WYSIWYG Approach to GUI for TensorFlow Deep Learning API

Connor Sedwick, Behnam Saeedi, Collin Dorsett

CONTENTS

1 INTRODUCTION

This section gives a scope description and overview of everything included in this SRS document. The purpose for this document is described and a list of abbreviations and definitions is provided within this section.

1.1 Purpose

The purpose of this document is to give a detailed description of the requirements for the WYSIWYG Deep Learning graphical user interface software. It will illustrate the purpose and complete declaration for the development of the system. It will also explain system constraints, interface and interactions with other external applications.

1.2 Scope

The WYSIWYG Deep Learning graphical user interface is a desktop application which helps users design and test deep learning algorithms. This tool will be developed to comply with Google's Deep learning API called "TensorFlow". This solution allows developers to use their implementation alongside their data in order to create deep learning applications.

1.3 Glossary

- **WYSIWYG:** This word is an abbreviation for "What You See Is What You Get". Microsoft PowerPoint and LibreOfice Impress are two good examples of such system. In this system, the end result is very similar to user's preview of the output during the devlopment.
- **Tensorflow:** TensorFlow is an open source software library for numerical computation using data flow graphs. TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well.
- **GUI:** GUI or Graphical User Interface, Is an interface that allows users to interact with a given system through graphical icons and representation of tasks as oposed to a text base representation.
- **Scene:** In our solution, Scene is the work space where user can place different elements of their program.
- **Block:** A block is a a GUI feature for the user to serve a specific purpose. A block could be variable/Constant, Method, class, probe input or output.
- **Block-Menu:** Block-Menu is a menu that contains all of the items and functionality icons that are ready to be dragged and dropped into the scene.
- **Run:** In our solution, the task of starting executing the program is referred to as Running the program.
- **Stop:** In our solution, stop brings the execution of the program to a halt. Resuming the process requires running the program again. This functionality will reset the state of the execution.
- **Extract:** The task of creating the executable version of the program which is developed by our solution is referred to as extraction of the program. In order for a user to have a stand-alone version of their implemented program, they need to extract their project.
- **Variable/Constant:** Variables are equivalent to programming variables. They are represented by rhombus in our graphical user interface. Constants are similar to variables, however, these values could not be changed while running the program. Constants are too represented by rhombus.
- **Method:** Methods are functions from the API that perform a specific task. They are represented by boxes. The circles on the edges of a method box are inputs and outputs of that method.
- **Class:** In our solution, class refers to an object that contains multiple variables and methods (Public or private). Classes are represented as transparent boxes around methods and variables. This box could be abstracted away to be displays as a solid color box.
- **Abstract:** Abstract is feature of class that will turn the class from transparent to solid color in order to hide the content of the class.

- **Layer:**
- **Channel:** Channels are connections between inputs and outputs that display the direction of data flow and route the data from one block to another.
- **Probe:** Probe is a UI element that displays the content of a channel. Probe can also modify the value which is being transmitted through the probe. If the value of a probe is modified, the modification will happen after the point which the probe is inserted into the channel. Every value that goes though the channel before the probe is left unchanged. (exception would be the pointer probes)
- **Input:** Input is a UI element in our system which allows the user to insert data into their program during the run process.
- **Output:** Output is a UI element in our system which allows the user to see the final result. Outputs indicate discontinuation of a channel.

1.4 References

1.5 Overview

The following chapters provide an overview of the system functionality and system interaction with other libraries and user custom functions. Chapter two introduces different types of users and their interaction with the system. Furthermore, it also mentions the system constraints and assumptions.

The third chapter provides the requirement specifications in detailed terms and a description of the different system interfaces. Different specification techniques are used in order to specify the requirements more precisely for different audiences.

2 Overall Description

This section will give an overview of the TensorFlow WYSIWYG system. The software will be explained in its context to show how the software interfaces with external libraries and introduce the basic functionality of it. It will also describe types of users and functionalities that are available for each type of user based on their needs. By the end, the constraints and assumptions for the system will be presented.

