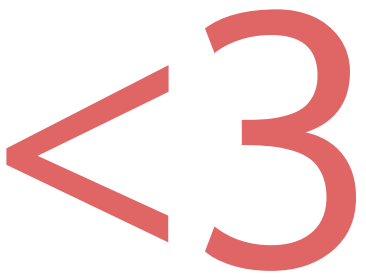


Heartrate Behavior and Analysis Tool (HBAT)

Ruhana Azam, Manoj Polisetti, Rajith Weerasinghe, Phillip Thain

Team 3

Table of Contents

Purpose

One tool autism researchers use to detect autism is to analyze heartbeat data against behavioral data. However, there is currently no intuitive and easy-to-use software available to do this. Current solutions such as Matlab or SAS require computer or programming outside the scope of knowledge of many researchers.

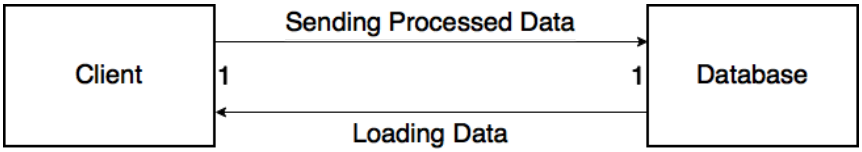
Our project will implement algorithms in a desktop application that will compare heart rate data with other data to provide a heuristic to assist in the diagnosis of infants who are considered at high-risk for autism.

The application will allow for the users to input physiological and behavioral data points. It then processes, analyzes, and exports the data for researchers to use in the future. In addition, the program will be capable of creating visual representations of the output data, which the researchers could use to draw their own conclusions.

Design Outline

High Level Overview

HBAT is a standalone application which will be able to analyze and visualize heartbeat data that the user inputs. Thus, the client itself will be responsible for majority of the functionality of the software. In addition, a database will be implemented, so that the user does not have to input data multiple times to view the same analyzed data.
To summarize, the two primary, high level, components of the application is the client and the database.



A. Client
    a. Interacts with the user:
        i. Allows the user to import and export CSV files
        ii. Displays a list of past data files
        iii. Allows user to open any past data files
    b. Parses CSV files
    c. Processes data that the user inputted
    d. Visualizes analyzed data into a line graph and data table
B. Database
    a. Stores all analyzed data and subject's profile information

Design Issues

Functional Issues:

1. **What details are going to be included in the "start page"?**
   Option 1 : Enter new patient data
   Option 2 : Load previous data
   Option 3 : Recent patients
   Option 4 : Edit previous patient data

   Decision: We have decided to implement the first three Options on the "start page", and leave out the 4th option. The goal of a start page is to create an intuitive and simple entry way into the program. In order to keep it simplistic, the menu will guide the user to either a new or old trial. Then, that trial may or may not be edited later on.

2. **What platforms will the program be able to run on?**
   Option 1: Windows
   Option 2: Mac
   Option 3: Unix
   Option 4: Browser (as web application)

   Decision - We have decided to implement option 1 and 2. Researchers commonly use both Windows and Mac computers to do their research, so it is important to support both to reach a wide audience, hence we are implement a Java application which runs on both Macintosh and Windows computers. The decision to not create a web application was primarily because of HIPAA, a law that ensures that patient information remains between them and their doctors.

3. **How should the application handle inputting multiple files at one time?**
   Option 1 : Only allow the user to input a single file at a time.

Option 2 : The user will be given the option to add physiological and behavioral data for a single data point. On the same window, the user will be given the option to add more files. (Look at UI mockup for example)
Option 3 : The user will be able to add a group of files via Control+click. The behavioral and physiological data will be linked via file name.

Decision - We have decided with Option 2 for the mean time but are considering Option 3 depending. Option 3 would allow the user to add groups of files easier but would expect the user to correctly name the files. We will speak to Professor Tonnsen for to decide which option would be more realistic for the user.

4. **When the the visualized graph and table GUI window is closed, should the program completely exit?**

Option 1: Yes. It would be the most intuitive way for the user to close all the program.
Option 2: No, the main menu should pop up again.

Decision - We decided to implement Option 2 as the software has a start up time and if the user accidentally closed the program it would take a time to reload the data. If the user wants to close the program completely, they would need to close the graphical window and the main menu.

