app() after initializing db/jwt/migrate. I also ran into env issues (missing SECRET_KEY/JWT_SECRET_KEY) that caused JWT failures/500s; adding those to .env and documenting them unblocked auth. Local testing on Windows had a few traps: PowerShell execution policy blocked venv/scripts, and cURL quoting/backticks caused 400 Bad Request; I switched to Invoke-RestMethod and produced a copy-paste PowerShell flow (plus a VS Code REST Client file) so the team can reproduce the full login to CRUD to 404 sequence. Another gotcha was 404s when <id> didn't match the JWT sub; I solved it by decoding the token and using that uid automatically. Along the way I also fixed a pagination dump bug in Symptoms (return .items instead of the Pagination object) and aligned the Git workflow (Issue linkage, feature/* branch naming, small README tweaks) so future work is smoother.*

Brooks Berry *- My biggest challenge was taking the initiative to design the app. I had a general idea of how I wanted this app to be, essentially taking the role of the client, as well as the role of a product designer to fit the requirements of the app. I have worked a little bit with python, flask, and react before, but have never initialized a web app before while integrating the specific technologies that we are using in this app. I worked*

to get things installed, initialized, and configured correctly and attempted to make an onboarding document that was professional and hopefully allowed for smooth onboarding for my teammates. My teammates and professor offered great feedback and suggestions in the app design which allowed us to achieve a successful first increment in our app development.

Jason Lee *- For me, the biggest challenge came from the fact that this was my first time working on a project that is relatively huge and complex compared to other programming assignments. First of all, I had to learn how to install and configure React, Docker, and the required dependencies as outlined in the README. Understanding how all of the pieces of code fit together and the course of familiarization also took time because of the complexity and size of the project. Another challenge that I faced during Increment 1 was deciding how the database structure should be structured. I updated parts of the schema based on the project proposal and the discussion from the meeting so that it could better support the feature that we planned to implement.*

Changes in Plan/Scope:

Overall, the initial scope of the project remained consistent with our original plan. From the beginning, the team agreed on the features and the technology stack (React for frontend and Flask for backend). The only notable changes had to do with natural progression of the app such as the creation of the database schema and the data points that we would be adding routes for. These were not specified in the initial requirements and were largely created as we developed. One significant decision that we made in terms of changing project plan was switching from vanilla java script using React to using typescript with React for ease of testing. Other notable changes included additions of libraries that we plan to use such as plans to use Toastify and Werkzeug security. We also initially said that we would implement other features and decide on implementing authentication if time provided, but we decided that authentication was an essential feature for an app built for sensitive data collection and storage.

Issues Encountered:

Fortunately, no major issues occurred during this increment. We each worked on our own branches, merged into the main branch, and had no significant conflicts. Aside from the need to refactor backend code for security and frontend compatibility, development has been relatively smooth. The incremental changes to the schema were manageable and did not disrupt progress.

5) Team Member Contribution for this increment

Daivion Brooks:

progress report*: Wrote the accomplishments and overall project status, focusing on backend progress and summarizing team achievements.*

requirements and design document*:completed operating environment and assumptions/dependencies*

implementation and testing document*: Documented backend tools (Flask-jwt,Marshmallow,) and explained how I tested authentication and partial CRUD routes using Postman.*

source code*: Built some part of the backend with Flask, including registration, login, JWT authentication,and symptom/treatment API routes (create and read), while testing endpoints in Postman.*

Brooks Berry:

progress report*: contributed to the changes in plan and scope section.*

requirements and design document*: wrote the project design overview section, wrote the functional requirements section.*

implementation and testing document*: reviewed the document, did not write anything.*

source code*: Came up with app design in the initial project proposal. Set up the initial app skeleton with its current architecture. Set up flask on back-end and react on front-end with hookup to test API routes in frontend/src/App.js. In /backend/main, implemented app initialization in __init__.py and initial configuration*

in config.py. Set up docker-compose.yml. Set up requirements.txt. Started initial database SQL schema. Made README for onboarding of team members.

