

**Maintenance Manual**

**Senior Project**

Tech Tutor

**Version**

1.0

**Product Owner**

Mathew Brimberry

San Juan Unified School District

**Team Byte Brigade**

Angelo Karam

Joshua Grindstaff

Ryan Naveira

Austin Nolte

Jacob Sherer  
 Collin Dunkle

Joaquim Pedroza

Chisom Iwunze

**TABLE OF CONTENTS**

[Introduction 2](#_Toc692765705)

[Implementation of Tools and Technologies 3](#_Toc1115421721)

[Programing languages Used, libraries, frameworks SDK's 4](#_Toc1641901880)

[Database 4](#_Toc923558393)

[Payment Services or other API's 4](#_Toc1691776677)

[Runbook 5](#_Toc1066048627)

[Steps for Development 6](#_Toc1294159372)

[Step 1: Supported OS’s 6](#_Toc448171908)

[Step 2: Installing Python 6](#_Toc1744355197)

[Step 3: Installing python packages necessary for development. 7](#_Toc1920193200)

[Step 4: Troubleshooting 8](#_Toc1570098709)

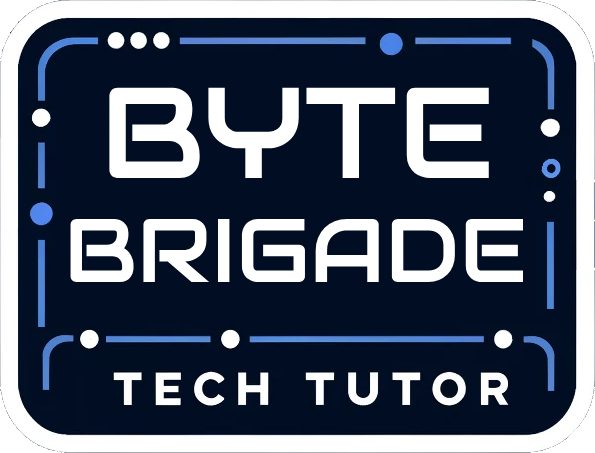
[Deployment 9](#_Toc1158265439)

[User interaction 9](#_Toc93713147)

[Database 9](#_Toc423214514)

[Appendices 10](#_Toc543432901)

## Introduction



TechTutor, developed by the Byte Brigade team, is a Python-based project leveraging PyTorch for machine learning tasks. The primary objective of TechTutor is to simplify the process of grading students' homework by employing machine learning algorithms to analyze images submitted by students. The images are automatically retrieved from a google sheet before being given a grade. The grading process assigns two grades based on a binary pass/fail system where failing submissions are missing at least one of the requirements for an acceptable image or set of images. The grades assigned are separated into two categories: activity packet images and desk images. The image requirements for packets include image clarity, ID and period number on every page of an activity, and all pages of the activity being visible. The desk images requirements include image clarity, desk number visible, calculator number visible, and desk caddy number visible.

## Implementation of Tools and Technologies

This section identifies the specific tools and technologies that would be needed to maintain the software product to its fullest extent.

##### Programing languages Used, libraries, frameworks SDK's

* + IDE used
    - Visual Studio Code (VS Code)
  + Programming languages
    - Python
  + Technologies, libraries and frameworks
    - Pytorch: an open-source machine learning framework used for building and training neural networks.
    - Kivy: a library for building user interfaces that work across multiple platforms.
    - LabelIMG: a graphics interface tool that is used to label objects in images using bounding boxes to allow machine learning programs to more easily recognize images.

##### Database

* + Google Drive: Used for storing and accessing data files related to the application.

##### Payment Services or other API's

* + Google API: Provides programmatic access to Google services, including Google Sheets for managing data. Google Sheets API is limited to 300 reads and 300 writes per minute under the free tier.

## Runbook

### Server goes down

Our program does not rely on dedicated servers to function. Instead, it interacts with Google Sheets and Drive API for its operations. If there are any issues with connections to the Google API, please make sure to check the Google Service Account being used with the application.

### How to Restart the Program

* On Windows: Click the “X” button in the top-right corner of the program window to close it, then relaunch it.
* On Mac: Click the red dot in the top-left corner of the program window to close it, then relaunch it.

### Database becomes corrupt

Data corruption is not applicable as we interact directly with Google through the API and google protects against data corruption for you.