Non - Functional Issues:

1. **How to store our data in the database?**

Option 1 : Save all the CSV files we are exporting
Option 2 : Save the data in a custom extension format

Decision - We have decided to go for Option 2, which enables us to work to make our own extension which would make loading data faster instead of processing

the CSV file every single time. This would also enable us to look into extension formats, etc

2. **How do we want to visualize our data and what we want to implement it using?**
   Option 1 : Python's Visualization Libraries
   Option 2 : Open Source Visualization softwares from GitHub

   Decision - We decided to implement Option 1, Python's Visualization libraries, as we can easily export the graph in an SVG format to display it to the user and also because it's easier to implement.

3. **What GUI classes are we planning to implement for our application?**
   Option 1 : Java Swing along with WebLaF
   Option 2 : Java FX
   Option 3 : External Open Source GUI libraries

   Decision - We have yet to decide on which GUI class to implement in our application as right now we are still evaluating and looking at the pro's and con's of each of these options.

4. **What languages we can use for processing data in conjunction with Java?**
   Option 1 : Java and Python
   Option 2 : Java along with Clojure
   Option 3 : Java along with Scala

   Decision - We are currently choosing Option 1, which is using Python for scripting and data processing work but we are also considering Clojure for its prowess in processing large amount of data and we haven't decided for sure yet.

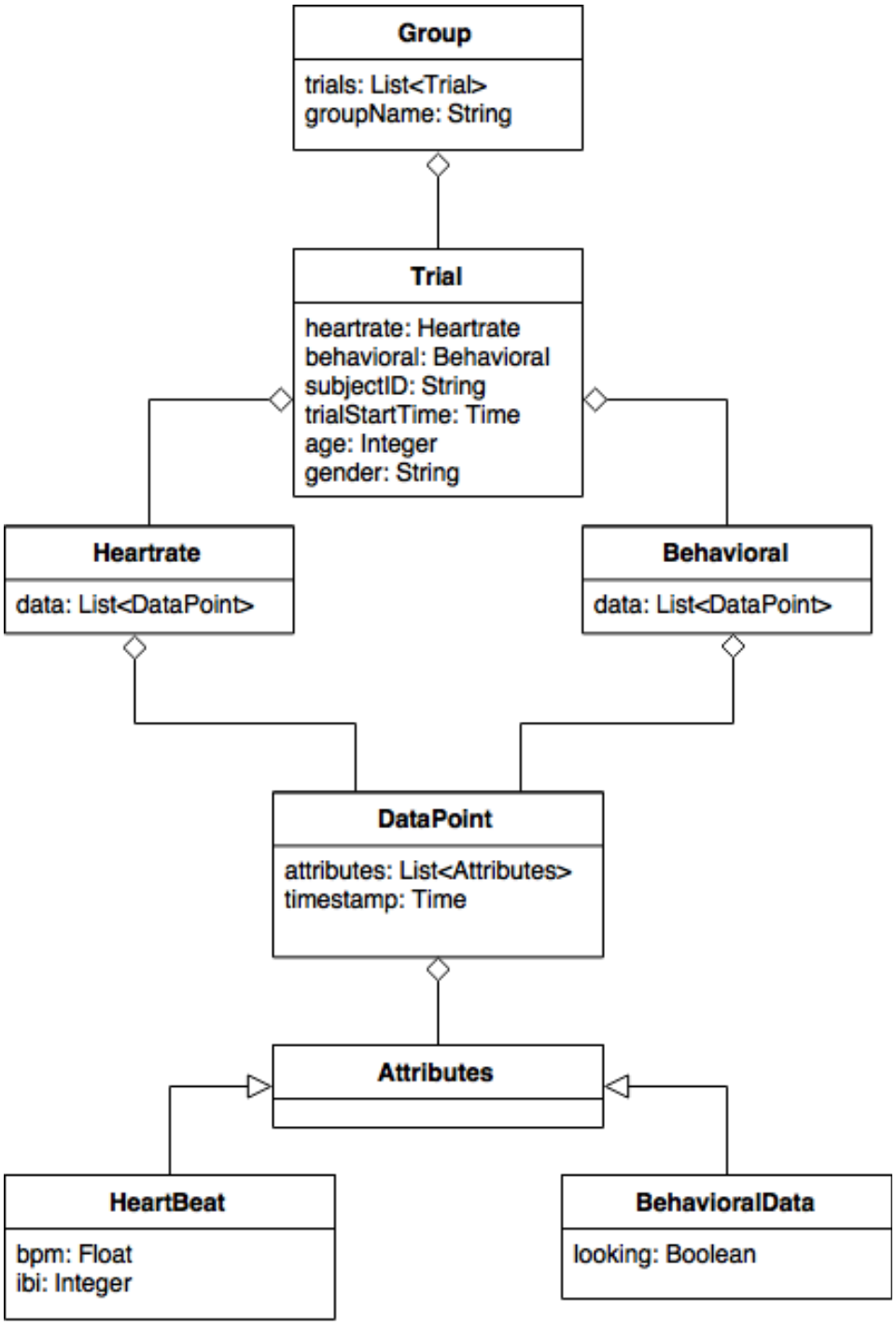
5. **How are we going to set up our database?**
   Option 1 : Cloud Database

