Health.Log

Prepared for

Alex Schmidt

Prepared by

Nicholas Gonzalez, Valentine Jingwa, Tamara Mahmoud,

Jean-Pierre Nde-Forgwang, Jonah Scott

Software Development – Cohort A

School of Advance Digital Technology

SAIT

16 October 2023

Data Bank of Health Information (DBHI)	3
Sponsor:	3
Sponsors System:	3
Team's proposed solution:	3
Key Takeaway:	3
Use Case Diagram & Description:	4
Class Diagram:	8
Appendix:	12
Software Analysis Class Team Contract	12
Team Members:	12
General Team Rules:	12
Communication	12
Meetings	12
Work Distribution	12
Decision Making	12
Responsibilities:	12
Team Leader	13
Quality Assurance	13
Developer	13
Documentation Specialist	13
Timeline:	13
Missing Deadlines	13
Poor Quality Work	13
Lack of Communication	14

Data Bank of Health Information (DBHI)

With the current healthcare system, tracking data has become vital for making an accurate diagnostic. This project target audience includes new parents, the elderly, and authorized aids (nannies and home nurses). In retrospect, for doctors and nurses, this project will have relevant set data that they can easily understand and conclude potential diagnostics.

Sponsor:

- Jose Rojas
- Director of Software and Tech Services at Active Conversations

Sponsors System:

- The sponsor doesn't have a current system.
- No current system developed by other institutions provides enough functionality for the medical professionals.

Team's proposed solution:

The main problem space this app will solve is the time it takes for a health professional to glean information you've tracked and accumulated, thus finding a commonality that leads to a diagnosis. The main subset is a collection of vitals, medication, intake/output, medical history, and symptoms. It allows medical professionals to dissect such information and reduce wait times at health institutions.

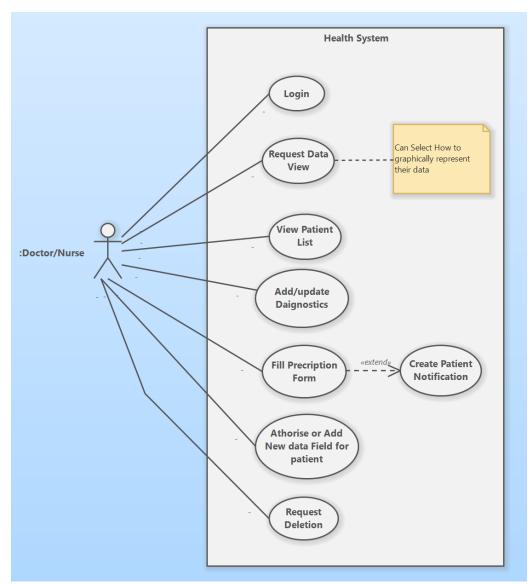
Key Takeaway:

- Create a mobile application that help user track their activities and converts some of the data into useful data used for diagnosis.
- The interface should be easy to use for parents, elderly and any adult.
- The app should be able to create graphs, charts or any pictorial representation of data.
- We need to engage the user into putting in the data just like social media does.
- Data should be easily transcribed to paper so user can have a hard copy i.e. pdf

Use Case Diagram & Description:

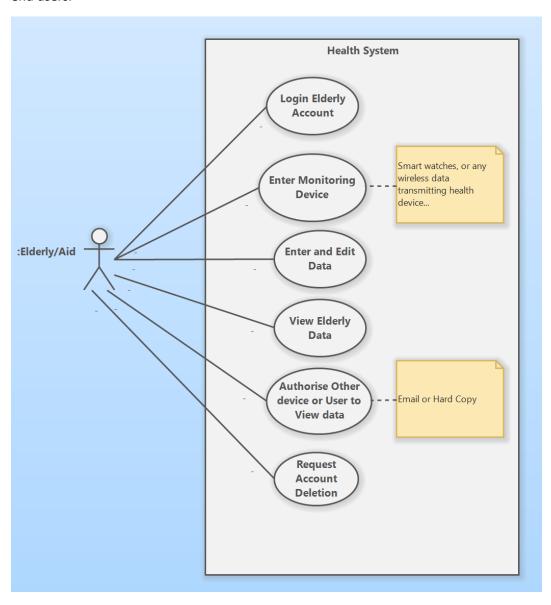
Title: Doctor/Nurse

Description: This use case represents the actor of Doctor and Nurse, or healthcare professionals in general. This case demonstrates Doctors and Nurses and the class functionality they have within the health system. The key takeaway is the ability of the actor to view and manipulate patient data.