Jason Lee:

***progress report**: Contributed to the 'Accomplishments' section by adding the accomplishments listing, and 'overall project status' section by refining and adding some contents, and 'Challenges' section by adding my personal challenges during Increment 1.*

***requirements and design document**: Contributed to the 'Non-functional Requirements' by writing out overall content, 'Use Case Diagram' and 'Class Diagram and/or Sequence Diagrams' section by drawing diagrams.*

***implementation and testing document**: Contributed to the 'Programming Languages' section by revising content about SQL, and 'Execution-based Functional Testing', 'Execution-based Non-Functional Testing', 'Non-Execution-based Testing section' by adding listings.*

***source code**: Reconstructed schema.sql based on the project proposal, fixing PostgreSQL syntax issues, adding new tables/columns, and improving overall consistency with constraints including data types and namings.*

Nadav Peper:

***progress report**: Created the demonstration video and wrote some challenges*

***requirements and design document**: Added some stuff about database safety*

***implementation and testing document**: Did not do much, focused on making the video*

***source code**: Developed and styled the entire frontend so far using ReactJS and HeroUI. Built the page layouts (home, navbar, footer, user profile page, sign in, sign up) and integrated JWT tokens to work with the backend for authentication.*

Nandagopal Nair:

***progress report**: Drafted the Treatments CRUD milestone notes, challenges (circular import, PowerShell policy), and added testing details.*

***requirements and design document**: Documented Treatments data model/fields and API contract (request/response, pagination, auth scope), and error codes.*

***implementation and testing document**: Wrote end-to-end test steps using PowerShell; included token decoding of JWT sub to ensure correct user scoping.*

***source code**: Implemented Treatments CRUD routes/blueprint; registered blueprint safely in `create_app()`; fixed Symptoms pagination dump; added README `.env` keys; added `treatments-username-flow.http` and `scripts/treatments-crud-copypaste.ps1` for reviewers.*

6) Plans for the next increment

Backend Plans:

- *Extend existing API routes to support full CRUD functionality for both symptoms and treatments (adding update and delete endpoints in addition to create and read).*
- *Improve error handling and input validation to ensure data integrity and better user feedback.*
- *Continue refining the database schema as needed to capture additional details for symptoms, treatments, lab information, etc.*

Frontend Plans:

- *Expand the UI/UX design by improving styling, layout, and accessibility.*
- *Implement more pages such as Symptoms and Treatments.*
- *Connect new frontend components to backend API routes for real-time data handling.*
- *Begin implementing basic graphing/visualization features to track trends in symptoms and treatments over time.*

7) Stakeholder Communication

***Dear Stakeholders**,*

We are pleased to share an update on the progress of the Med Assist application. Our team has been working diligently to establish the foundation of the platform, and we are excited about the momentum we have built.

Over the past development cycle, we successfully implemented the core infrastructure of the application. Patients can now create secure accounts and log in, ensuring their information is handled safely. In addition, we have built functionality to record daily symptoms and treatments, laying the groundwork for users to track their medical data over time. The initial version of the user interface has also been completed, offering a simple and intuitive starting point for interaction.

As with any project in development, we encountered some challenges. In particular, we had to refine our approach to ensure patient data was stored in a secure and standardized way, which required reworking parts of our system. These adjustments have strengthened the integrity of the platform and will benefit patients in the long run. We also made small refinements to our data model as we identified additional information that would be valuable to capture. These types of adjustments are a natural part of building a robust solution and are expected in projects of this scale.

Looking ahead, our next phase will focus on expanding functionality and improving the user experience. Specifically, we plan to:

- *i) Provide full control over recorded data by allowing users to update and remove entries.*
- *ii) Enhance the application's navigation so patients can more easily move between different areas of the app.*
- *iii) Improve the interface design to make it more engaging and accessible.*
- *iv) Begin incorporating data visualizations so patients can clearly see trends in their symptoms and treatments.*

We remain committed to delivering a product that will empower patients to take a more active role in managing their health. The work completed in this first stage has provided us with a strong foundation, and we are confident that the upcoming enhancements will bring us closer to achieving the full vision for Med Assist.

Thank you for your continued support and confidence in this project. We look forward to sharing further updates as development progresses.

8) Link to video

https://www.youtube.com/watch?v=t7zq3zzfQ8o