Option 2 : Local Database
Option 3 : Web Hosting

Decision - We chose Option 2, because having the medical data of the subjects on a cloud service would mean that we have to follow strict protocols and also having a local database was asked for by our stakeholder as it would mean the medical data is safer.

Design Details

Class Diagrams:



Group: This class allows the user to group trials. This way the user can group the data sets by experiment or other attributes.

Trial: This class holds information on a single trial of a subject. Profile information of the subject will be stored in this class as well as data of the behavioral and heartrate data sets. The timestamp will indicate when the data was collected for that trial.

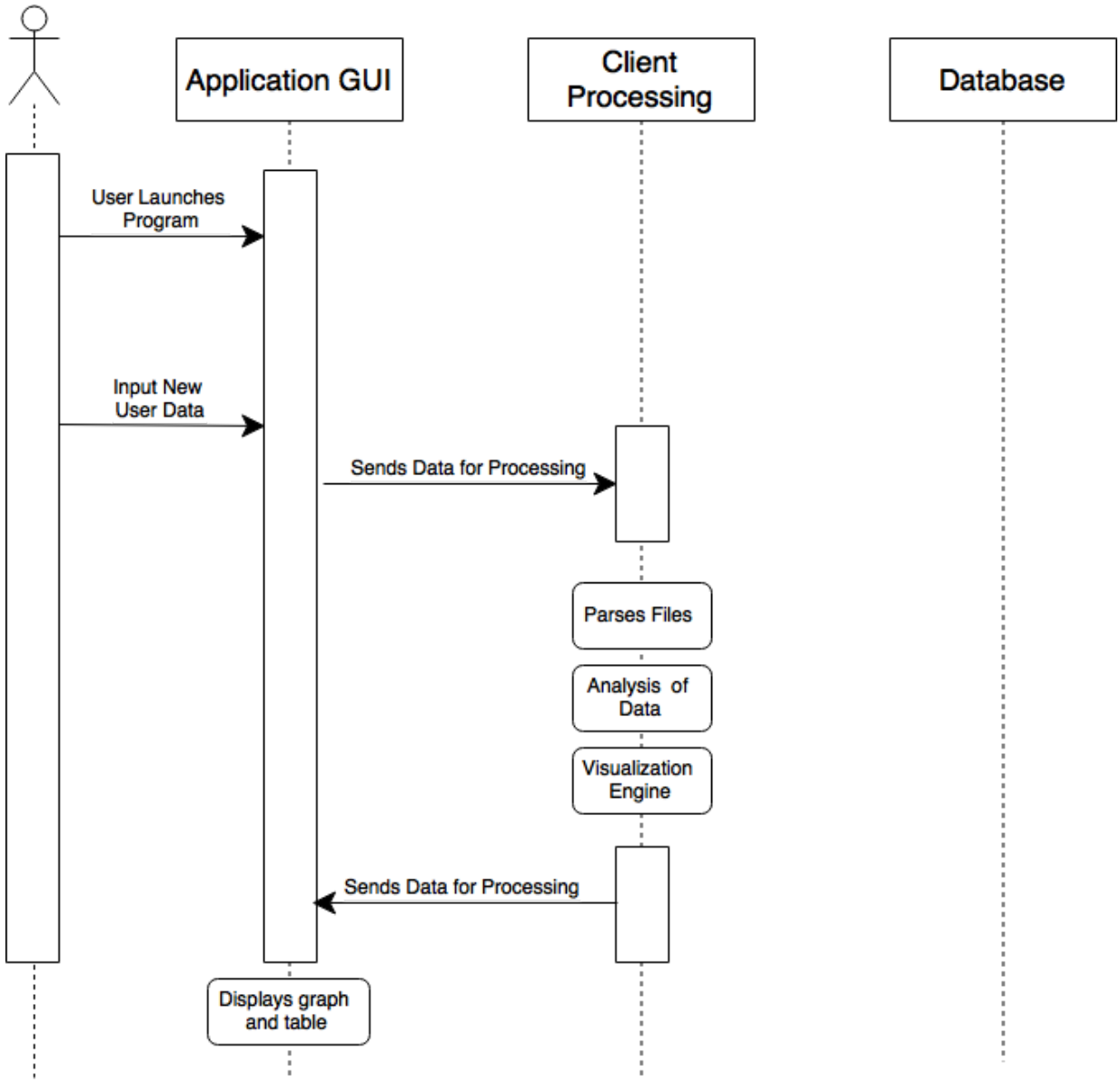
Heartrate: This class will contain all general heart rate information pertaining to that specific trail. This includes a list of data heartrate points and any additional summary information.