2.1 Product Perspective

This software will consist primarily of the graphical user interface which will communicate with Google's TensorFlow libraries. This software requires access to the TensorFlow libraries for the duration of development, execution and extraction. The interface will provide basic design and flow of data through user's program in a flowchart-like visual representation (WYSIWYG). The underlying code and algorithm designed by the user will be assembled in a file which is built and saved in the background.

2.2 Product Functions

With WYSIWYG graphical user interface, users will be able to design an algorithm by placing and connecting uniquely shaped blocks in a build space. Furthermore, in WYSIWYG GUI, drawing connections between the blocks represents dependencies, calls and flow of the data in the system. Build space represent files, classes or layers that are required to perform user's designated task. Files will be built based on the blocks and connections drawn between them in a build space. These files can be extracted and saved to a user-designated folder at the push of a button. The exctracted files are human readable and in the base language of Google's TensorFlow API (Python).

Developers will be able to set probes on connections drawn between blocks to either modify values or track values as they are manipulated by their algorithm. Helpful python compliant errors, warnings and alerts will let the user know if there is an error with the way they have designed their algorithm or execution errors with thir custom blocks.

2.3 User Characteristics

There are many types of users who will likely use this software. The target users of this tool includes but not limited to people who need to practice high level programming solutions such as machine learning techniques and lack the technical skills that are required to implement them. As it stands, there is only one development layout to be used for the graphical user interface. All users will be using the same interface whether they are students, developers, or employees whose project relies on deep learning software. However, WYSIWYG solution does support custom blcoks for users who are capable of implementing their own functions.

2.4 Constraints

This project is very user based and the only constraints are imposed on usability of our software and the core library that we are trying to mask. This naturally prohibits users from modifying and tampering with the code provided by Google's TensorFlow API since it forces the toolset to recreate all the GUI elements. The graphical user interface must be easily understood by the user and developer. The display of results must be reliable and dependable. The core system has its own limitations which will limit our GUI.

