

Health Log System

APP Name Vita



Prepared for

Alex Schmidt

Prepared by

Nicholas Gonzalez, Valentine Jingwa, Tamara Mahmoud,

Jean-Pierre Nde-Forgwang, Jonah Scott

Software Development – Cohort A

School of Advanced Digital Technology

SAIT

12 November 2023

Health log System Documentation	4
Introduction	4
Who: Target Users	4
What: System Functionality	5
When: System Accessibility	5
Where: Platform Availability	5
Why: Purpose and Rationale	5
Team constitution.....	5
Team Members and Roles	5
General Team Rules:	6
Communication.....	6
Meetings	6
Work Distribution	6
Decision Making	6
Responsibilities:	7
Team Leader – Tamara Mahmoud	7
Quality Assurance - Valentine Jingwa.....	7
Developer – Valentine Jingwa, Jean-Pierre Nde-Forgwang	7
User Interface (UI) Designer – Jonah Scott, Nicolas Gonzalez	8
Timeline:	8
Missing Deadlines	8
Poor Quality Work	8
Lack of Communication	9
Use case diagrams and use case descriptions	10
User Use Case	10
Brief Descriptions.....	10
Use Case Description in Extended Format	11
Alternate Flow	12
Error Flow	12
Admin Use case.....	13
Brief Descriptions.....	13
Use Case Description in Extended Format	14
Alternate Flow	14

Error Flow	15
Class diagram	16
Admin:.....	16
Admin Management:.....	16
User:.....	16
User Management:	17
SubUser class:	17
Sequence Diagrams	17
Preliminary User Interface Design	19
• UI/Design - Prototype 1	19
Data Storage and Persistence.....	25
Home/Profile	25
Profile Detail	25
View Calendar	25
View Days Only	25
Day Selected View	26
Add or Record	26
Food Graph	26
Bowel Graph	26
Water Graph	26
Bladder Graph.....	27
Settings	27
Print Menu	27
Share Menu.....	27
Entity Diagram	28
Revised Class Diagram	26
System Architecture and Patterns.....	29

Health log System Documentation

Introduction

- **Statement of the project and project sponsor**

The project entails the development of a mobile application, with a preference for a mobile platform, dedicated to serving as a personal medical log. The primary target audience for this application comprises new parents of infants and toddlers, as well as the elderly. The fundamental purpose of the application is to facilitate the recording of medical incidents, enabling users to present a comprehensive log during medical consultations. The recorded information includes timestamps, event durations, and accompanying visual documentation such as photographs.

The motivation behind this initiative arises from the recognition of a prevalent issue wherein non-emergency medical appointments are often scheduled with a delay of several days. Consequently, individuals, particularly new parents, encounter difficulties recalling the precise dates and times of medical occurrences during doctor visits. Drawing from personal experience as a parent, the challenge of accurately documenting various aspects of infant care, such as diaper changes, bowel movements, and daily feeds, becomes apparent within the initial year of parenthood.

For the elderly demographic, particularly those residing alone or alongside peers of similar age, the application serves as a valuable tool for logging incidents such as falls, dietary intake, medication adherence through reminders, and even minor health complaints that, when recurring, may signal underlying health issues. In essence, the proposed application aims to address the inherent challenges associated with remembering and reporting critical medical information, enhancing the efficiency and accuracy of medical consultations for both new parents and the elderly.

Who: Target Users

Parents: Particularly new parents managing the health of infants and toddlers.

Adults: Individuals seeking a dependable method to track personal or family health data.

Elderly: Older adults need a straightforward system for regular health monitoring.

Home Aids: Caregivers assisting with health management in domestic settings.

What: System Functionality

Store Data: Secure retention of health-related information provided by users.

Graph Data: Visual representation of health data, facilitating easy comprehension and monitoring.

When: System Accessibility

Anytime Access: Users can log into the system at their convenience, ensuring continuous and flexible health tracking.

Where: Platform Availability

Mobile Application: An app designed for smartphone use, emphasizing portability and ease of use.

Online Platform: A web-based application accessible via various devices, catering to users preferring larger screens or different interfaces.

Why: Purpose and Rationale

Enhancing Health Management: By providing a tool that simplifies the logging and visualization of health data, the system aims to improve the overall management of health, particularly for those who may struggle with remembering or organizing such information.

Bridging Communication Gaps: The system serves as a bridge between users and healthcare providers. By presenting organized and visually interpreted health data, it facilitates more informed and productive medical consultations, especially when appointments are not immediate.

Team constitution

Team Members and Roles

- Nicholas Gonzalez

-Valentine Jingwa

-Tamara Mahmoud

- Jean-Pierre Nde-Forgwang

- Jonah Scott

Purpose: This contract is designed to clearly define what is expected of each team member, outline our shared responsibilities, and set the rules that will guide how we act and work together during the

Software Analysis course. Every member of our team has agreed to follow these rules to help us work well together and achieve our goals.

General Team Rules:

Communication

- Team members are expected to engage in open and transparent communication at all times. This includes sharing relevant project updates, expressing concerns or challenges faced, and providing constructive feedback.
- All interactions should be conducted with respect and professionalism. Team members should listen actively to one another and consider all viewpoints with an open mind before reaching a consensus.
- Team members must acknowledge receipt of emails or messages within 24 hours, even if a full response will take longer. This ensures that all members are aware that their communications have been received and will be addressed.

Meetings

- Team meetings will be held on a weekly basis to ensure consistent communication and alignment on project progress.
- The schedule for these meetings will be set in advance and agreed upon by all team members to accommodate everyone's availability.
- Attendance at all team meetings is mandatory to maintain a cohesive and informed team environment.
- In the event that a team member cannot attend a meeting, they are required to notify the Team Leader at least 24 hours in advance, except in cases of emergency.

Work Distribution

- Tasks and responsibilities will be allocated based on an equitable distribution of workload, taking into account each team member's skills, experience, and current commitments to ensure a fair balance.
- The Team Leader, in consultation with the team, will assign tasks. Assignments will be made transparently, and the rationale for task distribution will be communicated to ensure understanding and agreement.

Decision Making

- The team will aim for consensus in decision-making, where all members agree on the decision. This approach fosters collaboration and ensures that all voices are heard.

- If consensus cannot be reached after thorough discussion, decisions will be made by a majority vote. Each team member has one vote, and the option with the most votes will be selected.

Responsibilities:

Team Leader – Tamara Mahmoud

- Project Planning: Develop detailed project plans that outline key milestones, deliverables, and timelines. Adjust plans as necessary in response to any project shifts.
- Team Coordination: Coordinate the efforts of all team members to ensure that tasks are aligned with project goals and are completed on schedule.
- Conflict Resolution: Identify and address any team conflicts or issues that arise, facilitating a positive and productive work environment.

Quality Assurance - Valentine Jingwa

- Develop Test Plans: Create detailed, structured test plans and cases that cover all aspects of the application's functionality, including edge cases and potential failure points.
- Execute Test Cases: Carry out testing according to the test plans, including functional, system, integration, and user acceptance testing.
- Performance Testing: Assess the application's performance under various conditions to ensure reliability and speed.
- Usability Testing: Evaluate the application for user-friendliness and ensure it meets the usability standards required for the target audience.

Developer – Valentine Jingwa, Jean-Pierre Nde-Forgwang

- Write, Test, and Debug Code: Develop clean, efficient code based on project requirements. Conduct thorough testing and debugging to ensure functionality and reliability.
- Collaborate on Design: Work closely with the User Experience (UX) and User Interface (UI) designers to ensure that the application's design is effectively translated into functional code.
- Implement Features: Build and implement features and functionalities as outlined in the project specifications

Documentation Specialist – Jean-Pierre Nde-Forgwang

- Ensure Accuracy and Clarity: Verify that all project documentation is precise, clear, and easy to understand. This includes technical documentation, user manuals, and project reports.

- **Maintain Documentation Consistency:** Ensure consistency in style, format, and terminology across all documentation materials.
- **Collaborate with Team Members:** Work closely with developers, designers, and other team members to gather necessary information and insights for documentation.