Behavioral: Similar to the heartrate class, the Behavioral class will hold all general behavioral data. This includes a list of all behavioral data points and any other general behavioral summary information.

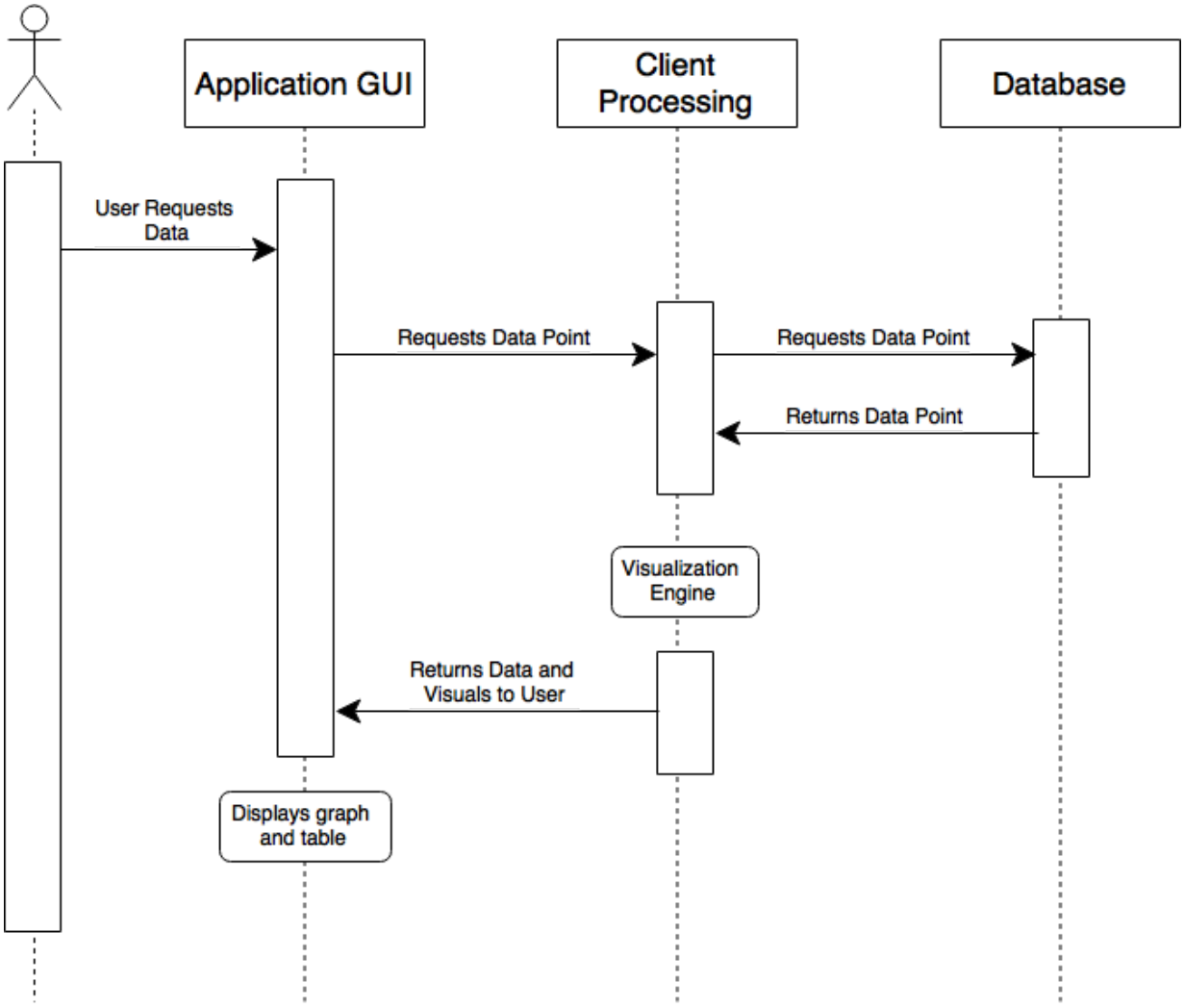
Datapoint: This is a parent class for all data points.

1. BehavioralData: This is a parent class of data points specific for behavioral data. The attribute “looking” indicated whether the child was looking at a stimulus. Other behavioral attributes specific to the datapoint can be defined here.
2. HeartbeatData: Similar to the BehavioralData class, HeartbeatData is a child of Datapoint. This class will contain data heartbeat data specific to it’s datapoint. This includes, but is not limited to, bpm (beats per minute) and IBI (interbeat interval).

Sequence Diagrams



When adding the trial, the user selects the “Add Trial” option which leads the user to a form to upload the heart rate and behavioral data files. Then, the Client processor parses the data, sends it to the analyzing program and then directs it to the visualizer. The visualizer creates the necessary graphs in the GUI that are visible to the user.



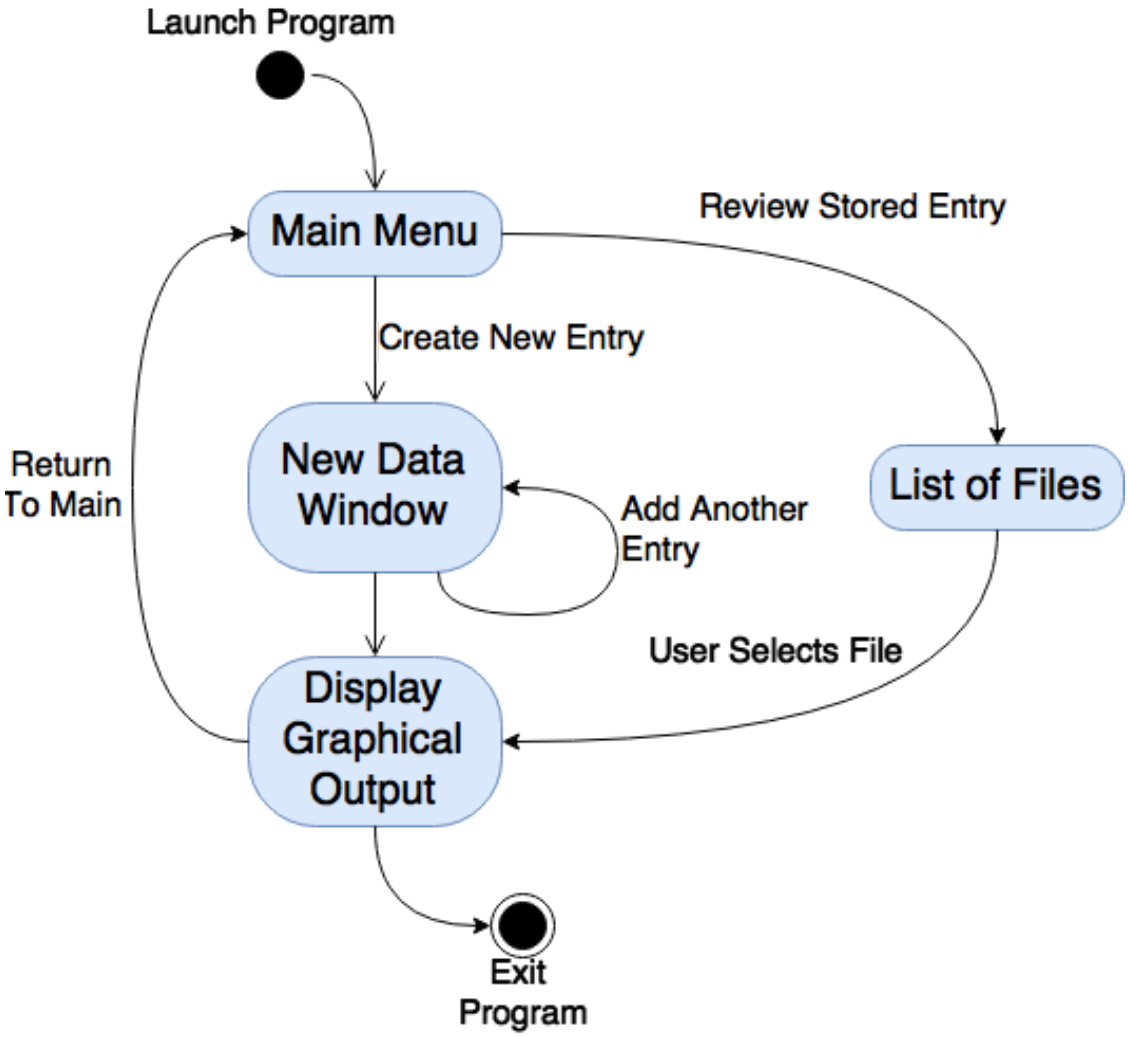
When loading a trial, the user selects the “Load Trial” option which directs the user to the database where they can select the file they want to load. Then, the file is sent to the visualization program which outputs previously processed graphs in the GUI to the user.