### How to restore from backup

No backup/restore functionality is built into the program since it does not maintain an internal database or server-stored files. Data resides in Google Drive, which maintains its own version history for recovery purposes.

### Third party services go down

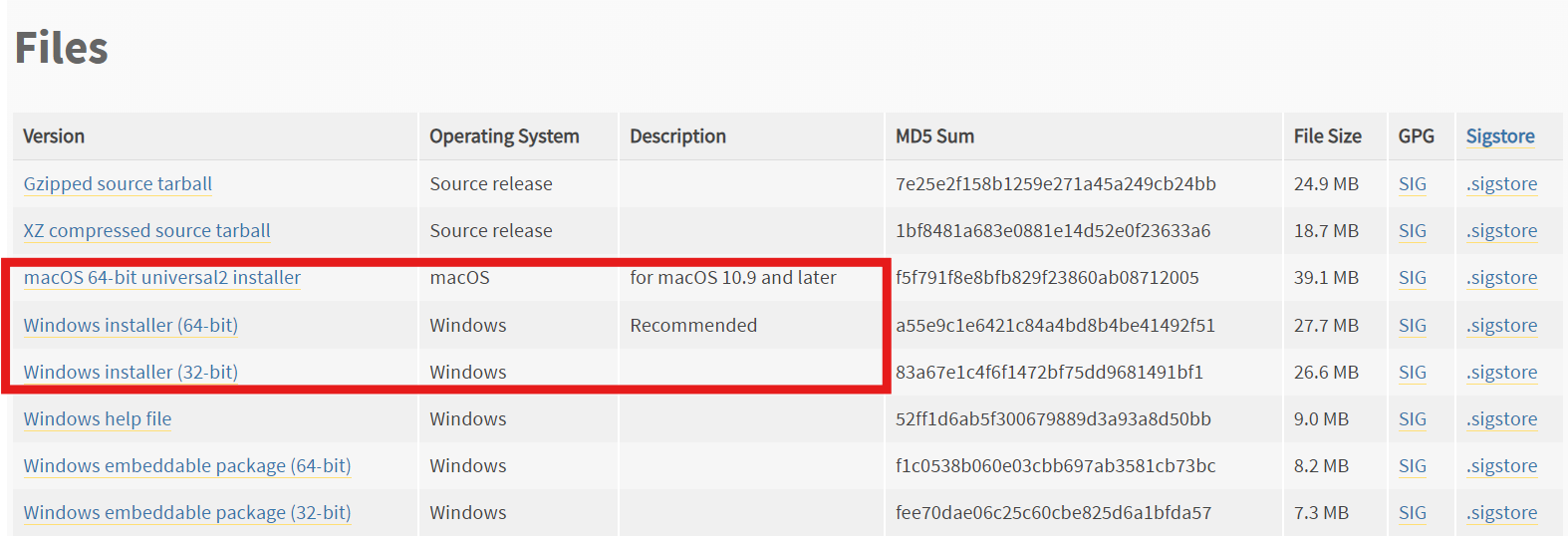
* If Google’s API goes down, please contact Google Support.
  + You can see if the API goes down as when running this application in an IDE you will see debug print statements about the http operations being run. When being run with successful API connections, there will be statements about connections, downloads, and urls being connected to. If these are not there then it is possible that the Google API is down. The likelihood of Google’s API being down for any extended period is slim, and therefore it is not a large concern.

### Steps for Development

#### Step 1: Supported OS’s

* Windows 11 and 10
* MacOS 13 and 14
  + Any other operating systems and versions than the ones listed above are not known to work and are at the discretion of the developer to fix any issues that may arise from using unknown OS versions.
  + Ensure that graphics drivers are up to date for OpenGL to run in version 2.0

