Functional Specification - Conor Hanlon (15445378)

1) Introduction

a) Overview

Handwritten notes and essays are a key part of the academic environment. While the digital world is constantly growing, a lot of students, lecturers and professors still prefer writing notes by hand. This has its drawbacks, such as pages possibly being lost or hard to find and not being able to search through notes instantly to find keywords.

My project is centered around converting a user's handwriting to a digital format. I plan to create and train a neural network model that will be able to identify different characters and words in images uploaded by the user and convert this into a text document. The machine learning process will use convolutional and recurrent neural networks to process the training images. The model will be deployed and utilised by a simple web application. Any person can open up the application and upload images of their handwritten notes. They will be given the option to just download the generated documented, or they can save it to their Google Drive if they are signed into their Google account. It will also be possible to share the document via email. This application will help users keep track of any handwritten notes or essays they may want to keep safe for future use.

I will be using machine learning to generate a neural network model which can convert images of handwritten words to text. I plan to use the IAM handwriting dataset to train the network, which contains ~1500 pages of text, ~13000 lines of text and ~113000 individual words.

b) Business Context

My handwriting to text converter would not have many business applications. This is because it is primarily aimed at students and academics who prefer to take notes by hand and want to convert them to documents after. The idea is that it would be made available by educational institutes to be accessed.

c) Glossary

Neural Network: A system of hardware and/or software patterned after the operation of neurons in the human brain.

Image Processing: The analysis and manipulation of a digitized image.

Convolutional Neural Network: A class of deep, feed forward artificial neural networks, most commonly applied to analyzing visual imagery.

Recurrent Neural Network: A class of artificial neural network where connections between nodes form a directed graph along a sequence.

Connectionist Temporal Classification: A type of neural network output and associated scoring function, for training recurrent neural networks.

Activation Function: The activation function of a node defines the output of that node, or "neuron," given an input or set of inputs.

Loss Function: Measures the consistency between the predicted value and the actual label.

Training Process: The process of minimising the loss function in order to make accurate predictions.

GPU: Graphics processing unit, used to handle computation in machine learning.

Docker container: A lightweight, standalone, executable package of software that includes everything needed to run an application.

API: Set of functions that allow the creation of applications which access the features or data of another application.

2) General Description

a) System Functions

Upload Image(s)
The user will be able to upload individual or multiple images of their handwriting in the JPEG or PNG format. Once these files are uploaded, the image processing functionality will separate the file into lines of text which will all be passed into the neural network model to translate the handwriting to text. The network will generate the resulting text, which will be joined together as a block and passed back to the user for approval.

Save document
On the frontend of the application, the user will be presented with a modal containing an input box for the document name as well as box containing their converted text. They can choose to edit it as they please. Once they are satisfied, they have the option to save the text in the form of a Google Document on their Drive or to download it to their local system.

Share document
As well as the option to save a document, users can also share their created Google Document via email. This option will be presented after the document has been created and saved to their Drive.

b) User Characteristics and Objectives

My application has a very large target audience, as it can used by anybody who may have essays or notes written by hand. There is no prior knowledge required to be able to use the application. Therefore the user interface will have to be usable, accessible and intuitive for the end user. I will have to take into account factors like colour contrast and how obvious information is when designing the frontend of the application.

The only requirement for users is that they have a gmail account. This is necessary for logging into my application, as well as the functionality to save files to Drive and share them via email.

c) Operational Scenarios

Scenario 1: Login

Current System State:
User opens up the application and they are presented with the login page.

Informal Scenario:
User clicks on the “Login with Google” icon. The application redirects them to the Google account login page, using the Gmail API. The user then enters their credentials to finish the login process.

Next Scenario:
If the login is successful, the user is redirected to the home page of the application.

Scenario 2: Image upload

Current System State:
The user clicks the “Convert Handwriting to Text” button and a modal is displayed to upload their images in PDF or JPEG format.

Informal Scenario:
The user adds a single image or multiple images and clicks “Convert”. The images are sent in a POST request to the client side of the application. They are segmented into individual lines of handwriting using OpenCV, and all these lines are passed one by one to the neural network model. The application then takes the results from the network and constructs the proposed resulting text.

Next Scenario:
The resulting text is then sent back to the user for them to approve or edit.

Scenario 3: Save Document to Drive

Current System State:
The user is presented with the “Converted Text” modal. This contains “Save to Drive”, “Save and Share via Email” and “Download as PDF” buttons. There is also an input field for the file name as well as a text box containing the resulting text from the neural network.