User Interface (UI) Designer – [Jonah Scott](#), [Nicolas Gonzalez](#)

- Design the visual elements of the application, including layout, color schemes, and graphics.
- Collaborate with the UX Designer to ensure that the visual design enhances usability.
- Create and maintain a consistent visual identity for the application.

Timeline:

- *Project planning:* [September 21st 2023 - October 20th 2023]
- *Development phase:* [October 20th 2023 – November 20th 2023] (Not set for sure, can change if needed)
- *QA and Testing:* [November 20th 2023 – December 10th 2023]
- *Submission:* [December 11th 2023]
- *Tools and Technologies:* [SIM Modelling, Mermaid Diagramming, Visual Studio Code]
- *Code Repository:* GitHub
- *Communication:* Discord

Missing Deadlines

- *First Offense:* Written warning
- *Second Offense:* Mandatory extra workload in the next phase
- *Third Offense:* Escalation to course instructor

Poor Quality Work

- *First Offense:* Opportunity for redo within a 48-hour window
- *Second Offense:* Peer review and extra supervision for the next task
- *Third Offense:* Escalation to course instructor

Lack of Communication

- *First Offense:* Verbal warning
- *Second Offense:* Written warning
- *Third Offense:* Escalation to course instructor

By signing below, each team member acknowledges and agrees to the terms set forth in this team contract.

– Jonah Scott



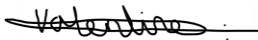
– Nicholas Gonzalez



– Jean-Pierre Nde-Forgwang



– Valentine Jingwa

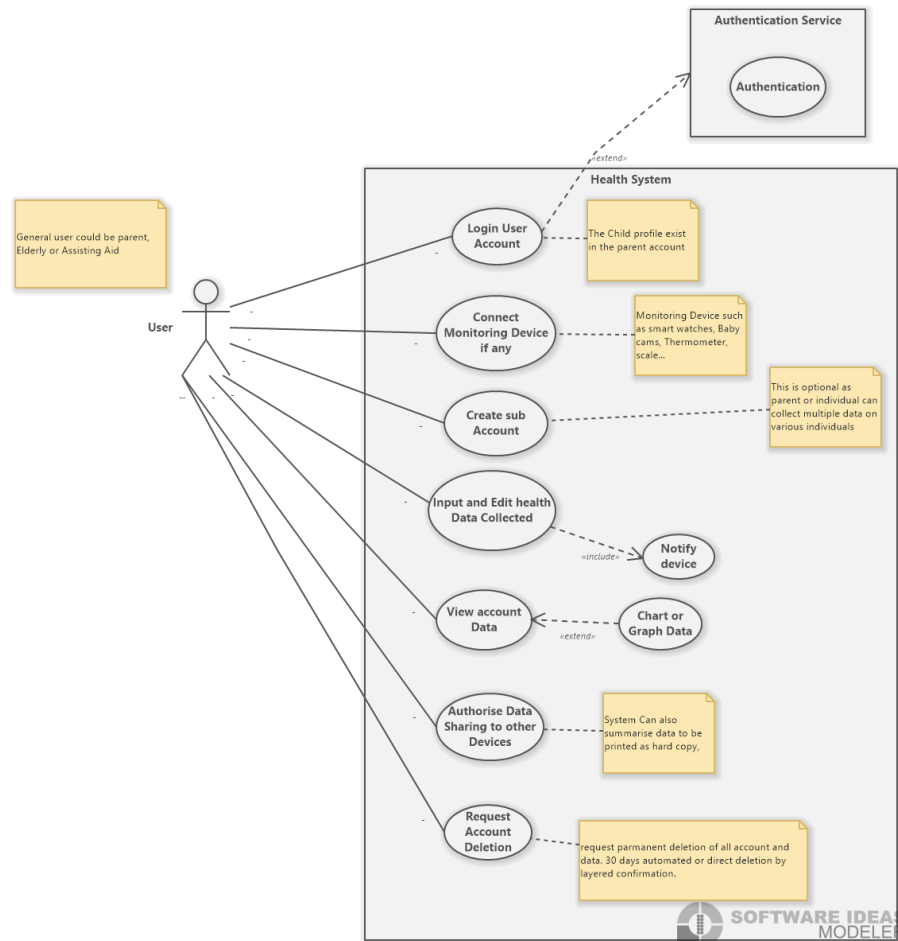


– Tamara Mahmoud



Use case diagrams and use case descriptions

User Use Case



Brief Descriptions

User: This entity is a general representation of whomever is using the app for a respective benefactor. This could be a parent, an adult, a caregiver, whomever.

Login User Account: This represents the action in which the actor can authenticate themselves and access data specific to themselves. Upon logging in, you have access to data within the app.

Connect Monitoring Device if any: This action is in relation to devices that track data, such as heart rate and steps.

Create Sub Account: This action is a representation of creating an account (child element) based on the main (parent element). These sub accounts have actions of the user account; except they cannot add nor delete an account.

Input and Edit Health Data Collected: This action allows users to input and fill in notes and forms based on the app; this action represents the entry of Data.

View Account Detail: This action represents the ability for end users to view information stored by the app and represents it in a visual manner.

Authorize Data sharing to other Devices: This action allows for the end user to share data with (child element) other devices. The end user will also be able to print out a summary of the data.

Request Account Deletion: This action represents the end user's ability to delete their account and subsequent data associated with it.

Use Case Description in Extended Format

Use Case 1: User Use case

Precondition: User has successfully logged in. User dashboard or home screen is displayed.

Postcondition: User has completed their desired actions and returned to the dashboard or logged out.

Limitation: User can abort any operation and return to the dashboard at any time.

Actor Action	System Response
-User logs in to their account.	-System authenticates the user and displays the main dashboard.
- User chooses to connect a monitoring device.	-System searches for available devices and pairs with the chosen device.
-User opts to create a sub-account	System presents a form to input data for the new sub-account and creates it upon submission.
-User inputs or edits health data collected.	System updates the health log with new or edited entries.
-User chooses to view account data.	System displays the account information, including health logs and connected devices.

-User authorizes data sharing to other devices or services.	System updates permissions for data sharing based on user preferences.
-User requests account deletion.	System initiates a confirmation process for account deletion.

Alternate Flow

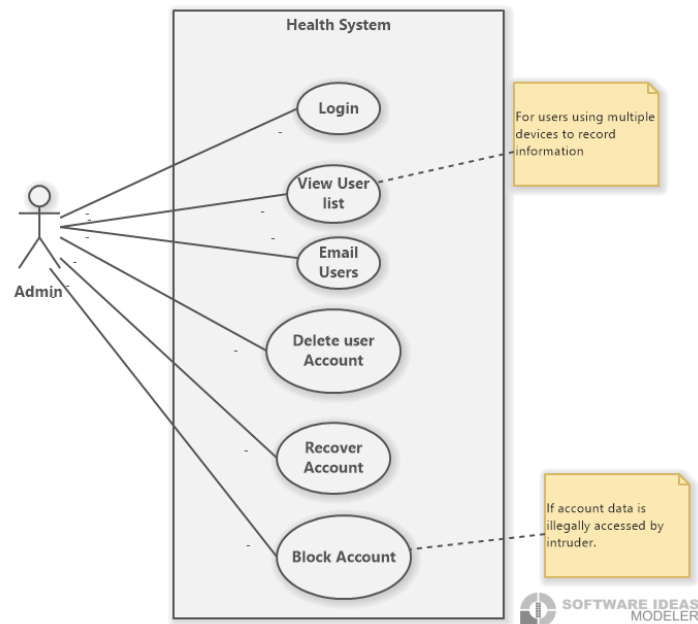
Actor Action	System Response
User selects to chart or graph data.	System generates visual representations of the health data.
User chooses to print the summarized data.	System provides an option to print the data summary.

Error Flow

At any time, invalid user information is entered, or a database operation fails.

Actor Action	System Response
An error occurs (invalid input, operation failure, etc.)	System displays an error message and offers possible actions to correct it.

Admin Use case



Brief Descriptions

Admin: This entity is a representation of an administrative user who has elevated permissions to manage the healthcare application.