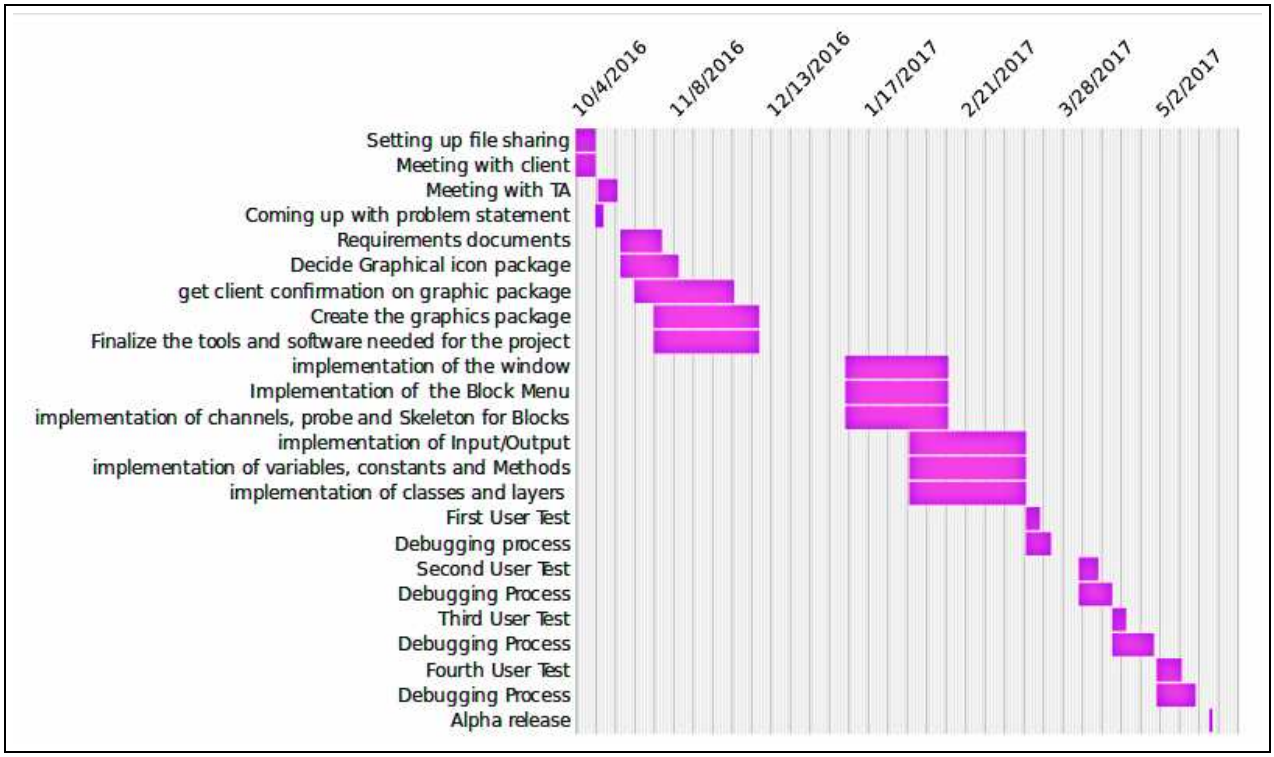
2.5 Assumptions and Dependencies

One assumption we have is that not all users will have prior knowledge on software programming when using this software. A user may not know what a variable is or what a function is and how it relies on variables. When piecing together an algorithm a user may not know the proper syntax that would be required when using a text editing software.

A dependency for this software is access to Google's TensorFlow libraries from the user's memory space. This will require a user to install the libraries along with TensorFlow otherwise the software will not be usable in some scenarios.

2.6 Apportioning of Requirements

In the case that the project is delayed, there are some requirements that could be transferred to the next version of the application.



3 SPECIFIC REQUIREMENTS

This section contains all of the functional and quality requirements of the WYSIWYG system. It provides a detailed description of the system and all its features.

3.1 Interfaces

3.1.1 User Interfaces

The graphical user interface will begin by displaying a menu allowing the user to ”Start a new project.” Once this is chosen, the user will be brought to the main menu screen and build space. From this screen the user will have the ability to choose different types of nodes for either coding or debugging purposes. There will be buttons that allow users to build their file and run it.

Whenever a user wishes to pull the coded file of their algorithm in order to send it or apply it to other code they will have the ability to ”Extract” the file. This will be done by pressing a button and choosing a place in memory to store it. The file should be stored in the form of a .py file.

3.1.2 Software Interfaces

The graphical user interface should communicate with Google’s TensorFlow API. It should also communicate with the personal computer in the background to write the code file and executable file. If an error message is thrown the interface should catch and display the error to the user as a formatted message within the GUI.

3.2 Functional Requirements

This section includes the requirements that specify the fundamental actions of the software system.

ID: FR1
TITLE: Download the application
DESC: A user should be able to download the software from an application store or from GitHub. It should be free to download. The software should also come with a package of TensorFlow included.
RAT: In order for the user to download the software and have access to the require libraries.
DEPEND: None

ID: FR2
TITLE: Open the GUI and start session
DESC: A user should be able to start a new session.
RAT: In order for the user to open up and start a new project with a blank build space.
DEPEND: FR1

ID: FR3
TITLE: Session/Project Directory
DESC: The software should create a directory with a name specified by the user upon starting a new session or project.
RAT: In order for the software to build and store user files related to the current project in a main directory.
DEPEND: FR2

ID: FR4
TITLE: Node Menu
DESC: The software should contain a node menu.
RAT: In order for the user to find nodes for the building of their algorithm.
DEPEND: FR2