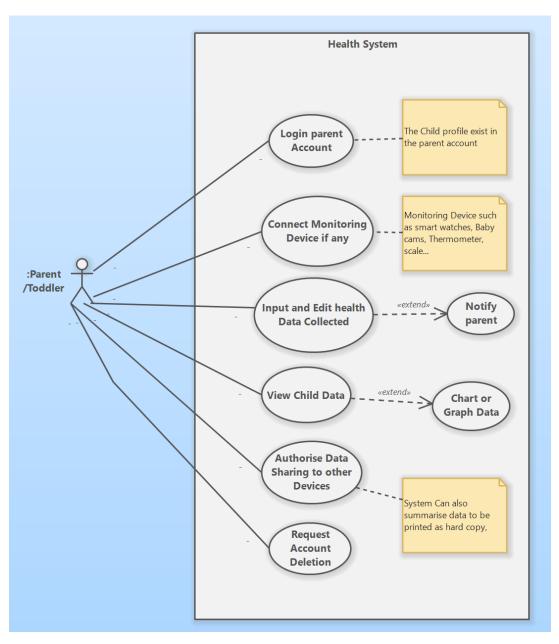
Title: Elderly/Aid

Description: This case represents the elderly (those over 65) and aids (those administering care). In terms of the system, key positionality includes the entry of data and authorization of viewing by other end users.



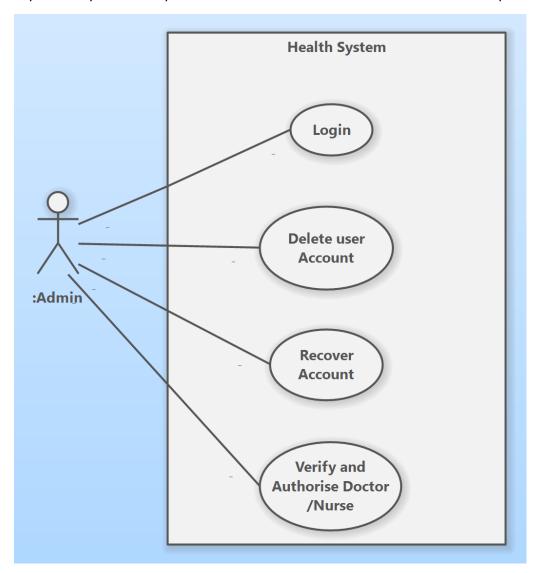
Title: Parent/Toddler

Description: This case represents the actor of Parent and Toddler, the key takeaway being the actors ability to enter data into the Health System. In addition to allowing authorization in terms of viewership for set data.



Title: Admin

Description: This case represents the administrative body behind the functionality of Health.Log. The key takeaway is the ability of this actor to control all accounts within the Health System and access data.



Class Diagram:





:Parent/Toddler

- parantName: String
- parantAge: Int
- parentContact: Int
- parentAddress: String
- toddlerName: String
- toddlerAge: Int
- toddlerBirthDate: Date
- toddlerID: Int
- + <<getter>> GetParentName (): String
- + <<setter>> SetParentName (): String
- + <<getter>> GetParentAge (): Int
- + <<setter>> SetParentAge (): Int
- + <<getter>> GetParentContact (): Int
- + <<setter>> SetParentContact (): Int
- + <<getter>> GetParentAddress (): String
- + <<setter>> SetParentAddress (): String
- + <<getter>> GetToddlerName (): String
- + <<setter>> SetToddlerName (): String
- + <<getter>> GetToddlerAge (): Int
- + <<setter>> SetToddlerAge (): Int
- + <<getter>> GetToddlerBirthDate (): Date
- + <<setter>> SetToddlerBirthDate (): Date
- + <<getter>> GetToddlerID (): Int
- + <<setter>> SetToddlerID (): Int
- + < <create>> :Parent/Toddler (parentName : string, parentAge : int, parentContact : int, parentAddress : string, toddlerName : string, toddlerAge : int, toddlerBirthDate : Date, toddlerID : int): Object

:HealthLog

- Vitals: String
- Medications: String
- Intake/Output: String
- MedicalHistory: String
- Symptoms: String
- Others: String
- + <<getter>> GetVitals (): String
- + <<setter>> SetVitals (): String
- + <<getter>> GetMedications (): String
- + <<setter>> SetMedications (): String
- + <<getter>> GetIntake/Output (): String
- + <<setter>> SetIntake/Output (): String + <<getter>> GetMedicalHistory (): String
- + <<setter>> SetMedicalHistory (): String
- + <<getter>> GetSymptoms (): String
- + <<setter>> SetSymptoms (): String
- + <<getter>> GetOthers (): String
- + «setter» SetOthers (): String
- + <<create>>: HealthLog (vitals: string, medications: string, intake/output: string, medicalHistory: string, symptoms: string, others: string): Object