#### Step 2: Installing Python

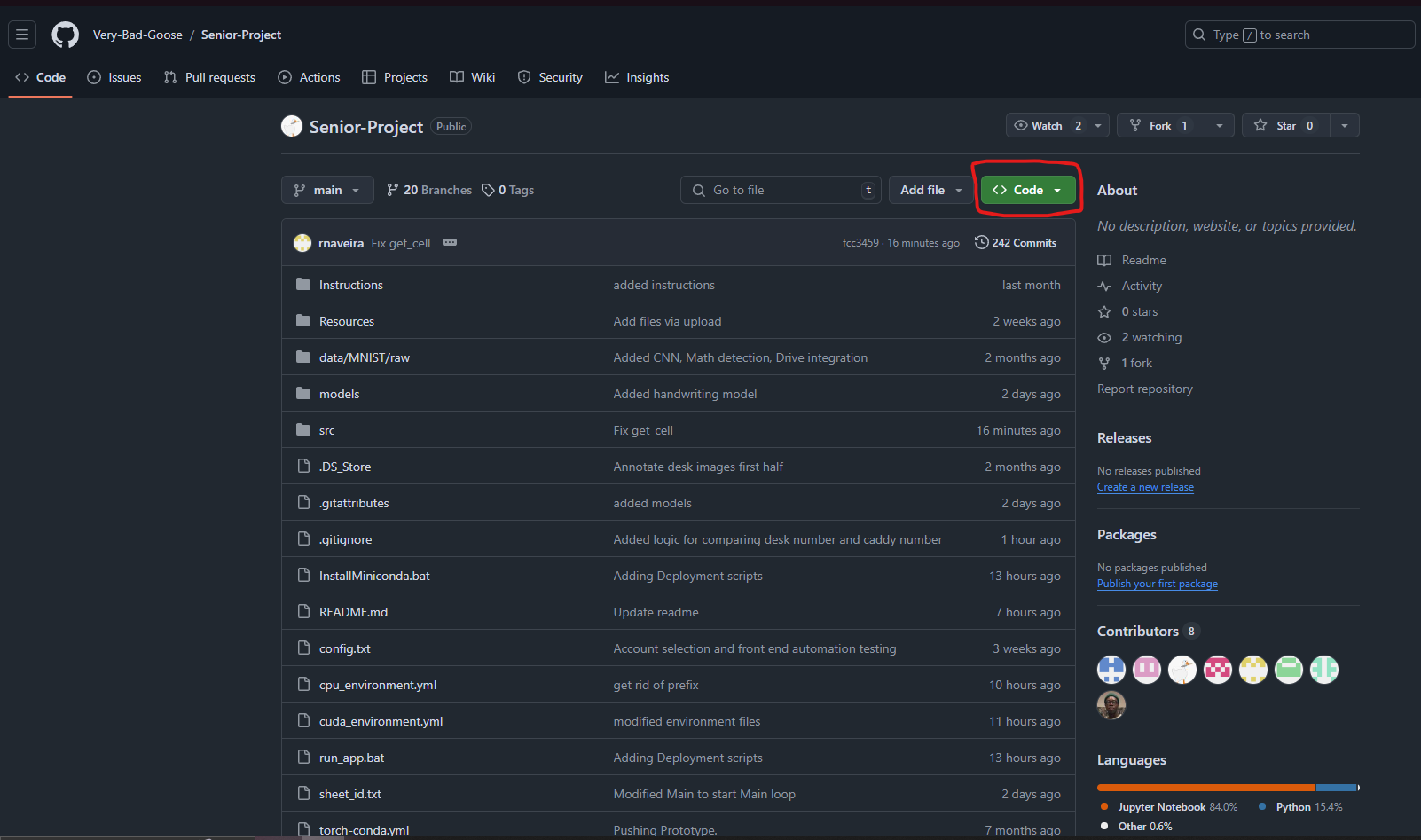
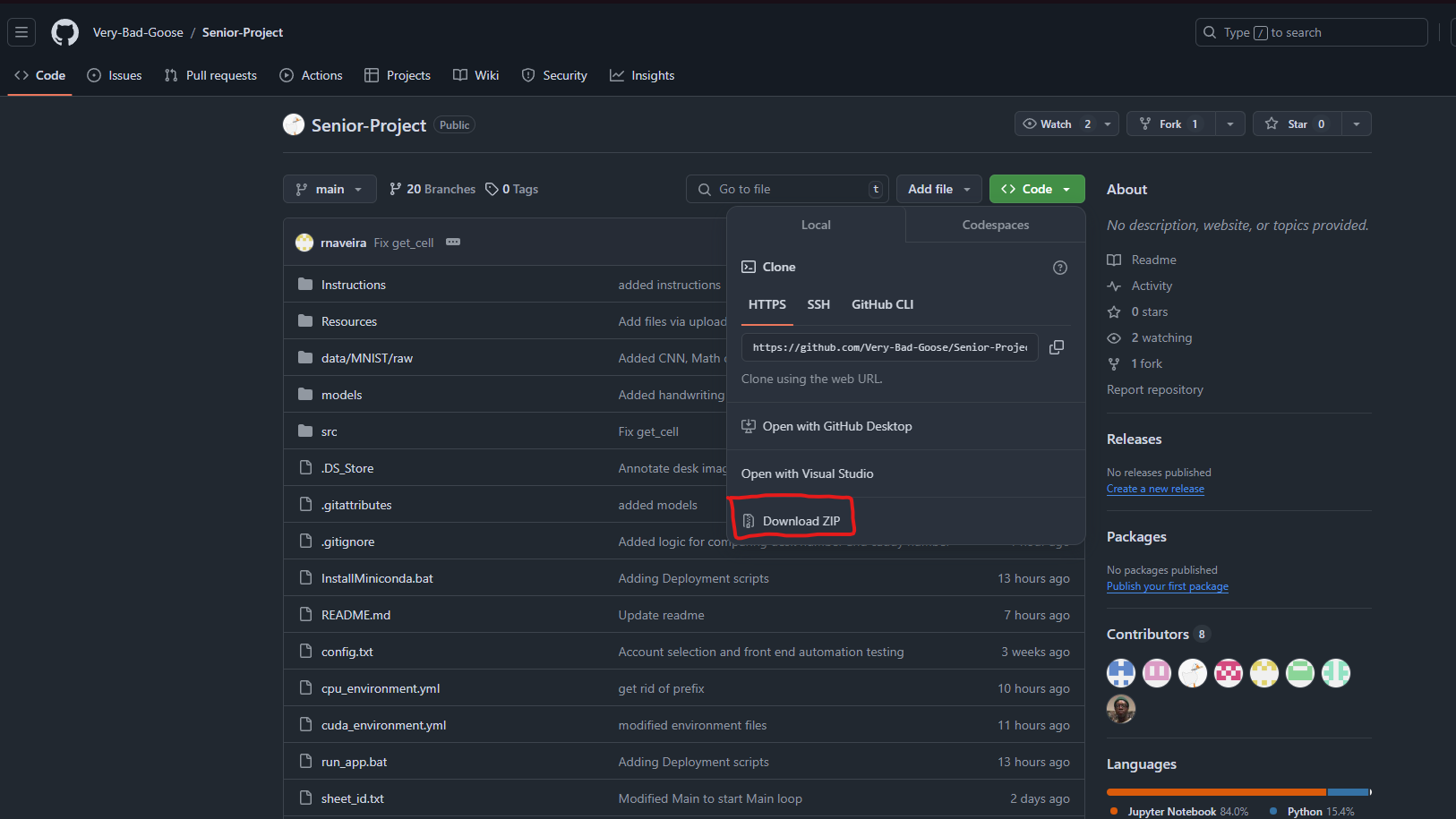
1. Version 3.10.11 or 3.11.9 of Python is necessary for development
   1. Head to this [link](https://www.python.org/downloads/release/python-31011/) and scroll to the bottom of the page. From there download the version corresponding to your respective platform.
      1. The installer version is recommended if you are unsure of what to use.
      2. 
      3. During installation ensure that you check the “**Add Python to PATH”** checkbox. This is **vital** for development purposes.
   2. Ensure python is installed by going to your terminal and running
      1. You should see the following output in the terminal **“Python 3.10.11 or 3.11.9”**