Login: This use case represents the action where the admin authenticates themselves to access the administrative functions of the app.

View User List: In this use case, the admin can view a list of all user accounts within the system

Email User: This action allows the admin to communicate with users directly from the system.

Delete User Account: This use case enables the admin to permanently remove a user's account from the system.

Recover Account: In this use case, the admin has the ability to restore previously deleted or deactivated user accounts.

Block Account: This action allows the admin to temporarily disable a user account without deleting it.

Use Case Description in Extended Format

Use Case 2: User Use case

Precondition: Admin has access rights and credentials to perform system management tasks.

Postcondition: Admin has completed their management tasks and logged out or returned to the main admin dashboard.

Limitation: Admin actions must comply with system security and privacy policies.

Actor Action	System Response
Admin logs into the system.	System authenticates the admin and displays the admin dashboard.
Admin chooses to view the user list.	System displays a list of all users registered in the system.
Admin selects to email one or more users.	System provides an interface to send emails to users.
Admin decides to delete a user account.	System prompts for confirmation and deletes the user account upon confirmation.
Admin opts to recover a user account.	System presents options to search for and reinstate a deleted or deactivated account.
Admin wants to block a user account.	System allows admin to block the account to prevent access due to security or misuse concerns.

Alternate Flow

Actor Action	System Response
--------------	-----------------

Admin selects the option to send a bulk email.	System presents a template or editor to create the email content.
Admin composes and sends the email.	System distributes the email to the selected user list and confirms the action to the admin.

Error Flow

At any time, invalid user information is entered, or a database operation fails.

Actor Action	System Response
An error is detected during an admin operation.	System displays a detailed error message and logs the incident.



The class diagram above has 3 main classes which are the **User**, **Admin** and **Sub User**. The user as stated in the use case description could be an **adult**, **Parent**, **Elderly**, **Aid** or **caregiver**. All users can create Sub Users example infant(s), sibling, for a parent(elderly) or for work purposes as a caregiver or nanny. The third-party authentication and online database aren't connected in this diagram but would be incorporated into the health system for accessibility and backend storage.

Admin:

The admin or administrator just has properties such as name, email, address, ... This information is only entered upon launching the system for the first time and the only information required to update would be the password.

Admin Management:

Admin management contains the functions of the core functions the admin could access from their device. A majority of what they can do is provide customer service for the user. They can recover deleted online stored accounts, block access to stolen accounts, and view users using online storage. The admin, for privacy reasons, cannot access the client data without client authority and password.

User:

The user class consists of the basic properties of the user which are the **name**, **id**, **cell number**, and **password**. The User can register sub classes under themselves and record information on behalf of those sub classes.

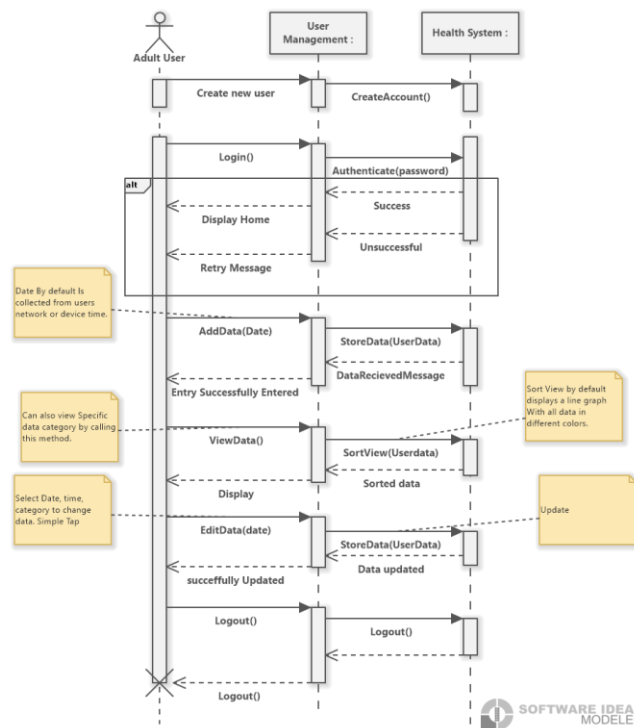
User Management:

The user management contains the core functions for the user. It also keeps track of the device it's registered in and also collects data from authorized applications(optional). The user can add whatever data they want to track it could be medication, the photos of an injury healing, food, bowel movement, and coloration and quantity of excretions (sweat, excrements, ...). They can view their data as a graph against time (any quantity they need to compare to).

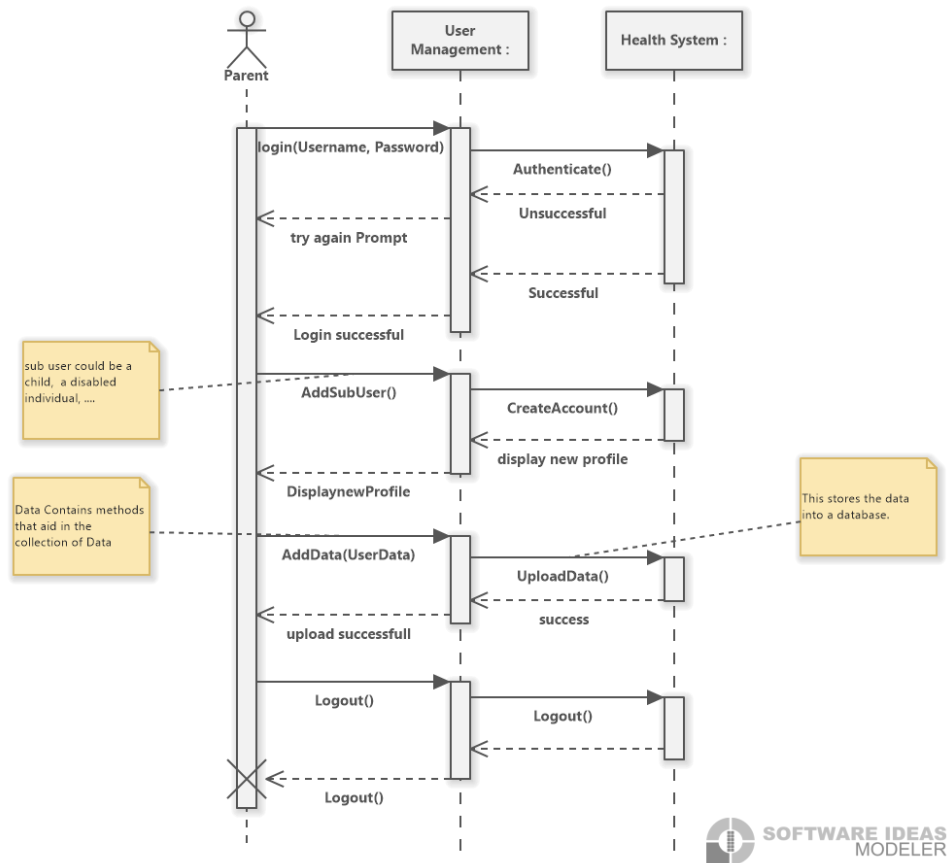
SubUser class:

This class is created by the User and stores functions just like a **user** class but cannot create subclasses.

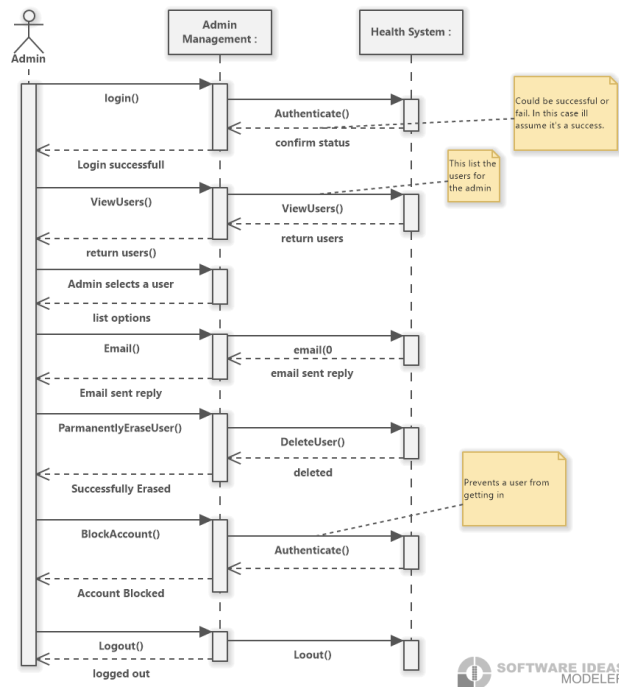
Sequence Diagrams



The sequence diagram above shows possible interactions that the possible adult user can make with the system. The sequence diagrams are brief as the system is a little more complex and could have many alternative routes to how the user can interact with the system.