- The Parser parses the inputted CSV files into a format which the Analyzer class can apply the algorithm on.
- The Analyzer class applies the necessary data processing and algorithms to the data and outputs the file to the Visualizer class.
- The Visualizer class takes the analyzed data and works with it using Python’s visualization libraries to create graphs in an SVG format which is outputted to the GUI to display to the user.

Activity Diagram

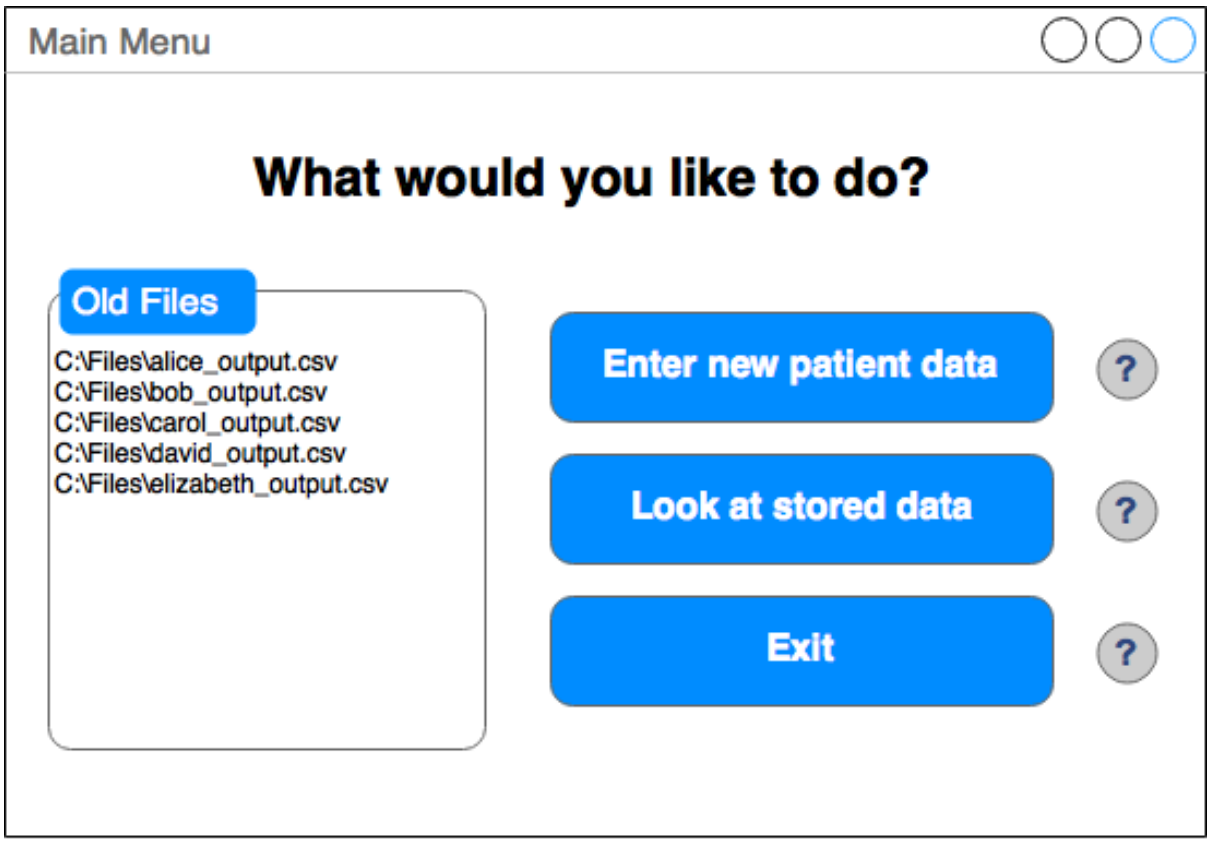
Our activity diagram describes the flow of the program. Each state corresponds to a window and has a specific set of behaviors and actions.