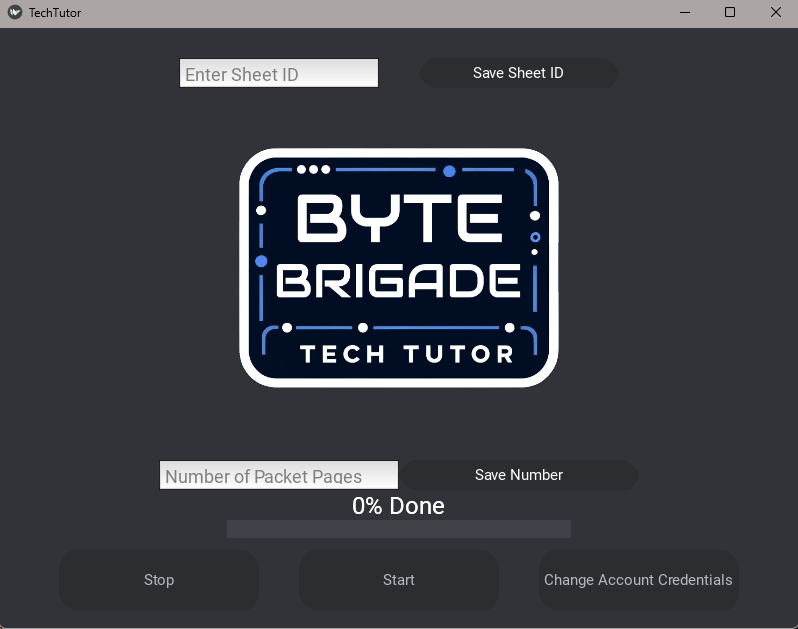
#### Step 3: Installing python packages necessary for development.

##### First ensure that git is installed on your computer by running this command in your terminal.

* + 1. If it is not installed, you can download it [here](https://git-scm.com/downloads).
  1. After installing rerun the command above to ensure it is installed.

##### Clone repo onto your device

* 1. Navigate to your desired install location in your terminal and run
     + 1. This will clone the repo to your device.
     1. If cloning fails:
        1. Go to <https://github.com/Very-Bad-Goose/Senior-Project>
        2. Click on the green code button in the top right of screen.
           1. 
        3. Click download Zip.
           1. 
        4. If download doesn’t work with a warning “Virus detected” follow the steps listed on microsoft’s website here: [Information about the Attachment Manager](https://support.microsoft.com/en-us/topic/information-about-the-attachment-manager-in-microsoft-windows-c48a4dcd-8de5-2af5-ee9b-cd795ae42738) and then retry the download process mentioned in steps 3 and 4.
        5. Then Extract the zip folder to your desired location.

1. Installing miniconda and python packages:
   * 1. Go to your installation folder of the application and find the InstallMiniconda.bat file
        1. Double click to run it and it will start the installation process of miniconda.
           1. Miniconda is a python environment manager and will be necessary for development work.
     2. After it has installed find the run\_app.bat file.
        1. Double click that to start the installation process of the packages necessary for development.
           1. **Note**: this installs a cpu version of pytorch, which is slow for machine learning. If you wish to use a gpu and have a cuda available gpu open run\_app.bat and replace cpu with cuda in “cputorch” and “cpu\_environment.yml”
        2. If the app starts up and you see a screen shown below, you are good to start developing!
           1. 

#### Step 4: Recreating the production environment

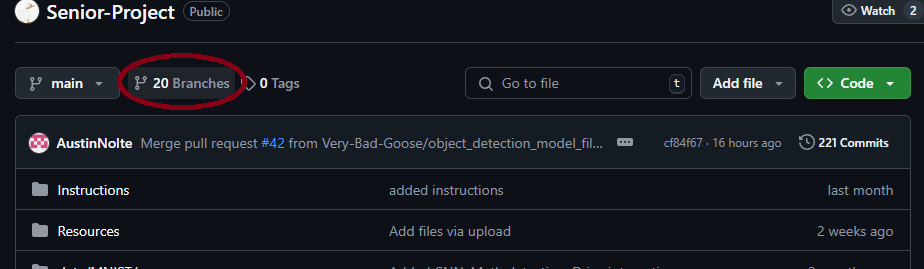
Due to the nature of the production environment being on a shared google sheet, and due to volatility of the sheets themselves, please contact the client to obtain accurate and current information for the sheets, the links the sheets use, and any other production components that they or their developer can offer. Recreating an accurate google sheet to the production google sheet is a critical part of this application. Multiple issues arose during development due to the client needing information from their developer on the current production system, leading to multiple delays and inaccurate information being passed along to the team, therefore if contact can be established with both the client AND their developer the team can more accurately recreate the production environment and avoid these pitfalls that were experienced in the current iteration of the application.

#### Step 5: Making Code Changes and Modifications

1. How to make a code change
   1. An example of adding a button:
      1. Clone the repository. This could be an issue as the repo is quite large and we were running into issues on our last days. If all else fails, download the repository as a zip file and do local changes on your ide.
      2. Create a branch with your name and some description as to not get confused with other branches.
      3. Make changes to the codebase and make sure that you are saving and committing them (commit, if possible, if not then just save them and do it the old-fashioned way with file uploads)
      4. Merge back in. Push your commit if you can, if you couldn’t clone the repository from git then make sure to upload your changed and or new files to your branch using Github’s upload file feature.
      5. Undo changes. If there are changes that you need to undo, there are a few options that you can use:
         1. git reset HEAD <file>
            1. This will unstage the file but retain the changes in your working directory.
         2. Git reset –hard
            1. This will permanently discard all uncommitted changes and should be used with caution.
         3. Git reset –hard HEAD~1
            1. This will undo the commit and discard the changes entirely. It is **irreversible,** so be very careful when doing this.
         4. Git revert <commit-hash>
            1. This will undo a pushed commit

There are more ways than just this to revert any changes made using git, but for now these should provide enough to get some changes undone if necessary. If you require something that is not mentioned above, please search up git commands for undoing any changes.

* + 1. Switch branches. When switching branches, make sure that whatever branch you’re on is as up to date as you need it to be. Sometimes it makes sense to have a branch to be a backup for certain milestones. Switching branches can be done in either your IDE or in github’s website, or through github desktop. In the case of IDE’s it will depend upon the IDE if there’s a GUI option to do it, or you can use the command line of: “git switch <branch-name>”. This works in github desktop and IDEs. To switch branches on the website, please use the branches button at the top of the repository and navigate to the branch that you would like to view, or create a new branch from an existing branch.



**Logging:**

Logging is done almost strictly through the running terminal that appears after using the app\_start.bat or running from an IDE. There are logs that will be shown in the google sheet as well for decisions being made and what the AI is seeing in columns at the end of the spreadsheet.

**How to run automated tests:**

To run automated tests using PyTest, make sure to flip an ignore flag for certain files such as test\_ui\_main.py and test\_google\_sheet.py as those tests are more specific unit testing. To test the google\_sheet operations run the test\_google\_sheet.py file and look at the returns, as well as the modifications made in the sheet itself. Though not necessarily automated, these tests are made to be simpler for users to determine unit testing for specific functions in the google\_sheet.py file.

**How to do “sanity” check to ensure code changes didn’t break anything:**

To ensure that changes are not breaking things, make sure to run the application and compare to expected results, as well as look for exceptions in logs or application crashing. Due to the nature of the application, determining code breakage is typically as straightforward as running the application, confirming compilation, and testing features for both expected results and exceptions.

## Deployment

To ensure a smooth deployment and operation of the TechTutor application, the following steps outline the installation and setup process for the client. This process is designed to be straightforward and user-friendly, leveraging batch files and GitHub.

#### **Step 1: Download the Required Files**

1. **Source Code and Resources**
   1. Ensure graphics drivers are up to date so that OpenGL 2.0 can install
   2. Navigate to the project's GitHub repository.
   3. Download the repository as a ZIP file by selecting the "Code" button and choosing *Download ZIP*.
   4. Extract the contents of the ZIP file to a folder on the client’s local machine.
2. **AI Model Download**
   1. The trained AI models required for the app will be provided in a dedicated Google Drive folder that is shared with the client.
   2. The client should access the folder and download the AI model file directly onto their local machine. Ensure the file is stored in the “models” folder in the project directory.

#### **Step 2: Install Miniconda**

To ensure compatibility and streamline package management, the application is built to run in a Python environment managed by Miniconda. The setup is automated through the following process:

1. Locate the InstallMiniconda.bat file in the folder extracted from GitHub.
2. Double-click the InstallMiniconda.bat file to initiate the Miniconda installation.
   1. This script will download and install Miniconda if it is not already present on the client’s system.
   2. After installation, the script will set up the required Python environment and install all necessary dependencies for the application.

#### **Step 3: Running the Application**

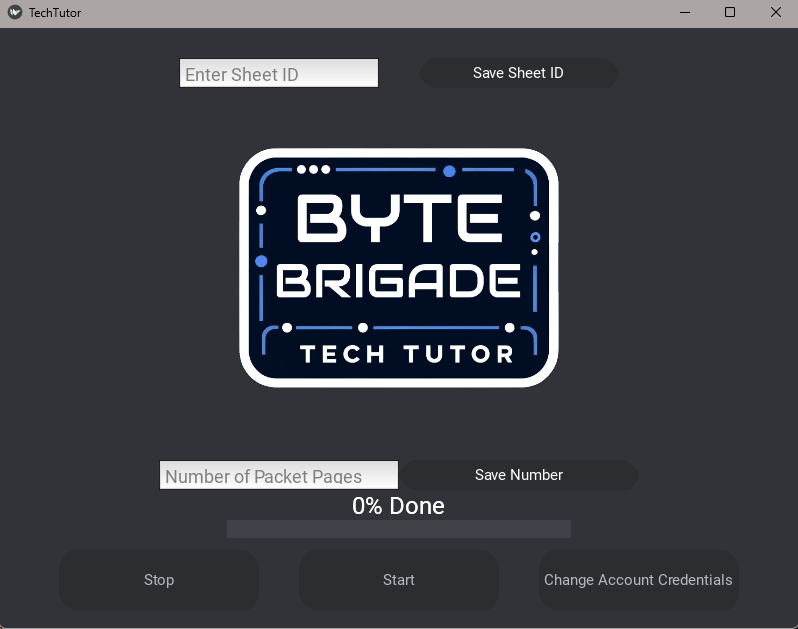
1. Locate the run\_app.bat file in the extracted folder.
   1. This script is pre-configured to activate the appropriate Python environment and start the main application.
2. Double-click the run\_app.bat file to launch TechTutor.
   1. The application will initialize and be ready for use once the GUI loads successfully.

#### **Notes for Deployment**

* Ensure that all files are downloaded and extracted correctly before running the batch scripts.
* Internet access is required during the initial setup to download dependencies through Miniconda.
* Make sure that the Google Sheet environment is accurately set up in accordance with the client’s actual production Google Sheet. This includes links and cell data that is contained in the Google Sheet, as well as permissions for directories and sub directories that those links go to.
* If any errors occur during installation or execution, consult the troubleshooting section in this manual or contact the development team.
* Make sure Graphics Drivers are installed and can run OpenGL 2.0
* Ensure models are correctly saved into the models folder. There are 4 required models, 3 of which are around 150Mb in size while the other is around 4.6Mb

This deployment process has been structured to minimize manual intervention and technical complexity, ensuring an efficient setup for the client.

### User interaction

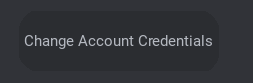


This is the UI screen that allows the user to interact with the program that connects the AI models to the google sheets. This UI is made from Kivy.

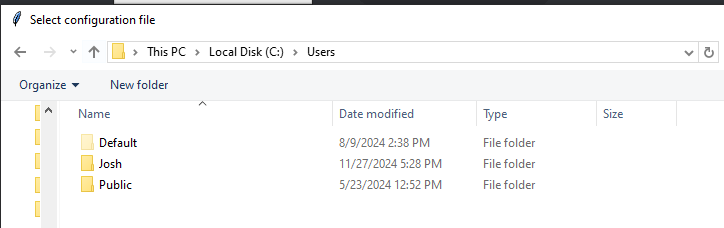
The format of the UI is in the TechTutor.kv file

Most UI programming for UI elements is located in src/main

Logo and background color images are in the src/ui\_Images folder

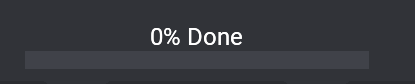


This button will open the operating system’s browser and allow the user to select the file that contains their google API credentials

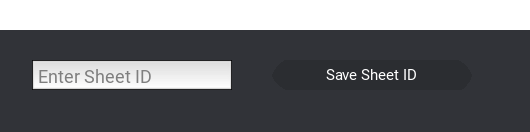
This functionality is in the main.py file in the class MyFloatLayout under the function **select\_json\_file** and **save\_json\_file**.

This will save the credentials in a config file. The saving and loading of this config file is handled under the **save\_config** and **load\_config** functions

Change Account credentials should only have to be dealt with once after installation, after that the application will store the credentials and will be able to call them upon every subsequent run.