ID: FR5
TITLE: Build Space / Scene
DESC: The software should offer a blank space ox within the GUI for users to place nodes..
RAT: In order for the user to find nodes for the building of their algorithm.
DEPEND: FR2

ID: FR6
TITLE: Drag and drop
DESC: The software should allow the user to drag nodes from the node menu and place them in a build space.
RAT: In order for the user to manipulate their algorithm.
DEPEND: FR3

ID: FR7
TITLE: Channel drawing
DESC: The software should allow users to draw a channel between two nodes. This should be done simply by selecting them by clicking on them and choosing an option to ”Draw channel”.
RAT: In order for the user to send values between nodes.
DEPEND: FR3, FR4

ID: FR8
TITLE: Node manipulation
DESC: The software should allow users to enter numerical and string values into variable nodes and probes.
RAT: In order for the user to write their algorithm.
DEPEND: FR4

ID: FR9
TITLE: Background file creation
DESC: The software should create and update user-named files in the background as nodes are added to the build space. Files should be updated in a user-defined directory.
RAT: In order for the user to have a raw and formatted file of their software.
DEPEND: FR2, FR3, FR6, FR7, FR8

ID: FR10
TITLE: Build button
DESC: The software should compile all files in the user’s build directory at the single press of the build button. An executable file should appear in the user’s current project directory if the algorithm compiles correctly.
RAT: In order for the user to create an executable in order to run the software.
DEPEND: FR4

ID: FR11
TITLE: Error handling
DESC: The software should display a text box or error message to the screen letting the user know if there is an error with their build. There should also be a helpful hint on what the user may do to fix their problem.

RAT: In order for the user to know that there is an issue with the format of their current build of their software.
DEPEND: FR10

ID: FR12
TITLE: Data input
DESC: The software should allow the user to include a file of their own data to be used by their software.
RAT: In order for the user to test their software on their own data.
DEPEND: FR2

ID: FR13
TITLE: Layered builds
DESC: A user should be able to create multiple layers to their software by opening multiple build spaces.
RAT: In order for the user to visualize the structure of their software.
DEPEND: FR2

ID: FR14
TITLE: File extraction
DESC: The software should allow the user to save their current build file to a user-defined space in memory.
RAT: In order for the user to have a raw and formatted file of their software in designated storage.
DEPEND: FR4

3.3 Performance Requirements

The requirements in this section provide a detailed specification of the user interaction with the software and measurements placed on the system's performance.

ID: QR1
TITLE: Easy installation
DESC: There should be a simple downloadable package including everything required to begin using the software after installation. There should be a different installation package depending on the operating system being used.
RAT: In order for the user to being using the software without having to worry about file dependencies.
DEPEND: None

ID: QR2
TITLE: Prominent node menu
DESC: The node menu for the flowchart pieces should contain labelled nodes and be legible.
RAT: In order for the user to navigate the menu and find nodes easily.
DEPEND: None

ID: QR3
TITLE: Prominent feature buttons
DESC: The buttons used to build and run the software should be easy to distinguish between each other. The extraction button should be indicated by as a button labelled "Extract". Buttons should respond to a single click and respond within less than a second of the user clicking them.
RAT: In order to avoid confusion when trying to build and run the program.
DEPEND: None

ID: QR4
TITLE: Prominent alerts
DESC: Alerts signaled by the compiler should be displayed to the screen as soon as they are sent. When an error has been fixed its corresponding alert should disappear. Alerts should appear in colored text boxes: red boxes for fatal errors, yellow boxes for warnings.
RAT: This will cut down on confusion when rebuilding a file and trying to find issues with a solution.
DEPEND: None

ID: QR5
TAG: Response time
DESC: The speed at which the interface communicates with the TensorFlow libraries and displays output to the screen.
SCALE: The time of a build.
METER: Measurements obtained by 1000 builds during testing.
MUST: No more than 3 seconds 100% of the time.
WISH: No more than 2 seconds 100% of the time.

4 REFERENCES