In addition, each window will have a corresponding mockup of the user interface. The appearance is intended to be agnostic to any particular library or framework, and only show what components will appear on each page and in what arrangement.



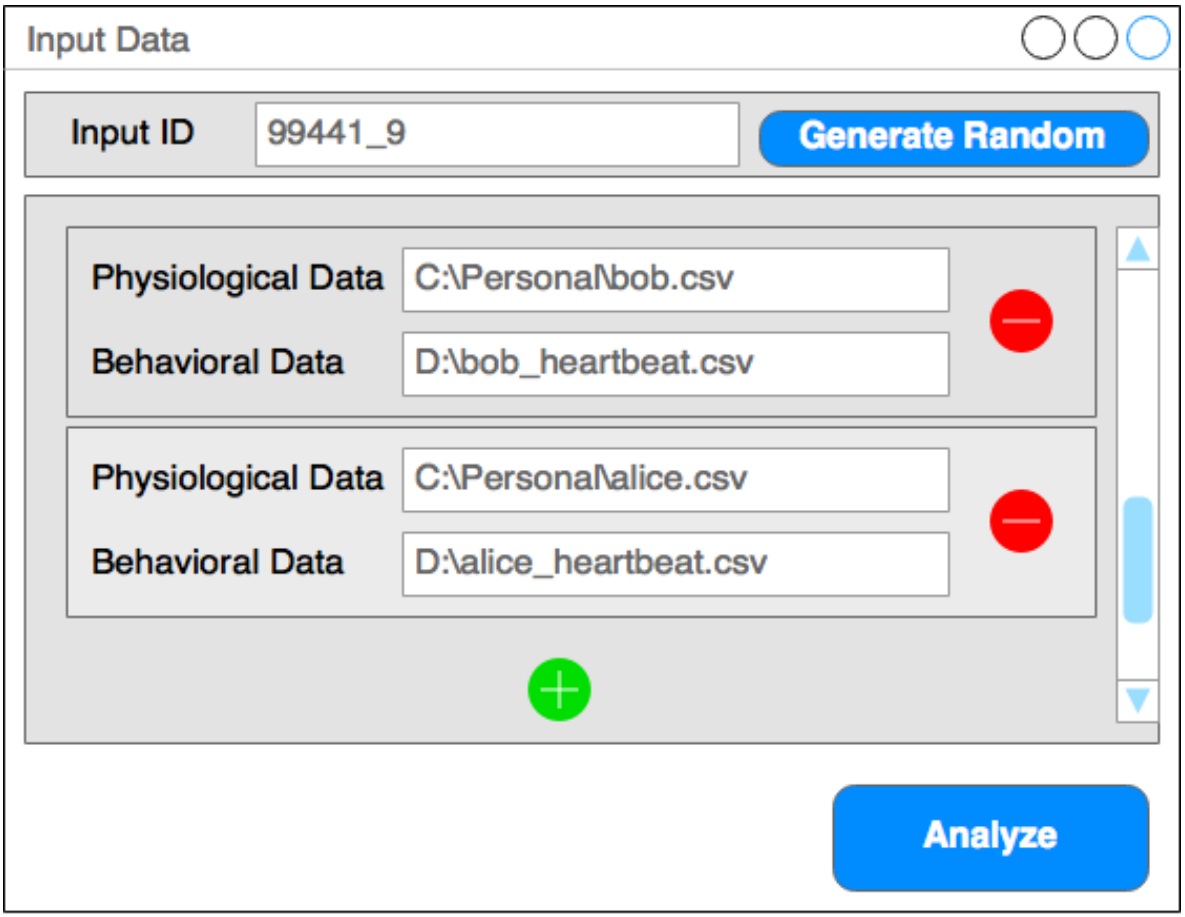
Main Menu

A. The main menu will be the first window that pops up when the software is opened. The user will be provided with the following options:
    a. Input new data files into the system
    b. Browse old data files that have previously been analyzed.
    c. The five most recently viewed files will come up as options for user to easily access without having to browse through all previous files.
    d. Exit the program
B. The program will exit if this window is closed.



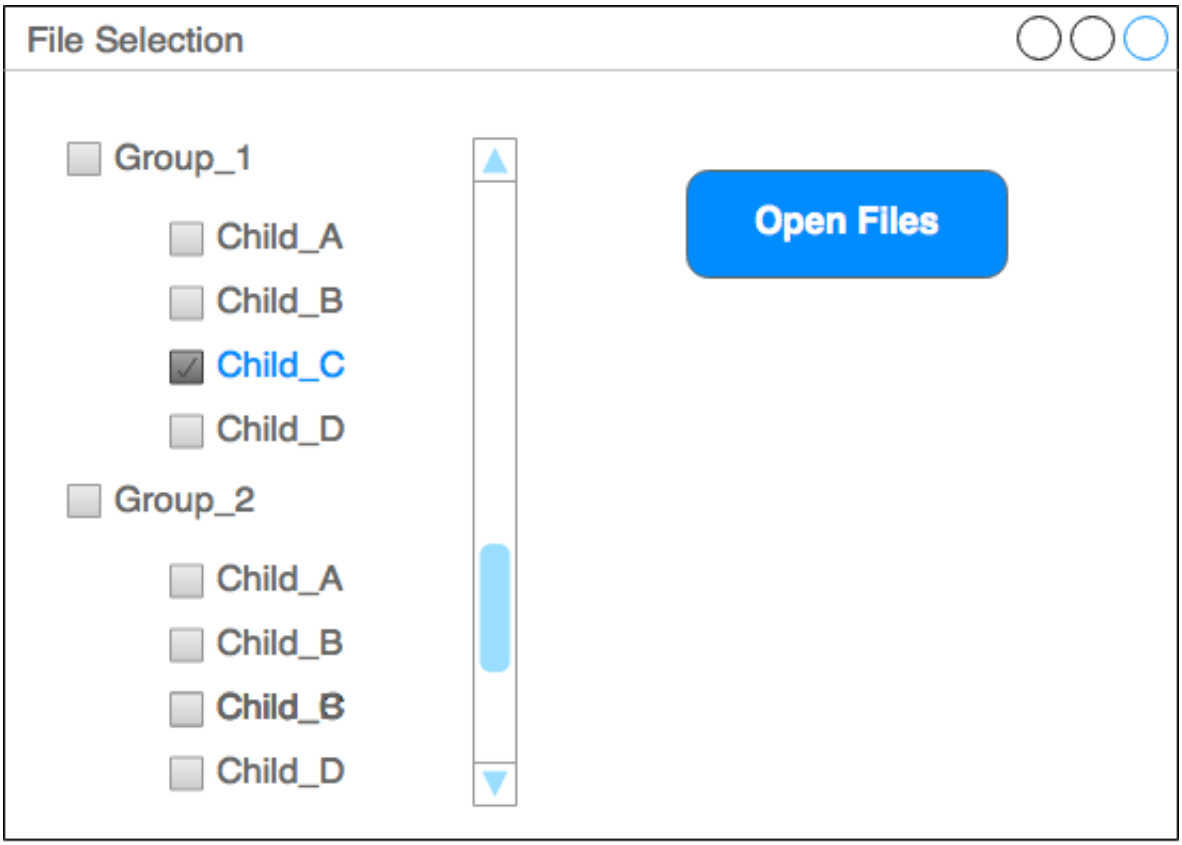
New Data Window

A. If the user chooses to input new data, this window will come up. This window will give the user to input new sets of data into the system to be analyzed. Each set of data will have: an ID, behavioral data file and physiological data file.
B. After this data is entered, the user will be directed to the Graphical Output Window
C. If this window is closed, it will return to the main menu.



List of Files

A. If the user choices to view old set of data this window will pop up. This window will let the user browse through previously analyzed data. All previous data will be batched by Experiment ID. Each file and experiment will have a checkbox next to them. The user will check each of the files he/she wants to view.
B. After all the files they want to view is checked, the user will be directed to Graphical Display Window
C. If this window is closed, it will return to the main menu.



Display Graphical Output

A. This window will include a line graph and data table for each data set the user wanted to view. It may also highlight certain portions of the graph that are of significance.
B. If the user chose to view several data sets, there will be tabs at the top of the window for each data set. This will allow the user to quickly switch between data sets.
C. There will be a button at the bottom of the page called “Add Data Set”. “Add Data Set” will allow the user to add another tab to set of data the is already being viewed. To do this, the List of Files Window will pop up again and the user will be able to check any additional data sets they would like to view.
D. When this window is exited, it will return to the main menu.