CS2013: Software Design Specification Outline

1.Introduction

1.1 Overview – purpose of systems

The purpose of this group project for CS2013/CS3013 is to design and create machine learning models for tracking gross-motor skills for our client Benoit Bossavit, a post-graduate researcher in the SCSS department. Our client is doing research on the development of gross-motor skills in young children and has asked that we create a machine learning model to help track certain movements to be able to determine whether the children can perform the motor-skills, and hence the overall purpose of our project is to create machine learning models that can recognize such motor-skills.

This project will help our client spend more time collecting data for his research rather than spending time developing the machine learning aspects himself.

1.2. Scope

This project consists of developing an efficient and accurate machine learning model that can identify what movement is being performed when a json file containing motion capture data is passed through the model.

The machine learning model will be presented with a motion capture data file and will determine which motor-skill it is relating to. It will look to determine how many times the skill was performed for each motion capture data file. In order for a motion to be considered 'successful' the machine learning model determines if it was performed sufficiently, i.e. if the motor-skill being checked for is a hop and the child jumped off one foot but landed on two, this is not considered a successful iteration of the motor-skill.

1.4. Definitions, abbreviations

Deep learning

Deep learning is an artificial intelligence function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence (AI) that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network.

Gross-Motor Skills

Gross motor (physical) skills are those which require whole body movement and which involve the large (core stabilising) muscles of the body to perform everyday functions, such as standing and walking, running and jumping, and sitting upright at the table.

Machine learning

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly

Neural networks

Neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. They interpret sensory data through a kind of machine perception, labeling or clustering raw input. The patterns they recognize are numerical, contained in vectors, into which all real-world data, be it images, sound, text or time series, must be translated.

1.5. References

Deep Learning

https://www.investopedia.com/terms/d/deep-learning.asp

Gross-motor Skills

https://childdevelopment.com.au/areas-of-concern/gross-motor-skills/

Machine Learning

https://expertsystem.com/machine-learning-definition/

Neural Networks

https://pathmind.com/wiki/neural-network

2. System Design

2.1. Design Overview

2.1.1 High-level overview of how the system is implemented, what tools, frameworks and languages are used etc.

The tools and technologies we have chosen are based on what we believe to be most suitable with some advice from our demonstrator. The main programming language we used for our project is Python as well as different libraries for the front-end and back-end.

The front-end will be able to load multiple motion capture files at once to be read by the machine learning model and be able to output multiple results and will be able to export results of the processed data into one of several file formats (e.g. .XLS, .CSV, .XML, .JSON, etc.)
.

We are going to be using the following tools to construct our front end.

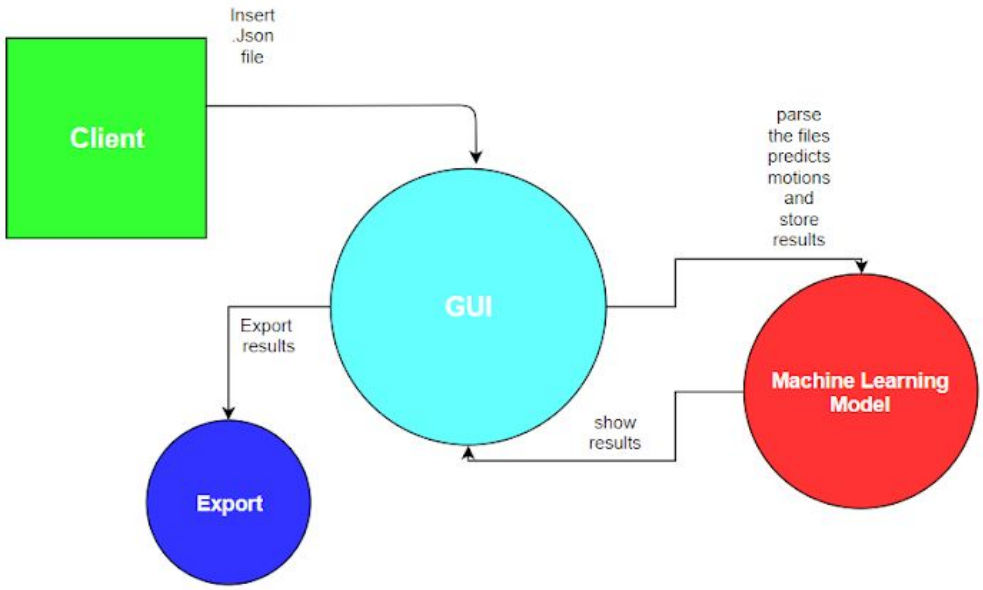
- PyQt: is a Python binding of the cross-platform GUI toolkit Qt, implemented as a Python plug-in.
- Tkinter : is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications

The backend will take in files selected in the selection system and parse the .json files into a readable data format. The system will have a pre-baked machine learning model that predicts motions at approximately 90% to 95% accuracy. After the .json files have been parsed, they will be sent into the pre-baked model, and will output a list of the motions that have been inputted to the frontend, followed by which motion they are most likely to represent.

We are going to be using the following tool to construct our back end.

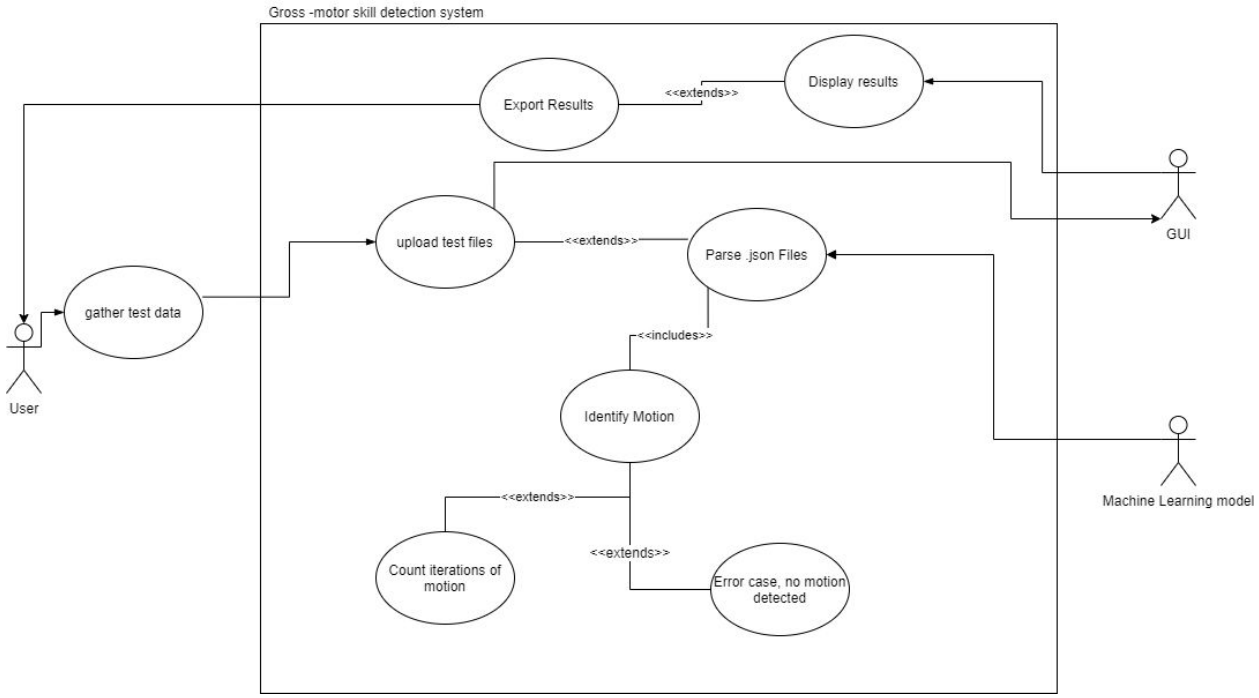
- TensorFlow: This is an open source library used for numerical computation and large-scale machine learning. Our machine learning model will be able to accurately predict the iterations of each motion from the .json motion capture file (e.g. if the motion is determined to be a Hop-Left, the system will be able to predict how many times a Hop-Left was done in the motion capture.). The results will be stored in a file document.