Informal Scenario:
The user examines the converted text in the modal. They can choose to edit the content or add more in addition to what was generated. Once they are satisfied, they input a name for the file that will be created and click the “Save to Drive” button. This will create a Google Doc in their Drive account with their converted handwriting.

Next Scenario:
When the file is created successfully, the application will display a message to the user then return them to the home page.

Scenario 4: Share Document via Email

Current System State:
The new word document has just been shared to the users Google Drive.

Informal Scenario:
After the file has been saved successfully, the application will ask the user if they would like to share the document. They click the “Share via Email” button and are presented with a new modal. They enter the email addresses that they would like to share the Document with into the input field and click “Share”. The document is then saved to their Drive and shared using the Google Drive API.

Next Scenario:
The file is shared and the application displays a success message before returning the user to the home page.

Scenario 5: Download file locally

Current System State:
The system state is the same as scenario 3.

Informal Scenario:
They can edit the content displayed to them or add more in addition to what was generated. Once they are satisfied, they input a name for the file that will be created and click the “Download File” button. This will create a word document and save it to the user’s computer.
Next Scenario:
When the file is created successfully, the application will display a message to the user then return them to the home page.

Scenario 6: Sign out

Current System State:
The user is logged in on the home screen of the page with no modals displayed.

Informal Scenario:
The user clicks on their profile icon in the top right hand corner of the page. This displays a drop down box with a “Sign Out” button present. The user clicks this sign out button.

Next Scenario:
The user’s current application session is terminated and they are redirected to the login screen.

d) Constraints

User Requirements:
The required ability of the end user should not impact the development process entailed for the application. A well designed and laid out user interface will make clear what steps should be taken by the user at any given point in using the handwriting to text converter.

Time Constraints:
The final delivery of the project is due on the 19th of May. I will have to take this into consideration when timelining all stages of the development process to ensure all necessary steps are feasible in their given duration.

Hardware Requirements:
The end product does not need any special hardware requirements when in production. During the training of the neural network, I will require access to GPUs as running the input images through the neural network is computationally expensive. I plan to split the training process between Google Cloud Platform and AWS, using a combination of credits provided by the School of Computing and credits from student passes.

Deployment Requirements:
Docker is needed to deploy the application. The Dockerfile will pull all necessary modules from online repositories to build the container, which will then host the Ember.js frontend, the Python Flask API and the trained Tensorflow model.

3) Functional Requirements

Gmail API

Description

The Gmail API will be used to handle the session login feature of the handwriting to text converter. Users will be prompted to input their email address and password on opening the application. If successful, the API will ask for the user's consent to their Gmail account data. They will then be redirected to the home page of the web app.

As well as the login, the API is also necessary to be able to share the created document. This is an option provided when the file is successfully saved to their Google Drive account.

Criticality
This is a mandatory step in gaining access to the application. For the purpose of this project, the user must have a Gmail account. This removes the need to develop my own token and key storage session for user logins.

Technical Issues
The main pitfall with using the Gmail API is ensuring there are no issues with maintaining the session persistence. It issues refresh and access tokens periodically to manage this. I will have to keep this in mind, and ensure that the system handles any stale tokens appropriately.

Dependencies with Other Requirements
This functionality affects the entire application, as the user must pass the login phase to enter the app. In particular, it has a strong relationship with the Google Drive API functionality. The session login is a prerequisite for gaining access to the user's Drive account and creating Google Docs files.

Google Drive API

Description
The Google Drive API is used to create a Google Docs file for the logged in user if they choose to do so. This is where the output of the handwriting to text converter will be saved to, once it has been approved by the user and a file name has been designated.

Criticality
This is a very important feature, because the entire point of the application is to generate a word file with the user's handwritten notes in text form. However, it is not the only form of output, as the user will also have the option to download a local version of the file instead of saving it to their Google Drive.

Technical Issues
This function should be relatively straightforward to implement, once the proper steps are taken following the API user manual.

Dependencies with Other Requirements

This feature requires the user to be currently in an active application session via the Gmail API login.

Image Processing

Description
This function is carried out by the Python Flask backend when the images of handwritten text are sent by the Ember.js framework. The server side makes use of OpenCV, a library of functions aimed at dynamic computer vision. The implementation will analyse the images and separate them into individual lines of handwriting, which will then be sent passed one by one through the Tensorflow model.

Criticality
The image processing in my application is a crucial step in generating the text file from user input. The dataset that the neural network is modelled around consists of individual lines of handwriting. If the images cannot be split into their separate lines of text, then the uploaded pictures can't be converted to a text file to the neural network.

Technical Issues
The accuracy of the image processing algorithm must be of a high standard to ensure that the handwriting is parsed correctly. A key factor that I will have to take into account will be ruled pages. It is most common that users will be writing their essays and notes in a notebook or refill pad, so the application will be redundant if it cannot handle this case.

Dependencies with Other Requirements
While the image processing does not require much from other components in the system, the neural network relies heavily on this function to generate its output.

Neural Network Model

Description
The neural network is the core component of my application. It provides the machine learning function that generates our output text from the user inputted images. The proposed model is separated into three different stages:

- **Convolutional Neural Network:** The image input is passed to the CNN layers of the model to extract any relevant features. There are 5 different layers, which all apply filters to the text, followed by an activation function. Finally, the layer gets a summary of the region and outputs a downsized version of the image.
- **Recurrent Neural Network:** The output of the CNN layers is passed to the RNN, which propagates information through the given feature sequence in order to generate character probability scoring. All data is passed through 2 of these layers in order to create accurate scores.
- **Connectionist Temporal Classification:** The CTC layer takes the output matrix from the RNN layers, along with the desired output of the image, and uses these values to compute the loss function of the network. It carries out decoding to

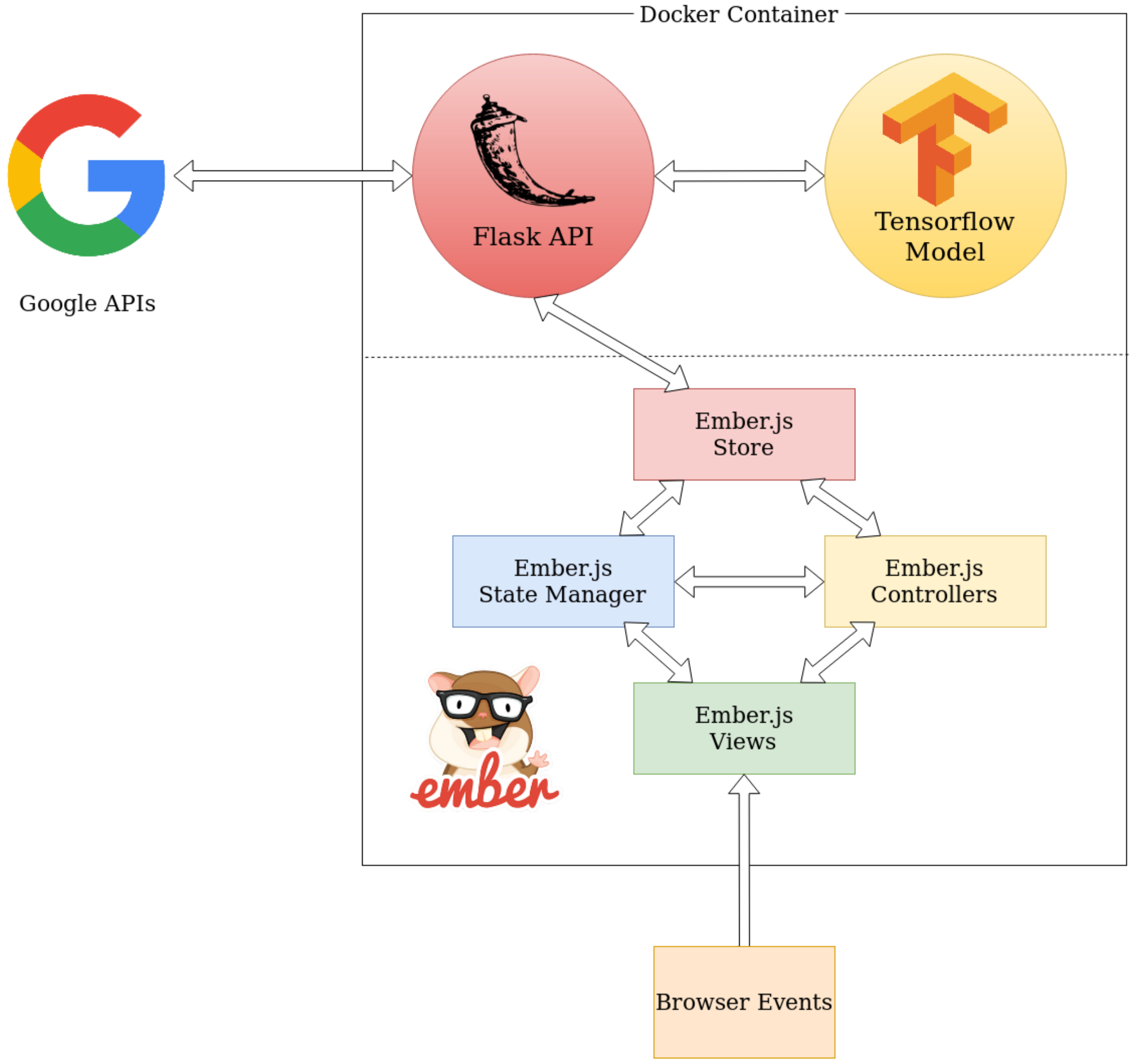
generate the final text output. I will be implementing the *word beam search* algorithm to enhance the decoding process and improve the accuracy.

Criticality
This is the most important function in the system I am designing. Without an accurate model, the application will not be of a satisfactory standard for the end user. It requires a well thought out design and caution during the training process.

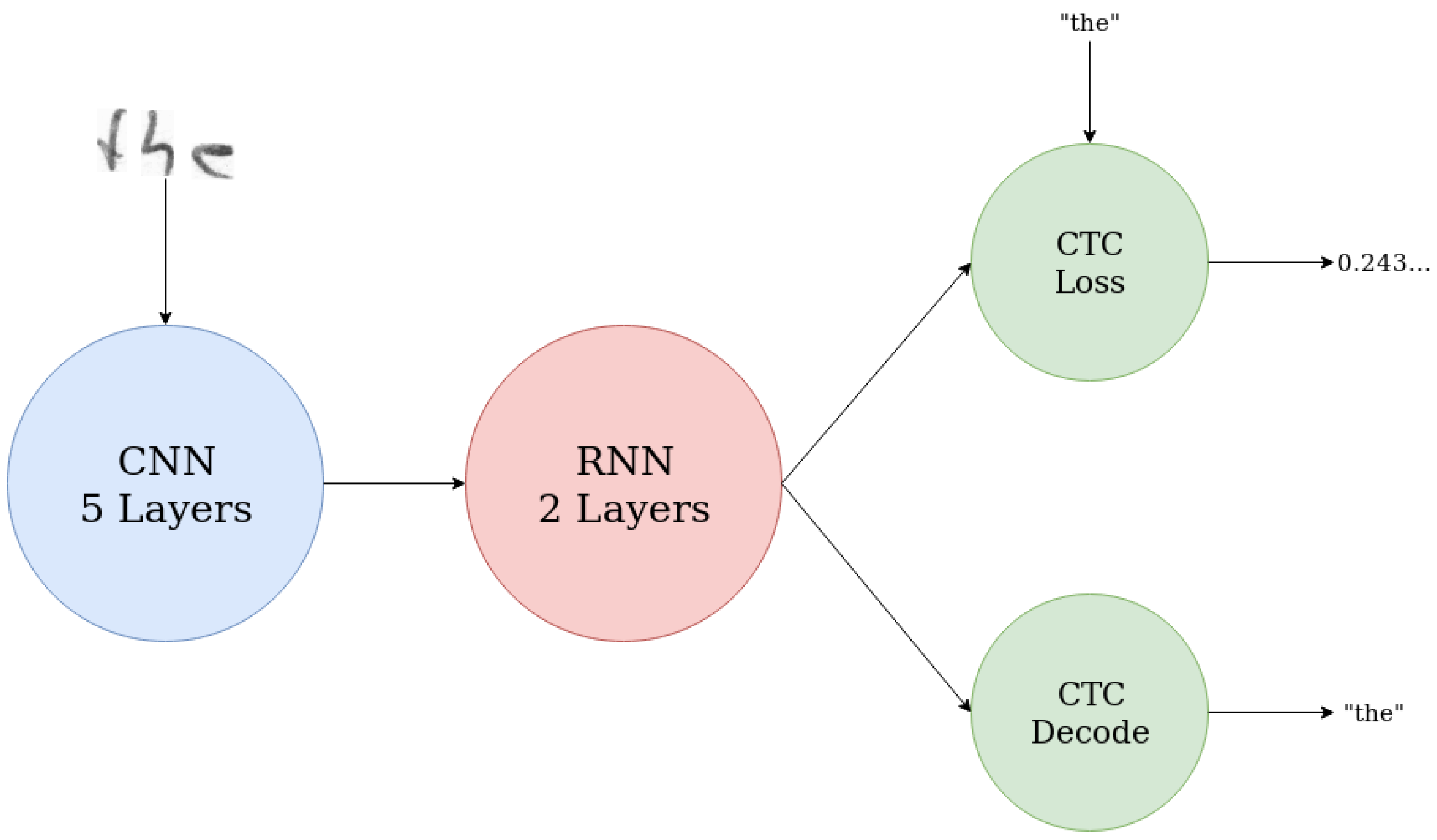
Technical Issues
The biggest pitfall with creating a neural network is the possibility that it will not be accurate after all the training. I will have to take precaution in order to avoid common problems in the process, such as noisy input data or overfitting the network to one particular dataset.

Dependencies with Other Requirements
This function entirely relies on the image processing functionality of the system, as the network cannot take full images of multi-line handwritten text as input data.

4) System Architecture

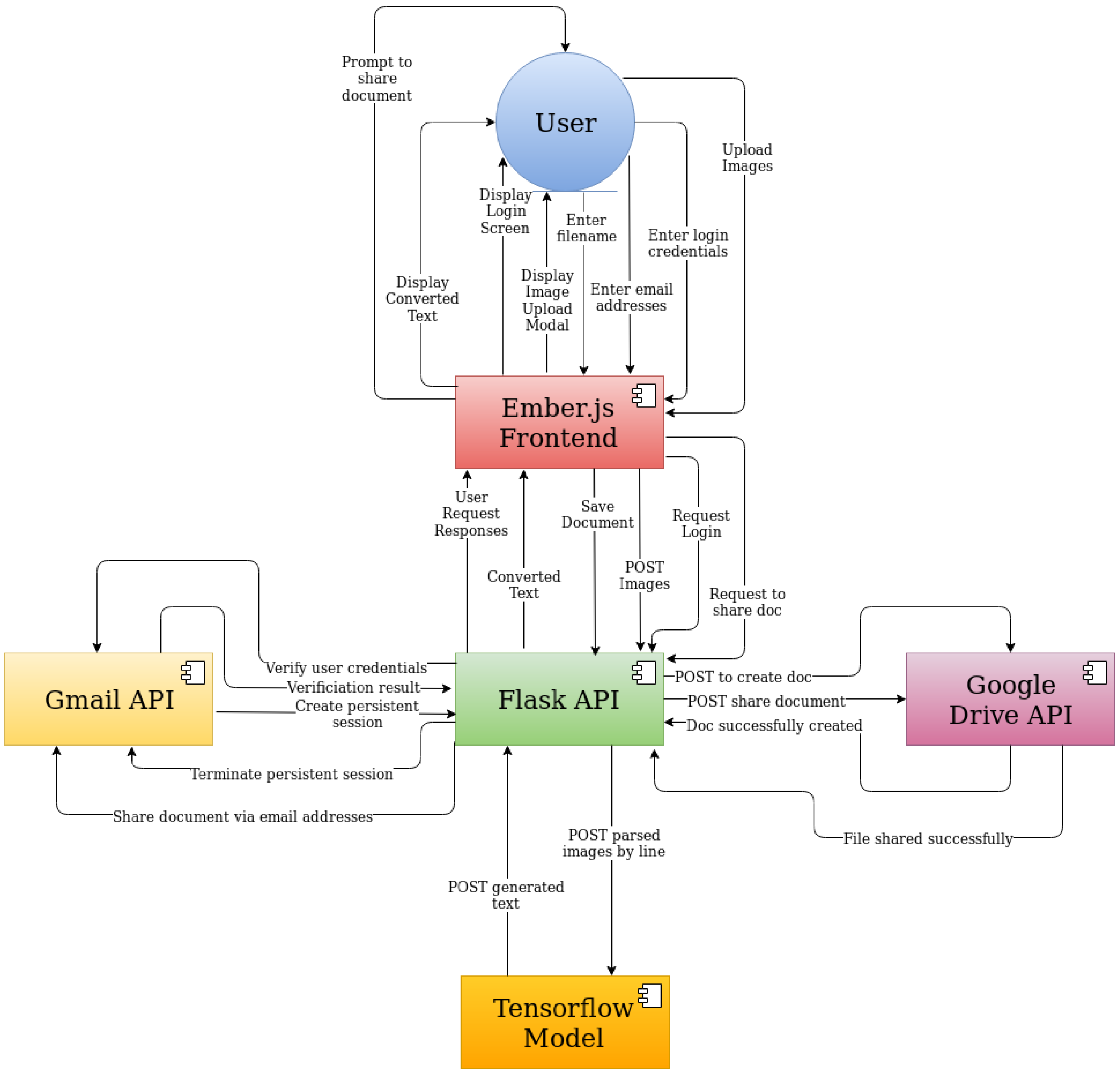


Neural Network Model

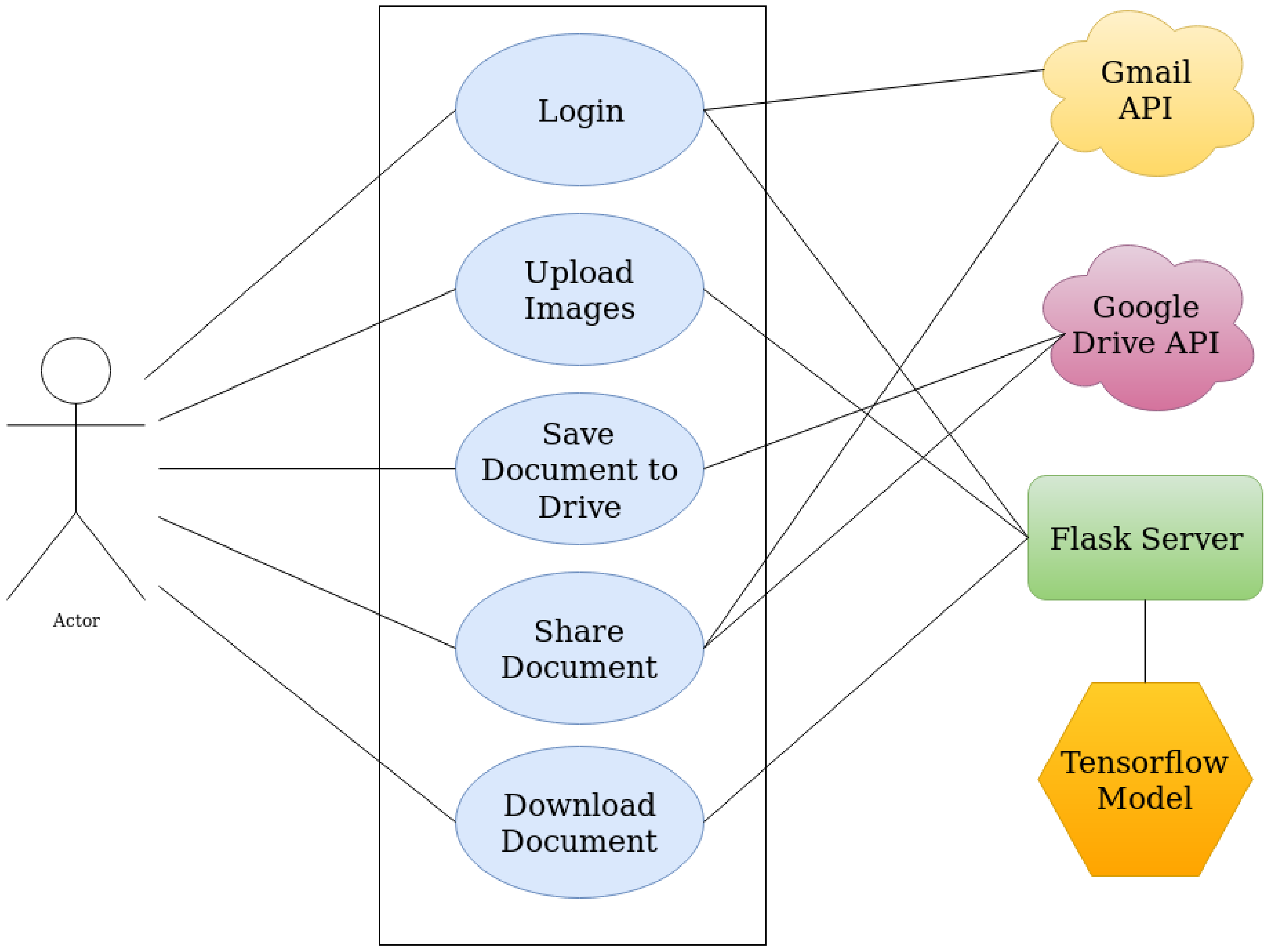


5) High-Level Design

Context Diagram



Use Case Diagram



Use Case Descriptions

USE CASE 1	***Login***
Goal in Context	User enters application session via Gmail login.
Preconditions	User has a Gmail account.
Success End Condition	Login is successful and the user can access the application.
Failed End Condition	Login is unsuccessful and the user must retry.