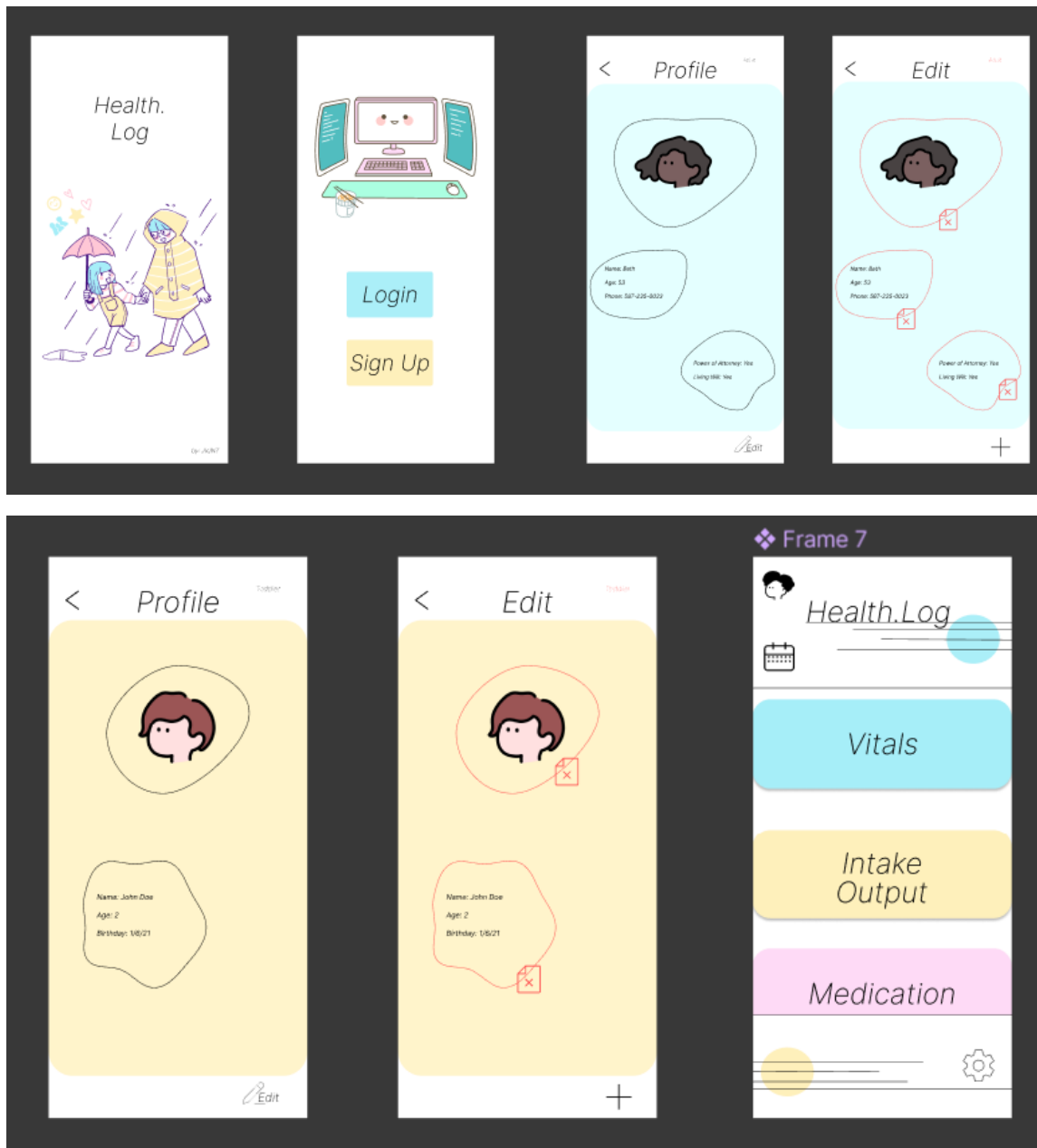


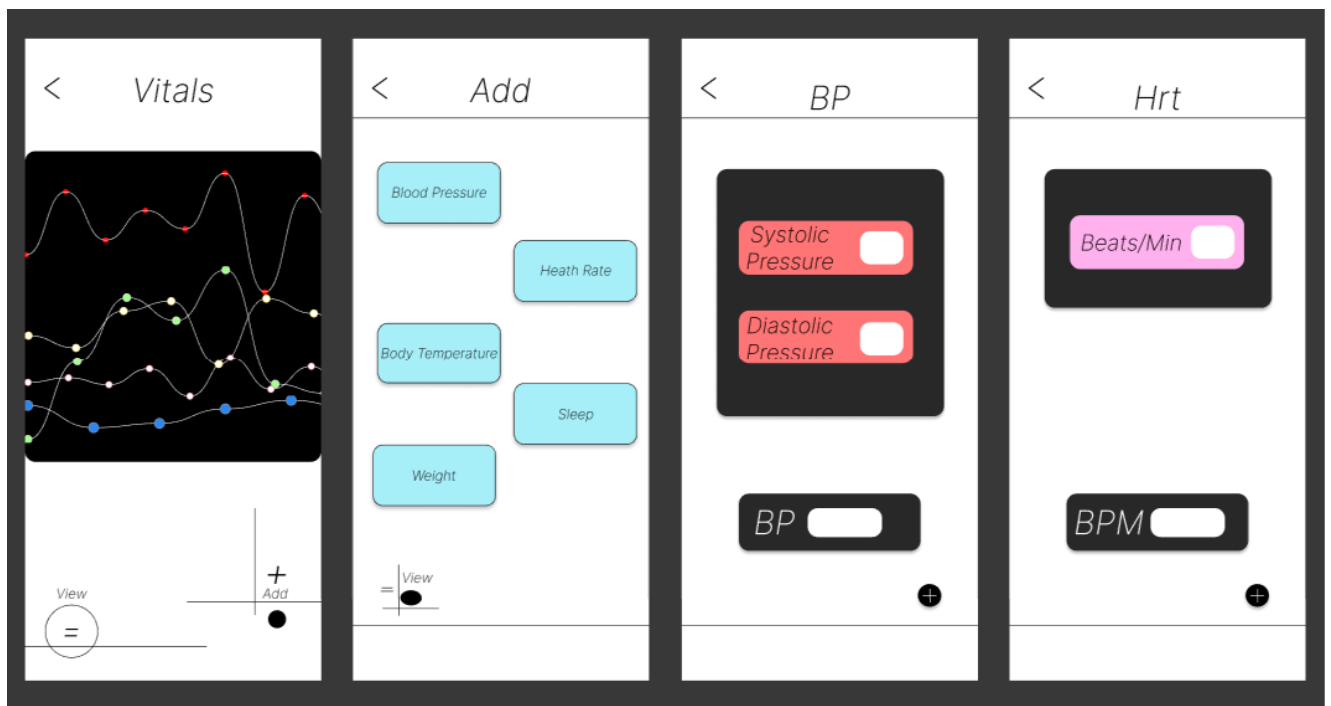
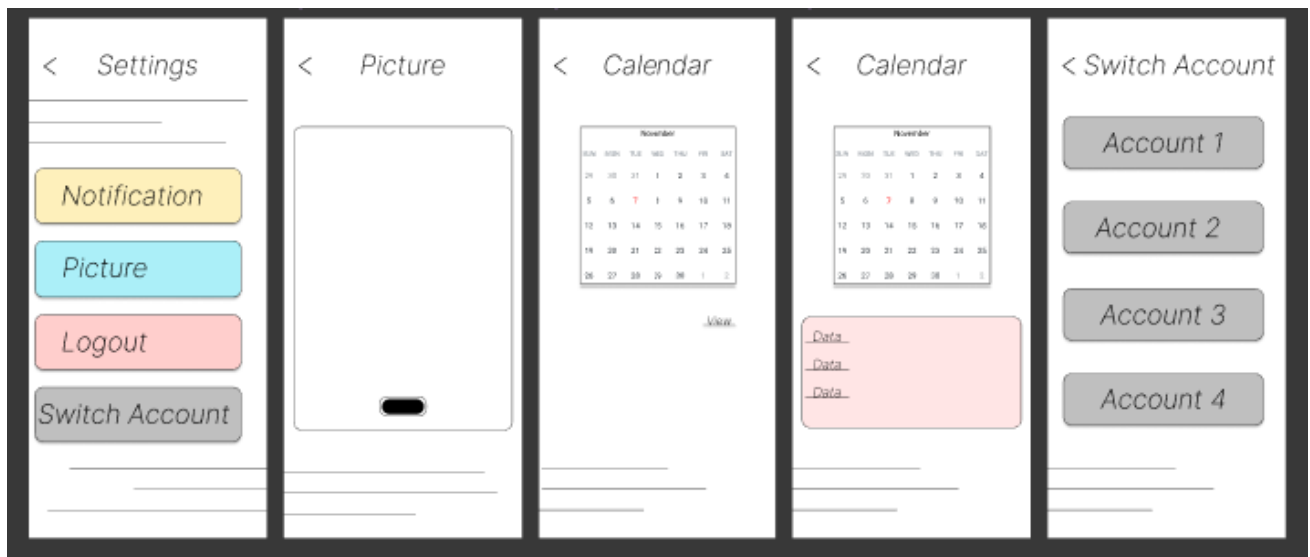
Above is the sequence diagram in the situation where the user is a parent.

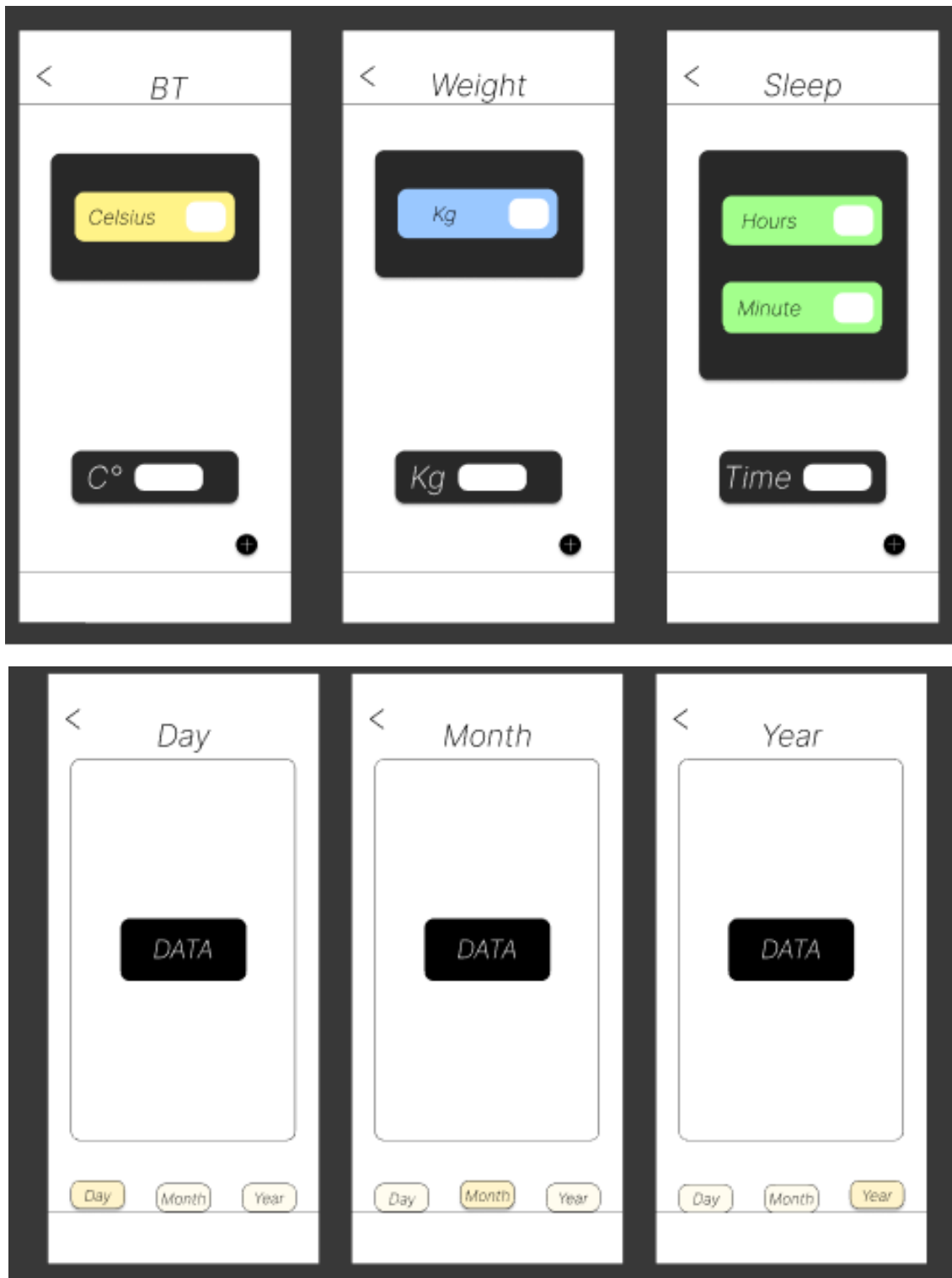


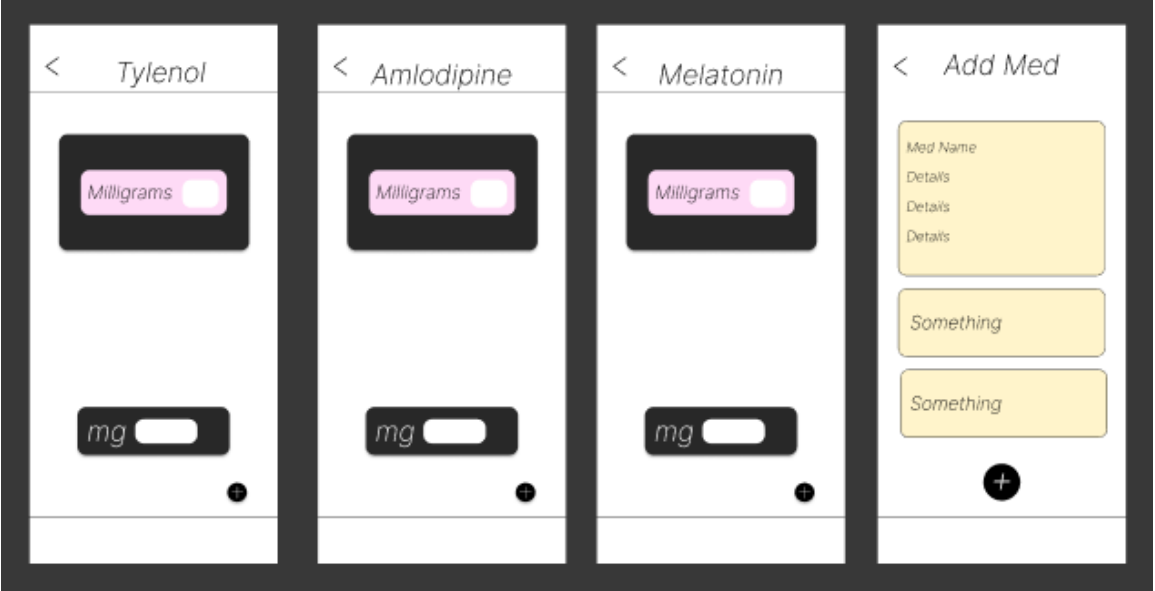
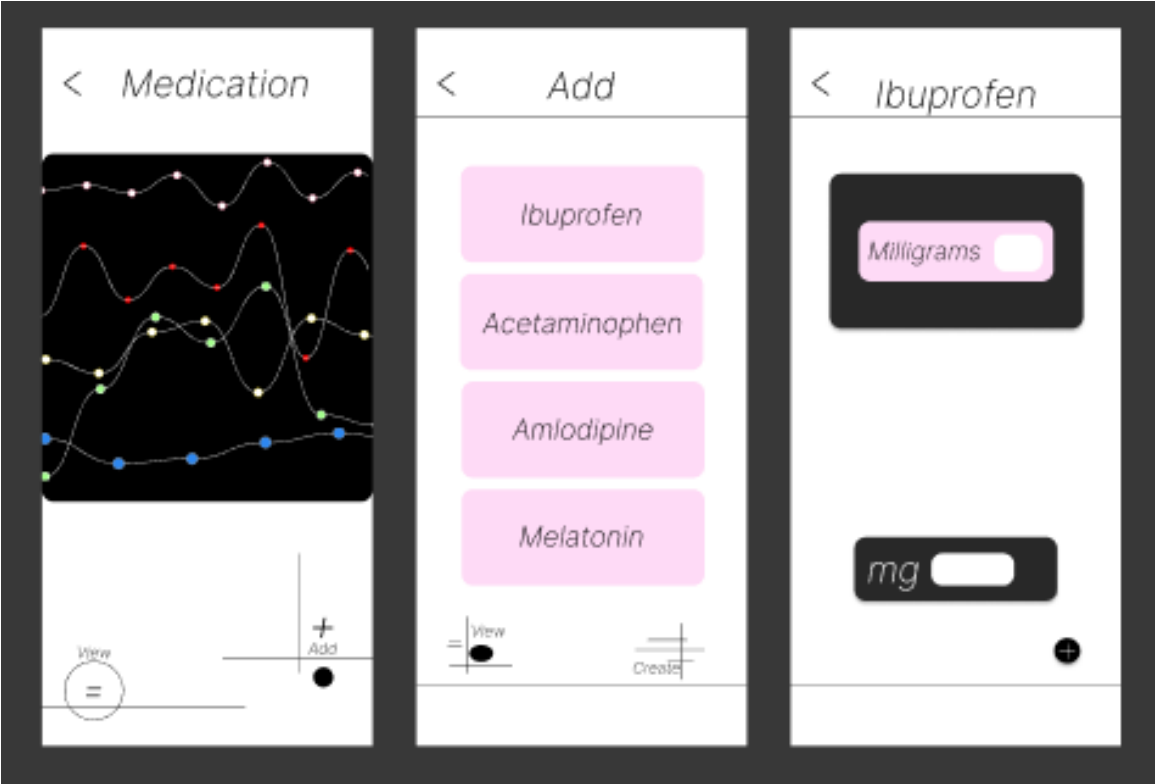
Preliminary User Interface Design