2.2.1. System Context



This context diagram outlines how each of the product's aspects interact as a system. The Gui interacts with all of the other components. The user is able to upload/insert .json files to the GUI which then passes them to the machine learning model which parses the files and performs the necessary analysis. The results are returned to the GUI and can then be exported

2.2.2 Use Case Diagram (from Requirements) ju

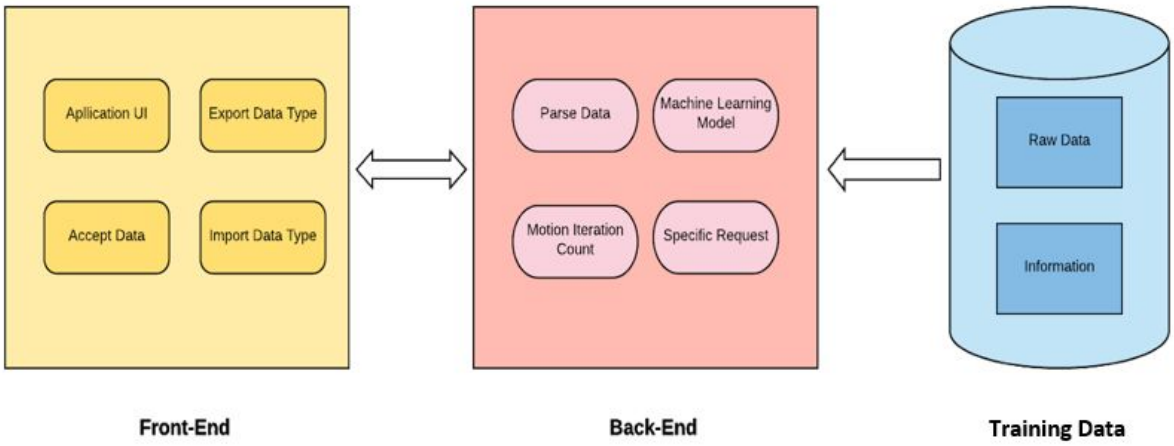


Textual Descriptions

Name: Upload test files Participating actors: User, GUI Entry condition: the user has test files to upload Exit condition: the files are uploaded to the GUI Normal scenario: the user uploads the files they want to process Error scenario: the user tries to upload non json files, told the file type is incompatible.	**Name: Gather Test Data(outside system)** Participating actors: User Entry condition: User has test subject and access to kinect. Exit condition: Data on coordinates frames is collected as a json Normal scenario: The test subjects movements are captured and mapped by the kinect and stored as a json file of coordinate frames Error scenario: The test subjects movements could not be mapped or recorded

Name: Parse .JSON Files Participating actors: Machine Learning Models Entry condition: files are uploaded via the GUI Exit condition: The data is parsed into a processable format Normal scenario: The files uploaded are parsed to be passed to machine learning model Error scenario: the files are unable to be parsed	**Name: Export Result** Participating actors: GUI, User Entry condition: The files were successfully processed and results were produced Exit condition: The user has an exported copy of the results Normal scenario: The user clicks “export” and the results are exported as a file Error scenario: The export fails, user told to retry
Name: Identify Motion Participating actors: Machine Learning Models Entry condition: Files have been parsed and user clicks ‘analyse’ Exit condition: The machine learning model identifies the motion in each file Normal scenario: The user clicks the ‘analyse’ and the testing data is passed to the learning model to identify each files motion Error scenario: motion could not be identified	**Name: Display Result** Participating actors: GUI Entry condition: The files were successfully processed and results were produced Exit condition: The outcome displayed on the screen and written into a file Normal scenario: The machine learning model produced results in a file and are displayed to the User on the GUI Error scenario: The machine learning model could not produce results.
Name: Count iterations of motion Participating actors: machine learning model Entry condition: the motion has been identified Exit condition: the number of times the motion was completed in each files is recorded Normal scenario: the model counts how many times in each file, the relevant motion is fully completed Error scenario: The model is unable to detect if the motion is not being performed properly	

2.2.3 System Architecture



This architecture diagram details the interaction between the different parts of the application.