This progress bar has its own class called MyProgressBar in main and is drawn in the TechTutor.kv file



This is where the user will enter the URL of the sheet where the AI will read and write too. The functionality for parsing this URL is in main.py in **MyFloatLayout** class, in the **save\_sheet\_id** function.

The sheet ID is text that is apart of the URL.

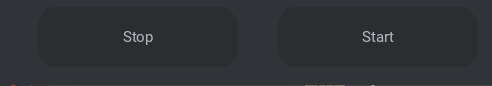
EX: https://docs.google.com/spreadsheets/d/16UPnOQqFubiKkFb5RN2C6B0B\_Tx7A-IYjiu\_uyE\_eg0/edit?gid=0#gid=0

The highlighted part of this URL is the sheet ID for this google sheets.

This will save the last entry into a text file and will read from the text file to run. **Please make sure that you are doing this every time you are entering in a new sheet**. If you do not update this then the application will run on the last sheet that it remembers and it could overwrite information!



The “Save Number” Button and the “Number of Packet Pages” field will store the number that is entered into the field. Enter the expected number of images that should be in the activity packet folder. If the image count does not match it will give the student a 0 and move on. Make sure to do this every time you run the application.



The start button function is implemented in the **start\_press** function and will start the AI. Stop button is implemented in the **stop\_press** function and will stop the AI.

## Database:

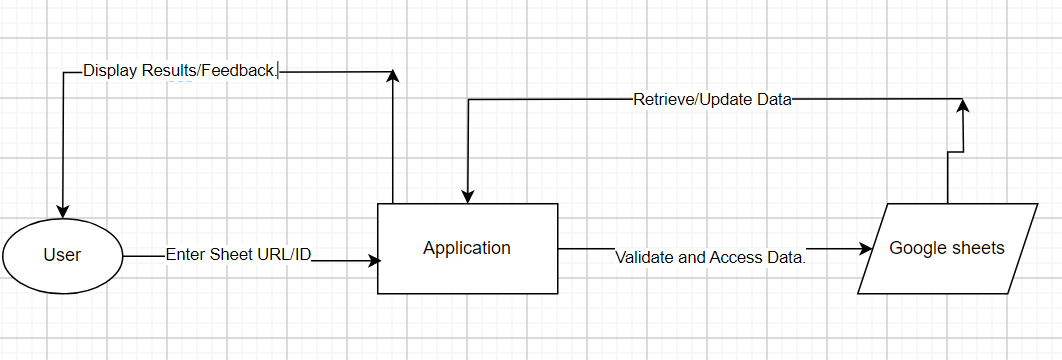
* The project does not use a traditional database system (e.g., SQL or NoSQL). Instead, it integrates with Google Sheets for data storage and retrieval. Sheet IDs are managed through the application and stored temporarily in a text file (sheet\_id.txt)
* Any Entity Relationship Diagrams:

An ERD is not applicable, as the project uses Google Sheets, which serves as a cloud-based data storage platform rather than a relational database.

* Other database diagrams if using NoSQL:

No NoSQL database is used in this project. Google Sheets serves as the data storage medium, and no additional database diagrams are required.

* Other Diagrams showing data flow:

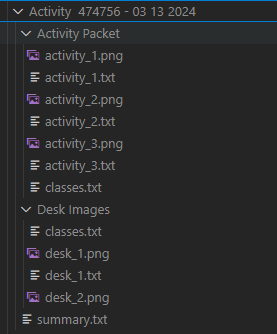


## Prototype Files

The data\_loader\_proto.ipynb and easyOCR\_Proto.ipynb are good ways for testing them to see if you have the correct environment and if you have the data organized in the correct way. They also show how to use the respective data\_loader and easyOCR\_Number\_Recognition models.

## Adding and Formatting Data

Dataloader expects data to be in a certain kind of format. The format is specified by our client and consists of multiple folders containing student submissions. Each folder contains an Activity Packet which has pictures of pages and Desk Images which consist of two desk images. This data should be provided by the client, Matthew Brimberry.



The .txt files that have the same stem name as the images contain the bounding boxes associated with that image. These bounding boxes were created with LabelImg and they have the yolo format. The yolo format is normalized from 0.0 to 1.0. with format (center x pos, center y pos, width, height).

|  |  |
| --- | --- |
|  |  |