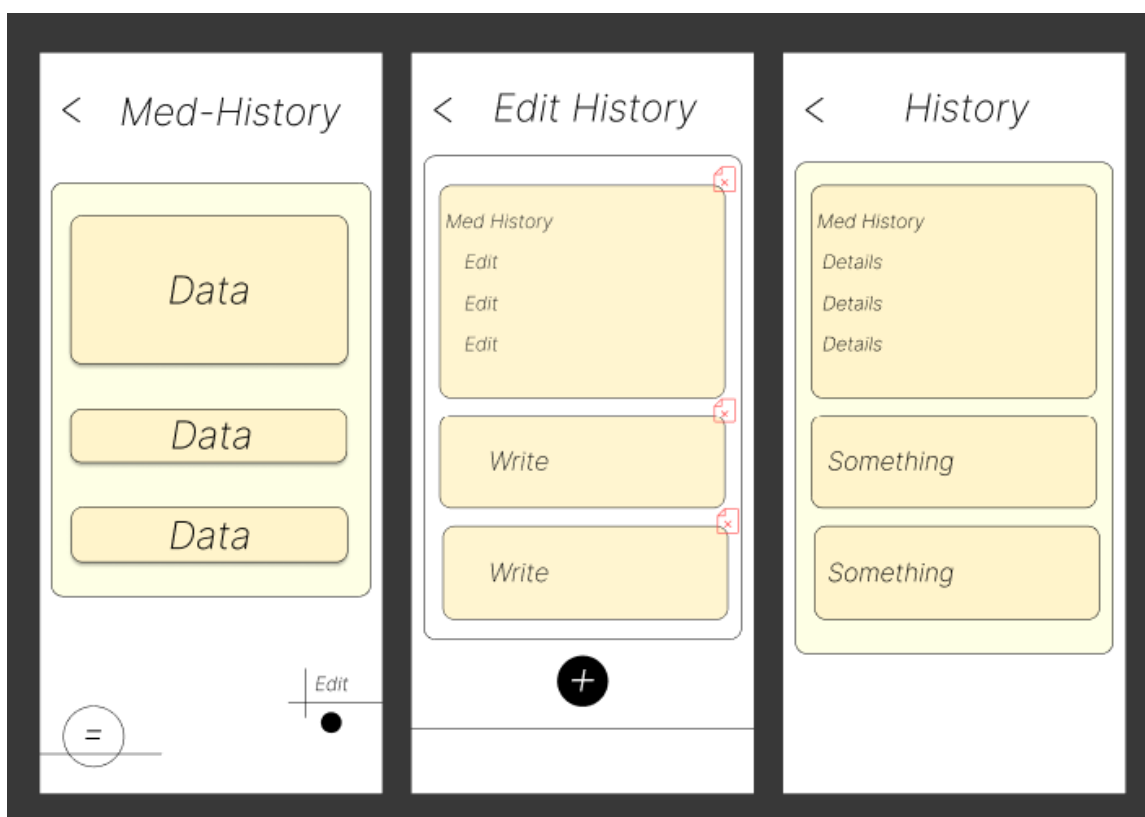
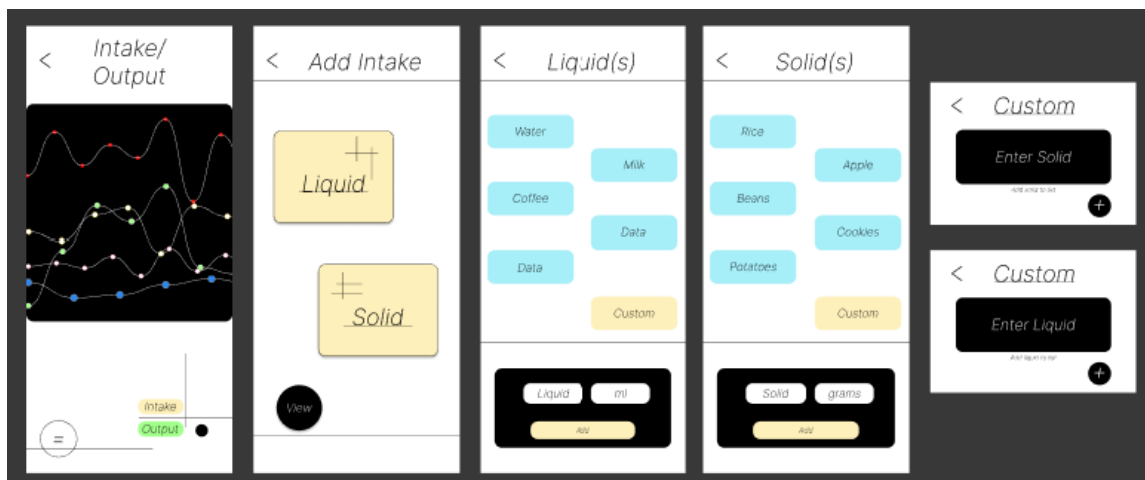
- [UI/Design - Prototype 1](#)
- [Visual](#)