The front end has the functionality to :

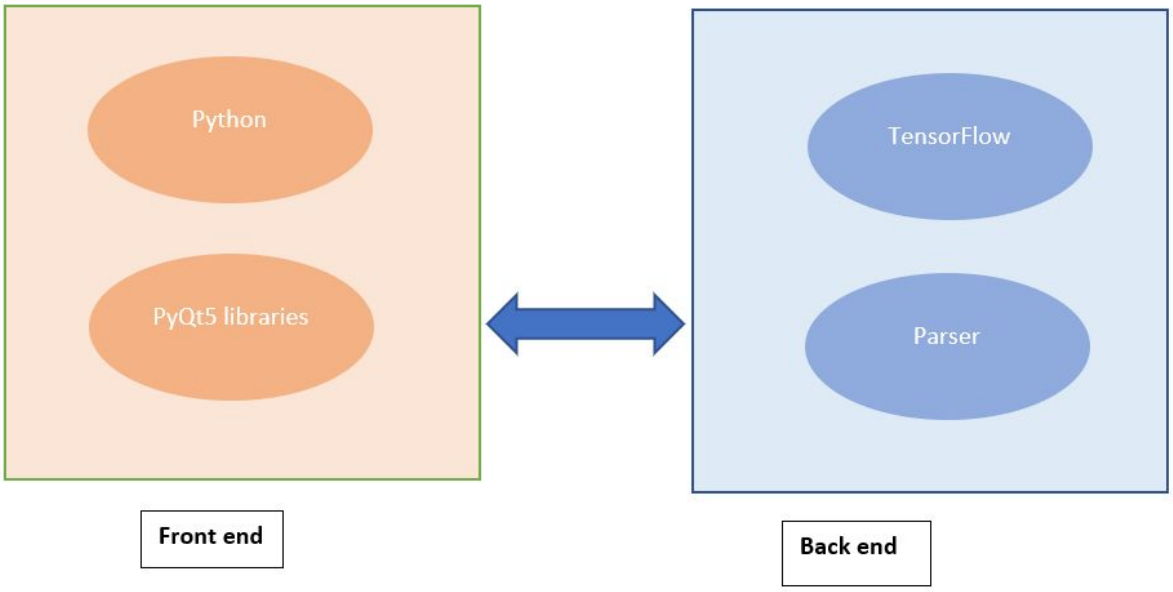
- import data
- export data
- accept data from the back end
- act as a user interface application where the user can drag and drop or manually select files to upload.

The back end has the functionality to

- parse data into a format that can be analysed
- count motion iterations
- use machine learning models to identify the motion
- make specific requests for certain motions for example if the motion is jump, it can request to compute the distance.

The training data is fed to the back end to train the machine learning models:

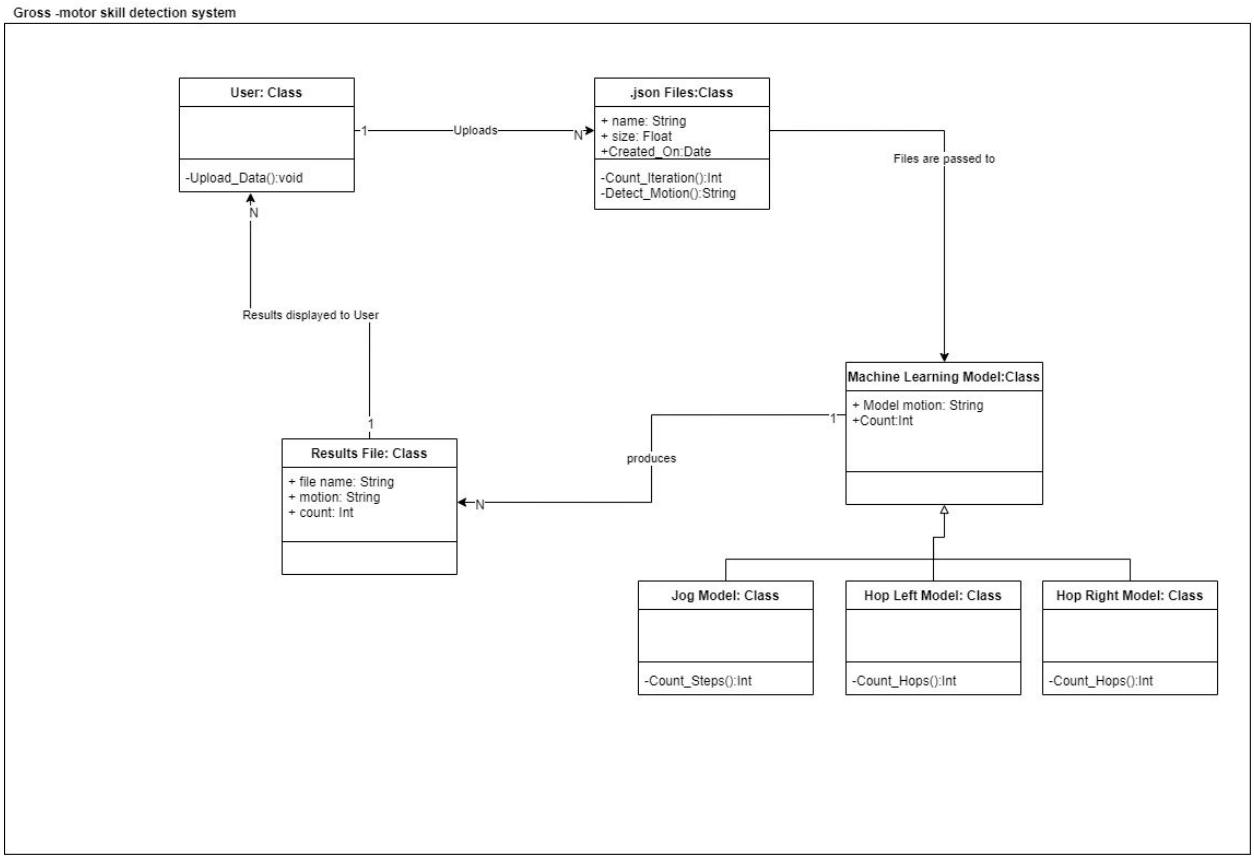
- We got the raw data from our client, which are the data frames about the motions.
- Create some new training data based on the original data by shifting the coordinates.
- We get the guide line from our client, demonstrator and mostly doing online research.