Actors	**Primary:** User **Secondary:** Gmail API	
Trigger	User opens the web application	
DESCRIPTION	**Step**	**Action**
	1	User clicks “Login via Gmail” button
	2	User is redirected to the Gmail login screen.
	3	User inputs their email address and password.
	4	User clicks the “Login” button.
	5	Gmail API asks for user consent for the web application to have access to their account information.
	6	User clicks “Accept” and is redirected to the application home page.
EXTENSIONS	**Step**	**Branching Action**
	4a	Login credentials are incorrect and the user is prompted to try again.

USE CASE 2	***Image Upload***	
Goal in Context	Uploading images of handwritten text and sending them to the Flask API.	
Preconditions	User is successfully logged in to the application.	
Success End Condition	Images successfully sent to application server.	
Failed End Condition	Images failed to send	
Actors	**Primary:** User	
Trigger	User clicks “Convert Handwriting to Text” button.	
DESCRIPTION	**Step**	**Action**
	1	Application displays modal to the user.

	2	User clicks “Upload Image(s)” button.
	3	User selects the image they want to upload.
	4	User clicks “Convert” button.
	5	Images are sent to the application server.
EXTENSIONS	**Step**	**Branching Action**
	3a	User selects multiple images and chooses to upload all of them.

USE CASE 3	***Save converted text document to Google Drive***	
Goal in Context	Word document is successfully saved to the user’s Google Drive.	
Preconditions	User has uploaded images to be processed.	
Success End Condition	Document is saved to Google Drive.	
Failed End Condition	The document is not created and saved to the Drive.	
Actors	**Primary:** User **Secondary:** Google Drive API	
Trigger	Server sends generated text result back to user interface for approval.	
DESCRIPTION	**Step**	**Action**
	1	User examines the converted handwriting in the modal text box to ensure that it is satisfactory.
	2	User inputs a name for the new file to be created.
	3	User clicks “Save to Drive” button.
	4	The user selects the directory for their new file to be saved to in Google Drive.
	5	The file is created via the Google Drive API.
EXTENSIONS	**Step**	**Branching Action**

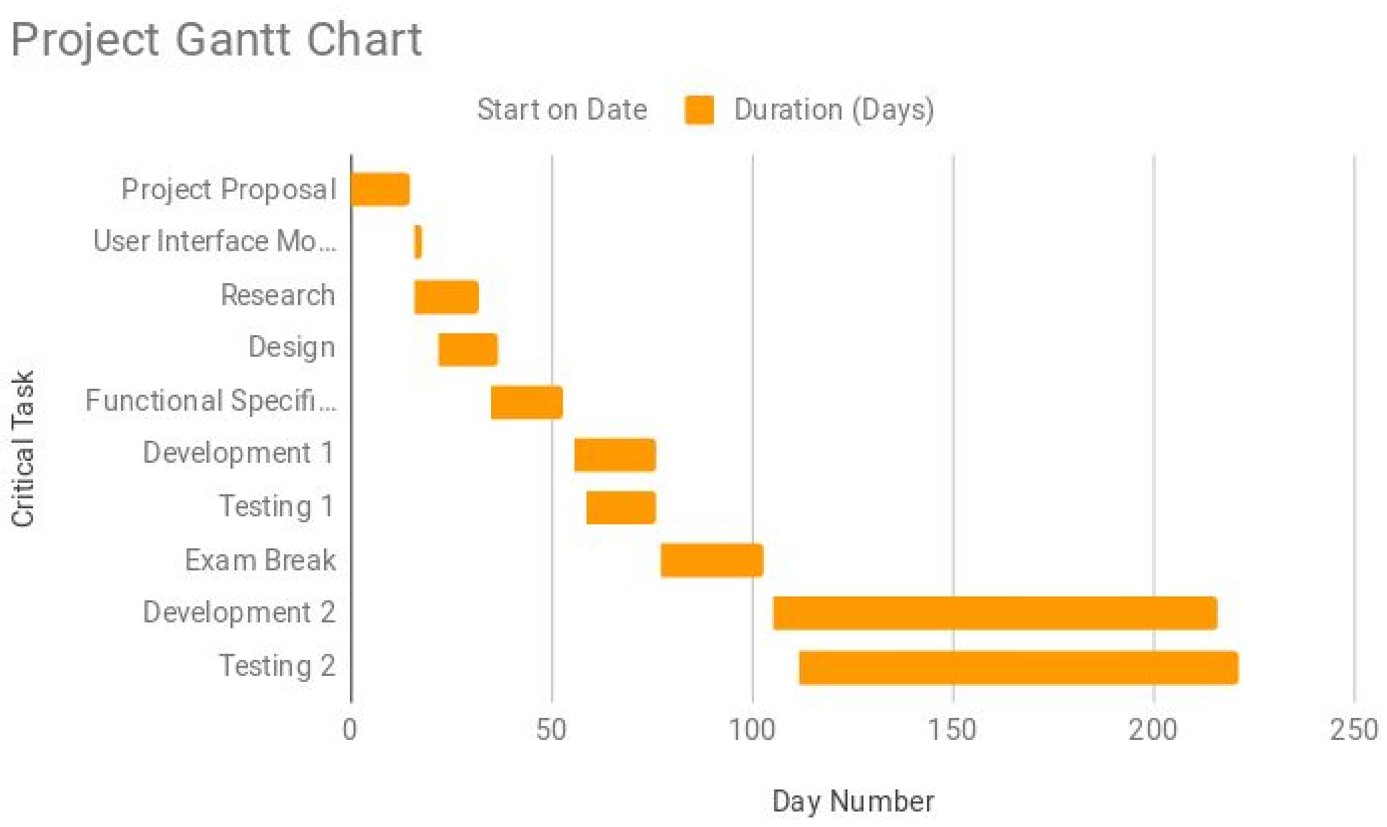
	1a	If the user is not happy with the output, they can make corrections to the text before proceeding.

USE CASE 4	***Share converted text document***	
Goal in Context	Share the newly created file with different Google accounts.	
Preconditions	User has successfully saved the file to their Google Drive.	
Success End Condition	File is shared successfully with the chosen Gmail accounts.	
Failed End Condition	File is not shared successfully.	
Actors	**Primary:** User **Secondary:** Gmail API, Google Drive API	
Trigger	The user selects the “Share via Email” option after the word document has been successfully saved to their Google Drive.	
DESCRIPTION	**Step**	**Action**
	1	The user inputs the email address of the account they would like to share the document with.
	2	User chooses what permissions the shared account will have with the Google Document.
	3	User clicks the “Share” button and the account will be given the specified access to the file.
EXTENSIONS	**Step**	**Branching Action**
	1a	User can choose to input multiple addresses to share with a number of accounts.

USE CASE 5	***Download created word document locally.***
Goal in Context	Create document and download it to user’s machine.
Preconditions	User has uploaded images to be processed.
Success End Condition	Document is created and downloaded successfully.

Failed End Condition	Application fails to create document and it cannot be downloaded.	
Actors	**Primary:** User	
Trigger	Server sends generated text result back to user interface for approval.	
DESCRIPTION	**Step**	**Action**
	1	User examines the converted handwriting in the modal text box to ensure that it is satisfactory.
	2	User inputs a name for the new file to be created.
	3	User clicks “Download File” button.
	4	The file is created and downloaded to the user’s machine.
EXTENSIONS	**Step**	**Branching Action**
	1a	If the user is not happy with the output, they can make corrections to the text before proceeding.

6) Preliminary Schedule



Critical Task	Start Date	End Date	Duration (Days)
Project Proposal	08/10/18	23/10/18	15
User Interface Mockups	24/10/18	26/10/18	2
Research	24/10/18	09/11/18	16
Design	30/10/18	14/11/18	15
Functional Specification	12/11/18	30/11/18	18
Development 1	03/12/18	23/12/18	20
Testing 1	06/12/18	23/12/18	17
Exam Break	24/12/18	19/01/19	26
Development 2	21/01/19	12/05/19	111
Testing 2	28/01/19	17/05/19	109

7) Appendices

Tensorflow: https://www.tensorflow.org/

Flask: http://flask.pocoo.org/

Ember.js: https://www.emberjs.com/

Docker: https://www.docker.com/

Google Drive API: https://developers.google.com/drive/

Gmail API: https://developers.google.com/gmail/api/

OpenCV: https://www.opencv.org/

IAM Handwriting Dataset: http://www.fki.inf.unibe.ch/databases/iam-handwriting-database