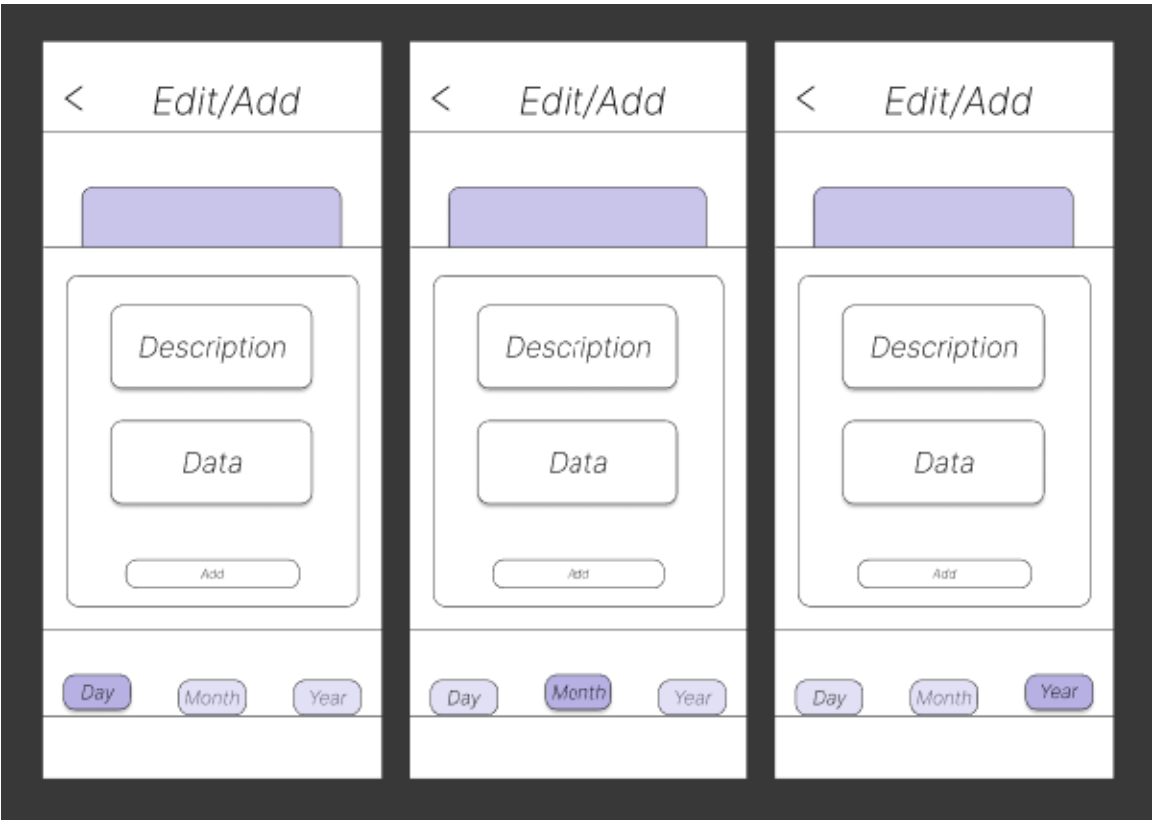
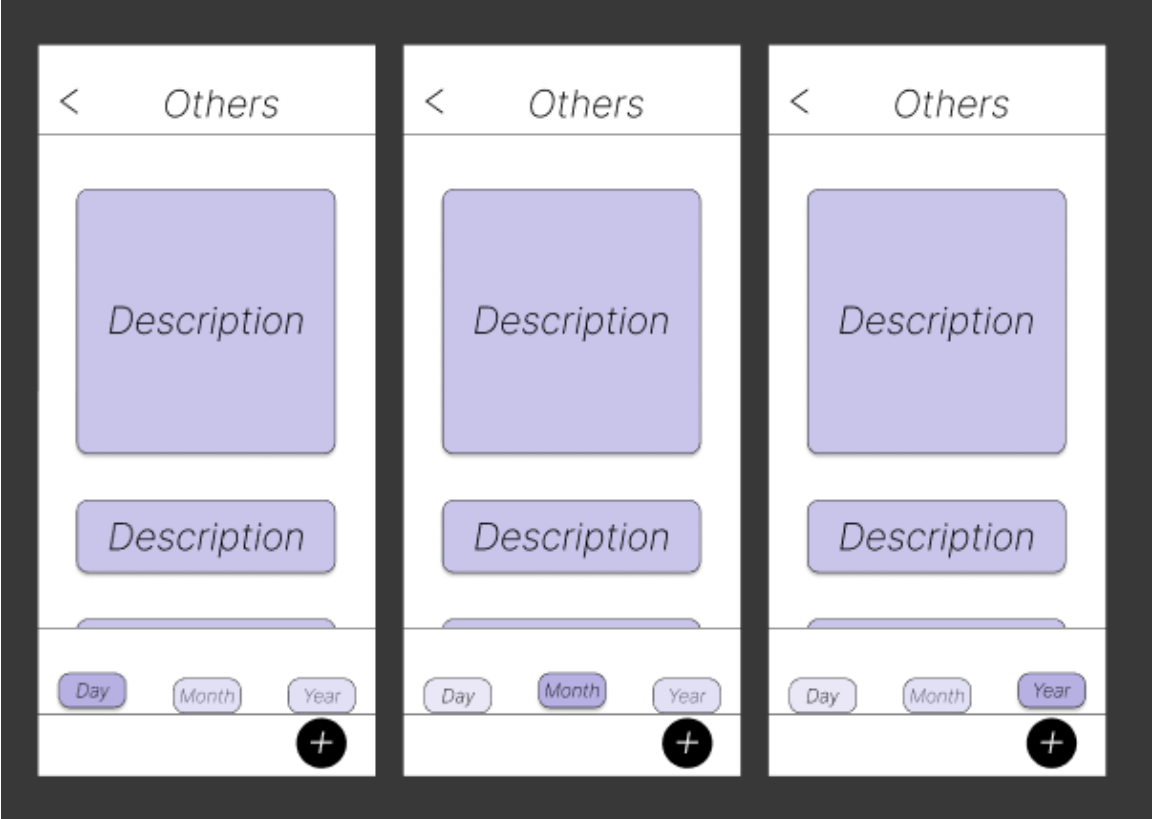












Data Storage and Persistence

When designing our application and our layout, we planned out how each screen would be implemented and what would need to come along with it. For our database needs and being a medical application, we want to always ensure security and efficiency when handling the users' important information. When going through our layout we have decided that for our:

Home/Profile

- The profile/account will store the users' personal information, health data and application preferences.
- Syncing the profile updates between the database and app when connected to the internet will be key in allowing offline accessibility.
- Local SQLite database can securely store the users' profile when created, and with its lightweight structure, it will have minimal impact on the devices storage while allowing quick access to data.

Profile Detail

- Our profile detail will cache and secure user health records/personal information for offline access.
- Encrypt the sensitive information that ensures the users' security and privacy.
- SQLite has a great encryption extension for storage, and with our app logging sensitive and private information, this is a key factor that this database addresses with data integrity and consistency.

View Calendar

- Once a user records an activity, appointment, medication intake, or diet update it will be logged and sorted through our calendar with the date and time of the recording.
- When a user opens the calendar screen, there will be an event log shown by a dot or highlight under the date on certain days where the user recorded an event for ease of access and availability to look back on previous recordings.
- SQLite has outstanding organizing and can date stamp recordings for future user requests, and having our calendar function SQLite is able to process those complex queries that a calendar function requires for sorting information.

View Days Only

- This will sort the users' recordings and display the time of which the recording was placed on the selected day.
- This will need a query system to help speed up and maintain the correct information to be displayed on the specific day.
- SQLite's indexing functionality can quickly retrieve stored health information by sorting the date for efficient and reliable representation.

Day Selected View

- Once the user has selected a day, the query system that displayed information that was placed in View Days Only will display the information that was logged.
- This screen will include a filtering system that allows the user to sort through a certain category if needed to reduce clutter, like medication intake, diet intake, bowel, bladder, water graphs.
- SQLite's previous View Days Only sorting allows fast retrieval of information to be displayed on the users' selected date input.

Add or Record

- Allows user inputs for new records, activities, or notes on the record to be stored.
- This entry will be stored and encrypted into the database to be sorted and categorized to be displayed if needed in the future.
- SQLite has a fast insert and update operation that will support real-time data recording to ensure the users accessibility to information. This database can record our users' input forms and categorize them efficiently.

Graph views will all be shown with the same settings and visualization preferences, and the database requirements specified in each graph form will be shared throughout each selected view:

Food Graph

- This screen will use the categorized information from previous user inputs and display them in a graph format for analysis on eating habits and trends in the user's diet.
- Our database will need to use indexes that improve the categorization functions and ensure consistency and efficiency of the data to be displayed in a graphical format and ensure the correct information is being displayed.

Bowel Graph

- This will be used to display the users' tracked bowel movements to be displayed in a graphical format to help visualize trends and inconsistencies to help spot possible health concerns.
- A database will need to correctly support and produce correct graphical information based on the users' query.

Water Graph

- Users stored and categorized inputs will monitor hydration levels depending on the user's timeframe choice.
- A database will be needed to ensure efficient updates and data retrieval from previous inputs and formatted into a graph.

Bladder Graph

- This will use the categorized and stored information to display urinary frequencies throughout the days, to spot inefficiencies and health concerns.
- Data storage will be optimized and displayed graphically for visualization and displayed based on the users' time frame specification.

For our applications graphical needs SQLite can handle the transference of data and structures based on the categorization and formatted into a graphical view. With this databases' real-time data recording, and fast reading operations, it can handle a high frequency of updates and formatting depending on the users' input.

Settings

- User profiles will have configuration and accessibility settings that will be stored and saved under the users' profile for logging out/in to be automatically changed on user sign in.
- SQLite can generate a database file that can be backed up and restored in case of an emergency which provides a flexible and secure settings manager for the users' settings.

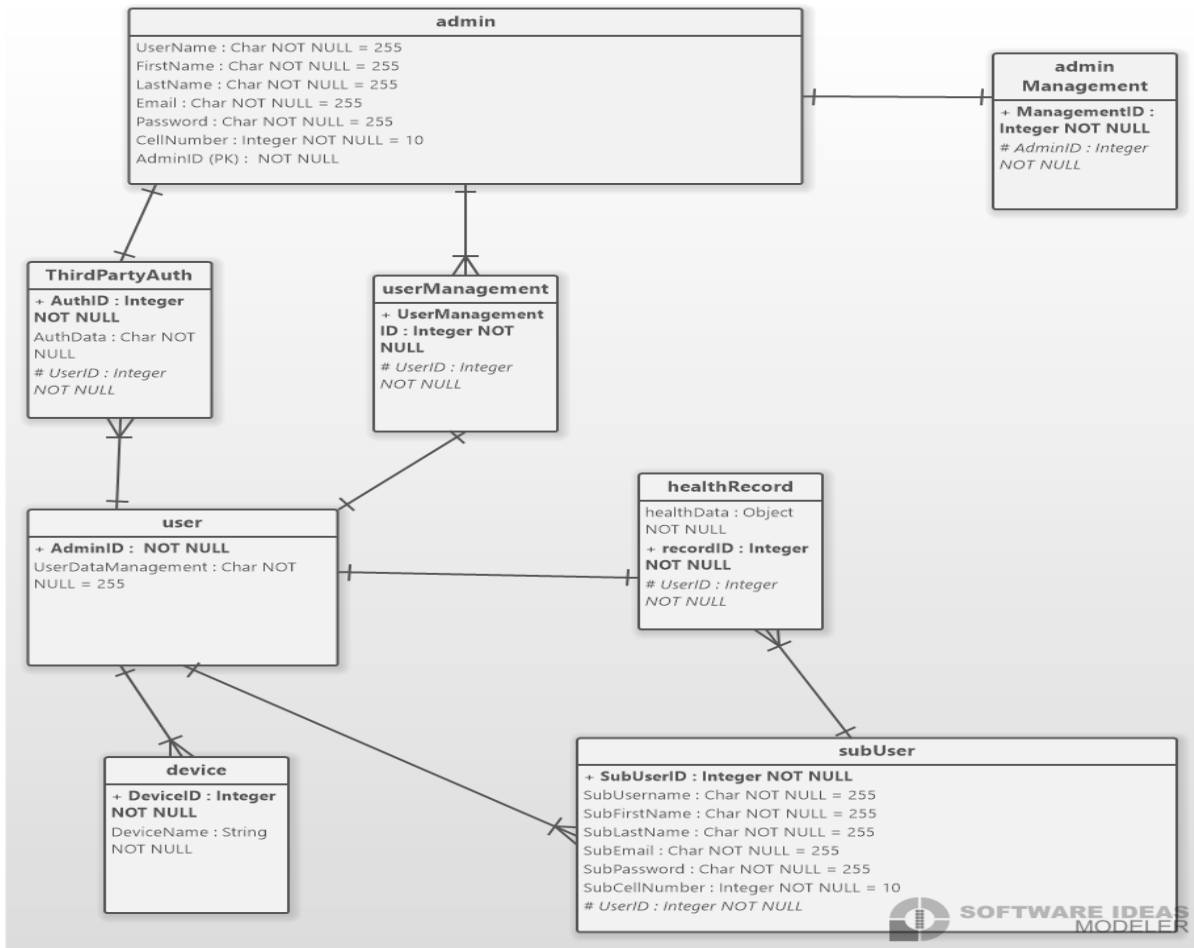
Print Menu

- Logged user information will be pulled and sorted with an addition of the document type, date displayed on the form, username, and phone number.
- Database will be used to ensure correct file formatting and printing.
- SQLite complies with the ACID agreement that ensures that print jobs are handled and processed reliably. With its high flexibility of different file formats to be transferred and printed data will be ready for output efficiently.

Share Menu

- Stored records can be manually selected based on the date and time range the user specifies, to be formatted for virtual sharing.
- Database security features will be applied to ensure the user has access over sharing preferences.
- SQLite maintains effective security measures and formatting to protect user information and allows information logs to be formatted and shared with the users' consent.

Entity Diagram



System Architecture and Patterns

Concerning Vita, the health log app, System architecture, and Design patterns are integral in transforming the concept of Vita into a functional application. System architecture defines the structural layout of the application, detailing how various components interact. Design patterns, on the other hand, provide standardized solutions to common design challenges, acting as a bridge between the conceptual design and the actual code. The system architecture of Vita can be broken down into three components, the user, the health log system, and a local database. The primary interaction is the users' ability to log/enter information into the health log system, and that system being able to store that information locally within itself. In saying that monolithic architecture is best for Vita; This is due to its simplicity, ease of deployment, and all components and applications are united into a single codebase. Regarding design patterns, a Model View Controller (MVC) and an observer pattern would be perfect for Vita. The Model will manage the data entered by the end user, such as heart rate, weight, and temperature, and handle the logic for storing, retrieving, and processing this data. The View will present this data via the user interface through graphs, input forms, and data displays. The Controller is the middleman; when changes update the View, the Controller will update the Model. In addition to the Controller, implementing the Observer pattern ensures that when the model is updated, the view is also updated. The MVC pattern ensures a clean separation of concerns, making Vita easier to maintain and update, while the Observer pattern guarantees that the user interface reflects real-time data changes, enhancing user interaction. In summary, the Vita health log app leverages a monolithic architecture for its simplicity and cohesive structure, ensuring all components are integrated within a single codebase for ease of management. The application employs the Model-View-Controller (MVC) pattern to efficiently separate data management, user interface, and control logic, enhancing maintainability. Additionally, the Observer pattern is utilized to ensure real-time updates between the user interface and data model, providing dynamic and responsive user interactions. Together, these architectural and design choices form the backbone of Vita, facilitating a robust and user-friendly health logging system.