:Adult/Aid

- adultName: String
- adultAge: Int
- adultContact: Int
- adultAddress: String
- adultID: Int
- aidName: String
- aidContact: Int
- aidRelationship: String
- hasLivingWill: Boolean
- hasDurablePowerOfAttorney: Boolean
- <<getter>> GetAdultName (): String
- <<setter>> SetAdultName (): String
- + <<getter>> GetAdultAge (): Int
- + <<setter>> SetAdultAge (): Int
- + <<getter>> GetAdultContact (): Int
- + <<setter>> SetAdultContact (): Int
- + <<getter>> GetAdultAddress (): String
- + <<setter>> SetAdultAddress (): String
- + <<getter>> GetAdultID (): Int
- + <<setter>> SetAdultID (): Int
- + <<getter>> GetAidName (): String
- + <<setter>> SetAidName (): String
- + <<setter>> SetAldivalle (). Stillig
- + <<getter>> GetAidContact (): Int
- + <<setter>> SetAidContact (): Int + <<getter>> GetAidRelationship (): String
- + <<setter>> SetAidRelationship (): String
- + <<getter>> GetHasLivingWill (): Boolean
- + <<setter>> SetHasLivingWill (): Boolean
- + <<getter>> GetHasDurablePowerOfAttorney (): Boolean
- + <<setter>> SetHasDurablePowerOfAttorney (): Boolean
- + <<create>> :Adult/Aid (adultName : string, adultAge : int, adult Contact : int, adultAddress : string, adultID : int, aidName : string, aidContact : int, aidRelationship : string, hasLivingWill : boolean,

hasDurablePowerOfAttorney : boolean): Object

:App

- userProfile: String
- userHealthLog: String
- calenderData: String
- + <<getter>> GetUserProfile (): String
- + <<setter>> SetUserProfile (): String
- + <<getter>> GetUserHealthLog (): String
- + <<setter>> SetUserHealthLog (): String
- + <<getter>> GetCalenderData (): String
- + <<setter>> SetCalenderData (): String
- <<create>> :App (userName : string, userHealthLog : string, calenderData : string): Object

:Doctor/Nurse

- name: String
- phoneNumber: Int
- email: String
- specialization: String
- licenseNumber: Int
- hospitalAffiliation: String
- patientList: List
- + <<getter>> GetName (): String
- + <<setter>> SetName (): String
- + <<getter>> GetPhoneNumber (): Int
- + <<setter>> SetPhoneNumber (): Int
- + <<getter>> GetEmail (): String
- + <<setter>> SetEmail (): String
- + <<getter>> GetSpecialization (): String
- + <<setter>> SetSpecialization (): String
- + <<getter>> GetLicenseNumber (): Int
- + <<setter>> SetLicenseNumber (): Int
- + <<getter>> GetHospitalAffiliation (): String
- + <<setter>> SetHospitalAffiliation (): String
- + <<getter>> GetPatientList (): List
- + <<setter>> SetPatientList (): List
- + <<create>> :Doctor/Nurse (name : string, phone Number : int, email : string, specialization : string, license Number : int, hospitalAffiliation : string, patientList : list): Object

:Admin

- adminUserName: String
- adminPassword: String
- adminPermission : Boolean
- userList: List
- + <<getter>> GetAdminUserName (): String
- + <<setter>> SetAdminUserName (): String
- + <<getter>> GetAdminPassword (): String
- + <<setter>> SetAdminPassword (): String
- + << getter>> GetAdminPermission (): Boolean
- + <<setter>> SetAdminPermission (): Boolean
- + <<getter>> GetUserList (): List (List)
- + <<setter>> SetUserList (): List (List)
- + <<create>> :Admin (adminUserName : string, admin

Password : string, adminPermission : boolean, userList : list): Object

Appendix:

Software Analysis Class Team Contract

Team Members:

- Nicholas Gonzalez
- -Valentine Jingwa
- -Tamara Mahmoud
- Jean-Pierre Nde-Forgwang
- Jonah Scott

Purpose: This contract serves to outline the expectations, responsibilities, and guidelines that govern our team's behavior and actions for the duration of the Software Analysis course. All team members agree to adhere to the terms stated herein.

General Team Rules:

Communication

- Team members will maintain open, honest, and respectful communication.
- All team members should be reachable via email or team chat within 24 hours.

Meetings

- Team meetings will occur at least once a week.
- Attendance at all team meetings is mandatory unless an excuse is provided 24 hours in advance.

Work Distribution

- Work will be divided equitably among all team members.
- Team members agree to complete their assigned tasks by the agreed-upon deadlines.

Decision Making

- Decisions will be made collectively, with each member having one vote.
- In the case of a tie, the team leader will make the final decision.

Responsibilities:

Team Leader

- Oversee project milestones and timelines.
- Facilitate team meetings.

Quality Assurance

Ensure that the project meets the required standards and guidelines.

Developer

Write, test, and debug code as per project needs.

Documentation Specialist

Ensure that all project documentation is up-to-date and accurate.

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: Slack
- Project Management: Trello

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