Above is a very simple depiction of the main technologies used in each aspect of the application.

- The front end GUI is made using Python and uses PyQt5 libraries to add functionality
- The back end machine learning models uses tensorFlow and a parser to allow the data to be converted into an analysable format.

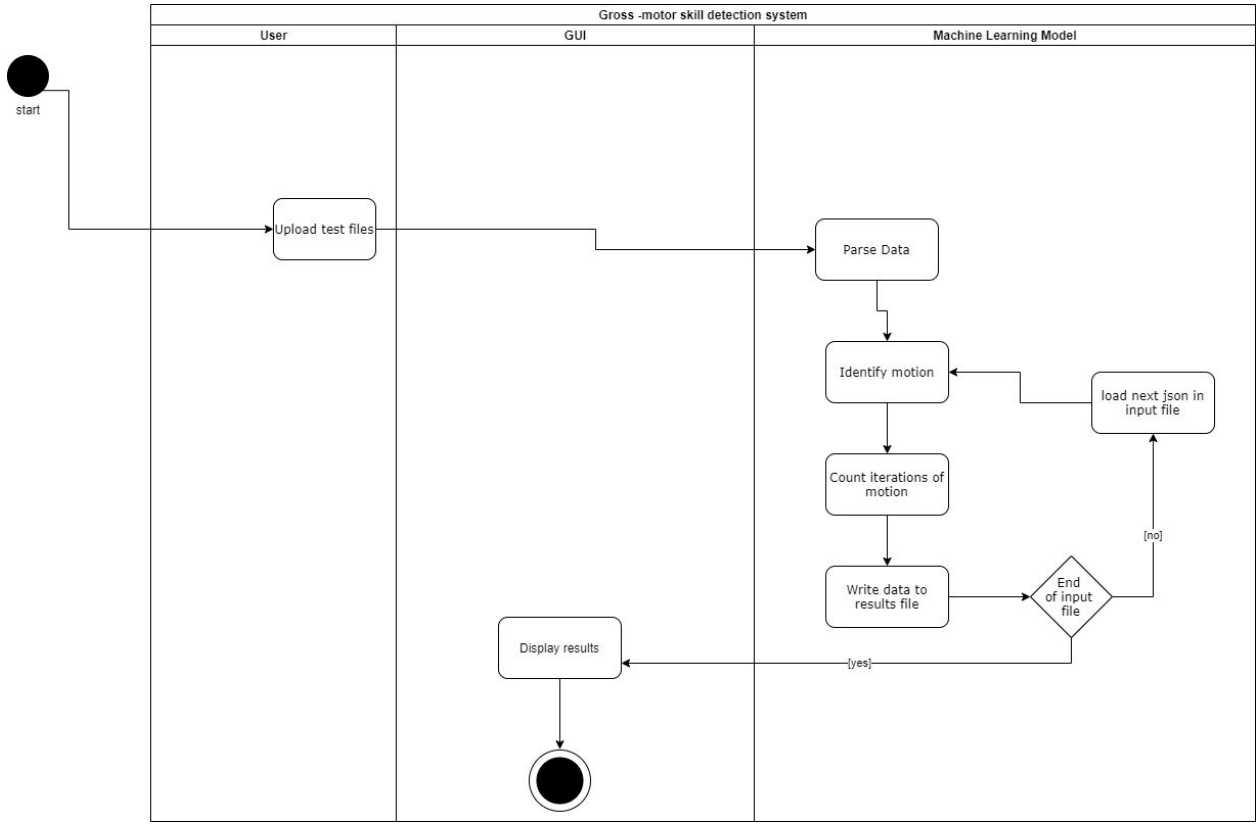
2.2.4 Class Diagram



The class diagram above shows how we expect the elements of our system to interact

- The user uploads multiple .json files
- These files are then passed to the machine learning model
- There are 3 machine learning models; jog, hop left and hop right (indicated by inheritance)
- The machine learning model then produces multiple results files
- The results are then displayed to the user

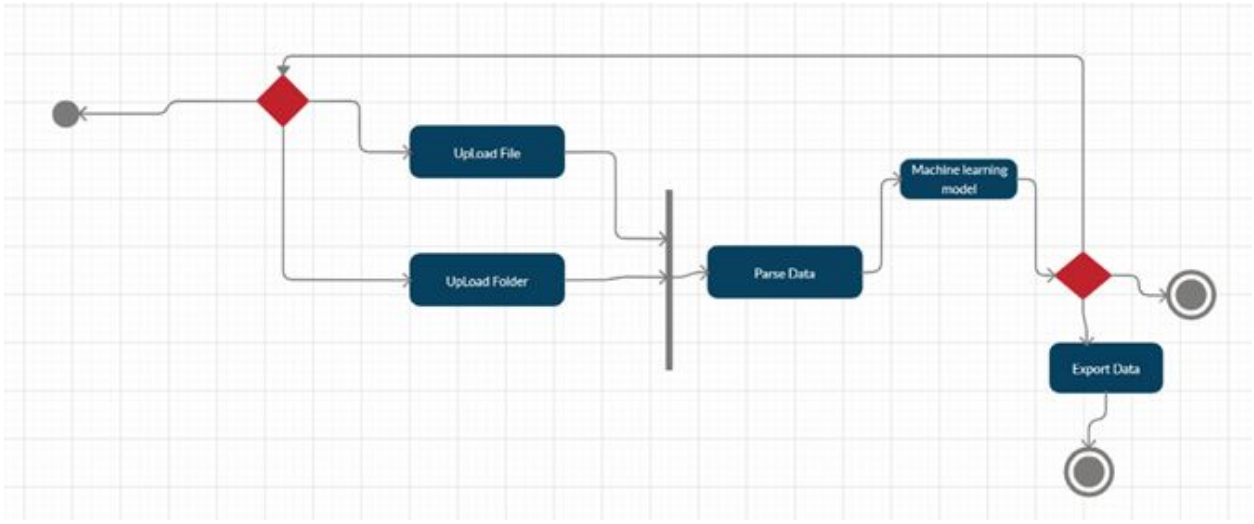
2.2.5 Sequence Diagrams



This sequence diagram outlines the happy case flow through the system.

1. When a file is uploaded it is passed to the machine learning model to be parsed
2. The machine learning model will work to identify the motion
3. The machine learning model then works to count how many times that motion was successfully performed
4. The results are then output
5. If there is no more files to be analysed then the results are displayed and the program is ended
6. Otherwise, steps 2 to 4 are repeated until all files uploaded have been analysed and the results are then displayed and the program ends

2.2.6 State Diagrams



This state diagram depicts a simple flow of events in our system.

- The client inserts or uploads a file or folder to the GUI
- The files will then be parsed
- then the machine learning model will accept the parsed files and provided there were files to analyse
- the results will be produced and exported back to the client.
- If there is no files to be parsed, the program will then terminate
- The program will continue to allow the user to upload files until they choose to terminate

2.2.7 Interface Mockups

When we began this project we drew up some user interface mockups and this was the design we decided to work towards:

After completing some of our coding tasks, we now have a user interface that works and we can see that we have a fairly similar UI design to what we had been looking for: