Installation Guide

for

AI MUSKS

Version 2.0 approved

Prepared by

Syed Usman - 20002484

Muhammad Shazail Sohail - 19736624

Shuber Ali Mirza - 20027047

Swapnil Rajesh Khatavkar - 20542401

Kshitiz Saini – 20857846

Curtin University, Dubai

1/12/202

Executive Summary

This document provides a manual installation procedure for the AI MUSKS platform. This allows users to customize the environment to some extent. It also leverages users to explore other operating systems such as Debian, CentOS, Red Hat, and Windows Subsystem for Linux. The document explains the method of procedure on Ubuntu 22.04. User is expected to install similar packages/requirements if they attempt to run this on other Operating Systems.

Installation Requirement

- Installation requires Ubuntu 22.04 as the operating system.
- Python 3.6 is installed on the base operating system or under the environment.

Installation of Python 3.6 using Conda (Optional)

Step 1: Update system repositories

Press "CTRL+ALT+T" to open the terminal of your Ubuntu 22.04 and run the below-given command to update system repositories:

Step 2: Install curl package by executing the following commands in the Ubuntu terminal.

```
sudo apt update
sudo apt install curl -y
```

Step 3: Prepare Anaconda Installer

```
cd /tmp
curl --output anaconda.sh https://repo.anaconda.com/archive/Anaconda3-5.3.1-Linux-
x86_64.sh
bash anaconda.sh
```

Step 4: Verify the Anaconda installation by running the following command in the Ubuntu terminal

```
conda list
```

Creating Python 3.6 environment using Conda (Optional)

Step 1: In the Ubuntu terminal run the following commands to create Python 3.6 environment.

```
conda create --name aimusks python=3.6
```

Step 2: In the Ubuntu terminal run the following commands to activate the environment created earlier.

```
conda activate aimusks
```

Downloading platform files

Step 1: Install git to download the platform files.

```
sudo apt update
sudo apt install git -y
```

Step 2: Download platform files from bit bucket platform by using following commands in the Ubuntu Terminal

```
mkdir aimusks
git clone https://Swapnil28527@bitbucket.org/ai-musks/aawb.git
```

Setup AI-MUSKS platform

Step 1: Install the python requirements needed to run the AIMUSK platform.

```
cd aimusks
pip install -r requirements.txt
```

Step 2: Object Detection Model requirements .

```
cd models/research/object_detection
pip install -r requirements.txt
```

Step 3: Install TensorFlow 1.15.

```
cd ../../
wget
https://files.pythonhosted.org/packages/3f/98/5a99af92fb911d7a88a0005ad55005f35b4c1b
a8d75fba02df726cd936e6/tensorflow-1.15.0-cp36-cp36m-manylinux2010_x86_64.whl
pip install tensorflow-1.15.0-cp36-cp36m-manylinux2010_x86_64.whl
```

If all the requirements are successfully installed, then the platform setup is completed. Users can move to the User Manual to run the platform.

File List

The following section highlights important files and their use within the platform.

Glossary:

Term	Definition
Django App	Module of Django that holds views, templates, and models for a specific feature
View	A view is the intermediary between a Django template and a database model. It handles all html requests and database accesses for communication between the front and back ends.
Template	An html file that allows the use of Django code blocks, which allow the use of things like loops and if conditions within the html. It also allows the use of variables, passed by a view from models, to be used in the frontend
Model	A database table, defined directly through the Django framework. Used to store all the data for the website
URL	An address defined for each template to be displayed in a web browser. Calls a view to define what functionality will be allowed on the page

Django:

All file paths start from the root folder of the Django project (where the manage.py file is located).

File	Description
manage.py	Main script to run Django commands, including running the development server, creating super users, making migrations, etc.
db.sqlite3	Current state of the sqlite database. Can be reset if this file is deleted
/musk/settings.py	Main file that holds all the configurations of the whole Django project, including all Django extensions, and custom Django apps.
/musk/urls.py	Main file that holds the urls for all the views from all Django apps.
/musk/asgi.py	Default Django file, document link present in file for more info
/musk/wsgi.py	Default Django file, document link present in file for more info

/musk/__init__.py	Default Django file, document link present in file for more info
/media/	Folder that holds media files, like images, for the whole Django project
/media/profile_pics/	Folder to store custom profile pics, uploaded by users
/media/pageImages/	Folder to store images got from users for all the image input mode pages
/custom/	Folder to store files for the "custom" Django app
/custom/admin.py	Used to register new models for access through Django's admin page
/custom/apps.py	Default Django file, defines the Django app name
/custom/forms.py	Holds Django forms, made for use in templates
/custom/models.py	Holds all the models used for the "custom" Django app
/custom/signals.py	Holds all the signals for the "custom" Django app
/custom/tests.py	Used for testing "custom" Django app
/custom/urls.py	Used for storing all URLs for the views and templates for the "custom" Django app
/custom/views.py	Stores the views used that handle templates and models for the "custom" Django app
/custom/templates/custom/	Standard Django templates folder convention, for storing all the templates for the "custom" Django app
/custom/static/	Folder for storing css, js, images, and other static files for the "custom" Django app
/imageInput/	Folder to store files for the "imageInput" Django app
/imageInput/admin.py	Used to register new models for access through Django's admin page
/imageInput/apps.py	Default Django file, defines the Django app name
/imageInput/models.py	Holds all the models used for the "imageInput" Django app
/imageInput/tests.py	Used for testing "imageInput" Django app
/imageInput/urls.py	Used for storing all URLs for the views and templates for the "imageInput" Django app
/imageInput/views.py	Stores the views used that handle templates and models for the "imageInput" Django app
/imageInput/templates/imageInput/	Standard Django templates folder convention, for storing all the templates for the "imageInput" Django app
/imageInput/static/	Folder for storing css, js, images, and other static files for the "imageInput" Django app
/imageInput/static/outputHTML/	Stores the html files created once a user makes an image input mode webpage. Folder structure used to store html and png files: /outputHTML/<username>/<projectName>/<pageName>.html, <pageName>.png
/imageInput/ImageInputSrc	Contains all the source files for the image input mode machine learning and image detection.
/textInput/	Folder to store files for the "textInput" Django app
/textInput/admin.py	Used to register new models for access through Django's admin page
/textInput/apps.py	Default Django file, defines the Django app name
/textInput/models.py	Holds all the models used for the "textInput" Django app
/textInput/tests.py	Used for testing "textInput" Django app
/textInput/urls.py	Used for storing all URLs for the views and templates for the "textInput" Django app
/textInput/views.py	Stores the views used that handle templates and models for the "textInput" Django app
/textInput/templates/textInput/	Standard Django templates folder convention, for storing all the templates for the "textInput" Django app

/textInput/static/	Folder for storing css, js, images, and other static files for the “textInput” Django app
/textInput/static/outputHTML/	Stores the html files created once a user makes an image input mode webpage. Folder structure used to store html files: /outputHTML/<username>/<projectName>/<pageName>.html
/TextInputSrc/	Stores files used for generating html pages for both the text input and image input modes, hence it’s located in the root folder, instead of just in the “textInput” Django app folder
/templates/	Folder to store templates used for the allauth Django library, used for making post requests, for login, signup, password change, and other user authentication functionalities that Django allauth library offers

Frontend:

File	Description
Custom/templates/custom/base.html	This File contains the navigation bar of the website and container div which uses block content for other pages to be extended
Custom/templates/custom/home.html	This file contains the home page HTML code. It has navigation buttons to a new projects, my projects, and login
Custom/templates/custom/profile.html	This file contains the profile page html code. This page shows the logged-in user information
Custom/templates/custom/projects.html	This file contains the projects page html code. This page shows all the projects of the user and filters it according to the build category under their header
Custom/templates/custom/purpose.html	This file contains the purpose page html code. This page contains the form to collect the basic data of the website
Custom/templates/custom /signuppage.html	This file contains the signup and login page html code. This page is used for login and singup purpose
Custom/static/css/Home.css	This file is used for the designing elements of Home Page
Custom/static/css/Purpose.css	This file is used for the designing elements of the Purpose page
Custom/static/css/Signup.css	This file is used for the designing elements of the Signup and login page
imageInput/templates/imageInput/text_input.html	This file contains the Html code for a single Image input project. This page list total number of pages of the website.
imageInput/templates/imageInput/ text_input_detail.html	This file contains the html code for image input mode. This page gives option to user to select the hand-drawn image of the webpage they want to create
imageInput/static/css/image_input_detail.css	This file is used for the designing elements of the Image input mode page
imageInput/static/js/ image_input_detail.js	This file contains the functionality of how the user can choose images using a

	different method.
imageInput/static/outputHTML/*	This folder contains the saved files of the image input mode projects and their web pages as a backup
textInput/templates/textInput/customization.html	This file contains the Html code for the customization page. This page shows the embedded Html page created by Image or text input mode and gives the form to make changes to the element of the embedded page and also gives the download functionality.
textInput/templates/textInput/text_input.html	This file contains the Html code for a single Text input project. This page list total number of pages of the website.
textInput/templates/textInput/ text_input_detail.html	This file contains the Html code of the text input detail page. This page contains the counters for the element user wanted and also gives the text area for text description they want in their custom build web page
textInput/templates/textInput/customization.css	This file is used for the designing elements of the customization page
textInput/static/css/text_input.css	This file is used for the designing elements of the text_input page
textInput/static/css/text_input_detail.css	This file is used for the designing elements of the text_input_detail page
textInput/templates/textInput/customization.js	This file conatins the functionality for the elements changes that are shown in the customization page form including the download functionality
textInput/static/js/ text_input_detail.js	This file contains the functionality for text input detail page which are counters of the specific element
textInput/static/outputHTML/*	This folder conatins the projects and webpage created by the user using text input mode.

PYtoHTML:

File	Description
textInput/TextInputSrc/Button.py	This class contains the implementation of the button web element. It inherits from Element class and the main functions here are the ElementToCSS and ElementToHTML functions
textInput/TextInputSrc/CheckBox.py	This class contains the implementation of the checkbox web element. It inherits from Element class and the main functions here are the ElementToCSS and ElementToHTML functions
textInput/TextInputSrc/DropDown.py	This class contains the implementation of the dropdown web element. It inherits from Element class and the main functions here are the ElementToCSS and ElementToHTML functions

textInput/TextInputSrc/Element.py	This class is the parent class for all the webpage elements, it contains the getter and setter functions and the fields shared amongst the elements for CSS
textInput/TextInputSrc/FlexiBox.py	This class contains the CSS and the ElementToCSS as well as the ElementToHTML function of the flexibox that is used in the webpage
textInput/TextInputSrc/Footer.py	This class contains the implementation of the footer web element. It inherits from Element class and the main functions here are the ElementToCSS and ElementToHTML functions
textInput/TextInputSrc/Form.py	This class contains the CSS and the ElementToCSS function for the form component of the webpage.
textInput/TextInputSrc/Header.py	This class contains the implementation of the header web element. It inherits from Element class and the main functions here are the ElementToCSS and ElementToHTML functions
textInput/TextInputSrc/Image.py	This class contains the implementation of the Image web element. It inherits from Element class and the main functions here are the ElementToCSS and ElementToHTML functions
textInput/TextInputSrc/Label.py	This class contains the implementation of the label web element. It inherits from Element class and the main functions here are the ElementToCSS and ElementToHTML functions
textInput/TextInputSrc/Link.py	This class contains the implementation of the link web element. It inherits from Element class and the main functions here are the ElementToCSS and ElementToHTML functions
textInput/TextInputSrc/Navbar.py	This class contains the implementation of the Navbar web element. It inherits from Element class and the main functions here are the ElementToCSS and ElementToHTML functions
textInput/TextInputSrc/TextArea.py	This class contains the implementation of the text area web element. It inherits from Element class and the main functions here are the ElementToCSS and ElementToHTML functions
textInput/TextInputSrc/Paragraph.py	This class contains the implementation of the paragraph web element. It inherits from Element class and the main functions here are the ElementToCSS and ElementToHTML functions
textInput/TextInputSrc/TextBox.py	This class contains the implementation of the text box web element. It inherits from Element class and the main functions here are the ElementToCSS and

	ElementToHTML functions
textInput/TextInputSrc/TextNLP.py	This class is responsible for the NLP part of the program. The process function of this class takes a text paragraph as an input and outputs a list with the object, attributes and adjectives
textInput/TextInputSrc/Webpage.py	This class is the main backbone of the image input mode. It contains the list of elements on the webpage and the WebpageToHTML function converts this list into a html webpage
textInput/TextInputSrc/ImageToHTML.py	This class is used to convert the text file outputted by the image input mode into an HTML page using the createHTML function of this class
textInput/TextInputSrc/RadioButton.py	This class contains the implementation of the radio button web element. It inherits from Element class and the main functions here are the ElementToCSS and ElementToHTML functions

ML:

File	Description
imgtInput/imgInputSrc /Preprocess.py	This function is used to process the uploaded image into the correct size and convert the image into a grayscale image for better recognition.
imgtInput/imgInputSrc /main.py	This is the main function, which receives the image, processes the image using OpenCV and TensorFlow and creates a list of elements that are passed to the next function to generate the HTML code.
imgtInput/imgInputSrc **/frozen_inference_graph_816.pb**	Trained model weights used to recognize the objects.

User Manual

for

AI MUSKS

Version 2.0 approved

Prepared by

Syed Usman - 20002484

Muhammad Shazail Sohail - 19736624

Shuber Ali Mirza - 20027047

Swapnil Rajesh Khatavkar - 20542401

Kshitiz Saini – 20857846

Curtin University, Dubai

22/11/2022

Table of Contents

Revision History

Name	Date	Reason for Changes	Version
AI MUSKS - Product Manual	01 – December – 2022		1.0

Executive Summary

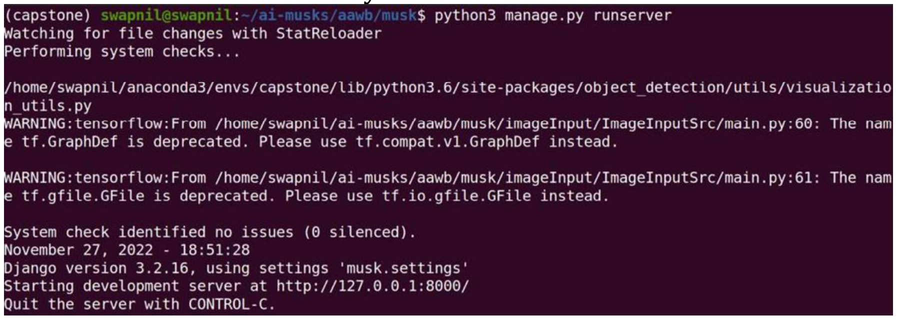
This section of the document will give a walkthrough of the flow of events when a user uses the AI MUSKS website.

Execution:

To start the program, we start the Django server. To start the Django server, open the Ubuntu terminal and execute the following command.

```
python3 manage.py runserver
```

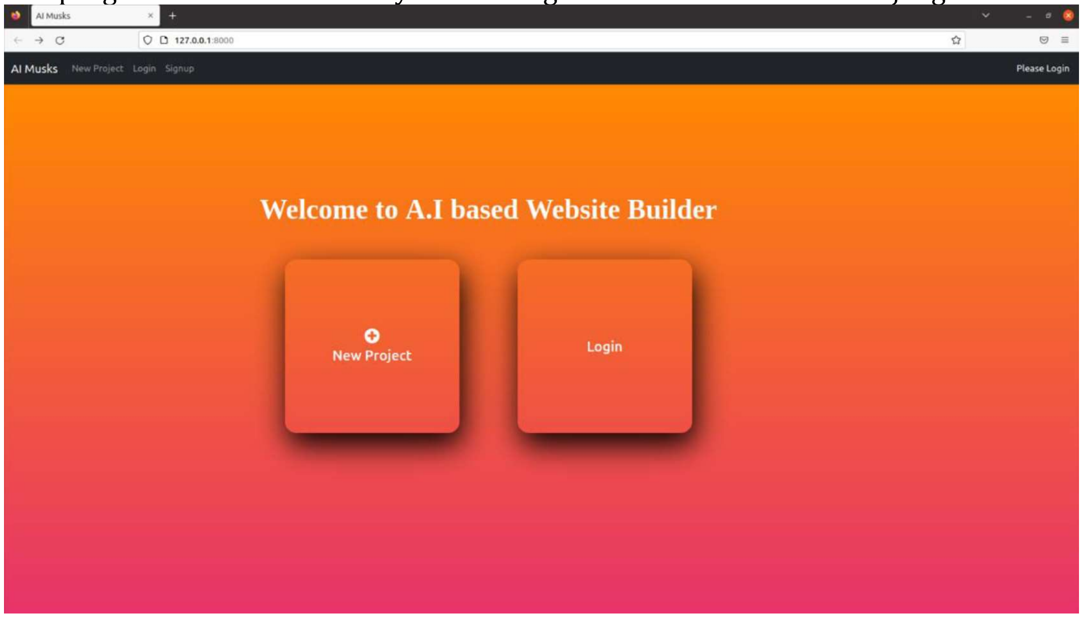
The output below shows that it runs successfully.



```
(capstone) swapnil@swapnil:~/ai-musks/aawb/musk$ python3 manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

/home/swapnil/anaconda3/envs/capstone/lib/python3.6/site-packages/object_detection/utils/visualizatio
n_utils.py
WARNING:tensorflow:From /home/swapnil/ai-musks/aawb/musk/imageInput/ImageInputSrc/main.py:60: The nam
e tf.GraphDef is deprecated. Please use tf.compat.v1.GraphDef instead.

WARNING:tensorflow:From /home/swapnil/ai-musks/aawb/musk/imageInput/ImageInputSrc/main.py:61: The nam
e tf.gfile.GFile is deprecated. Please use tf.io.gfile.GFile instead.

System check identified no issues (0 silenced).
November 27, 2022 - 18:51:28
Django version 3.2.16, using settings 'musk.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

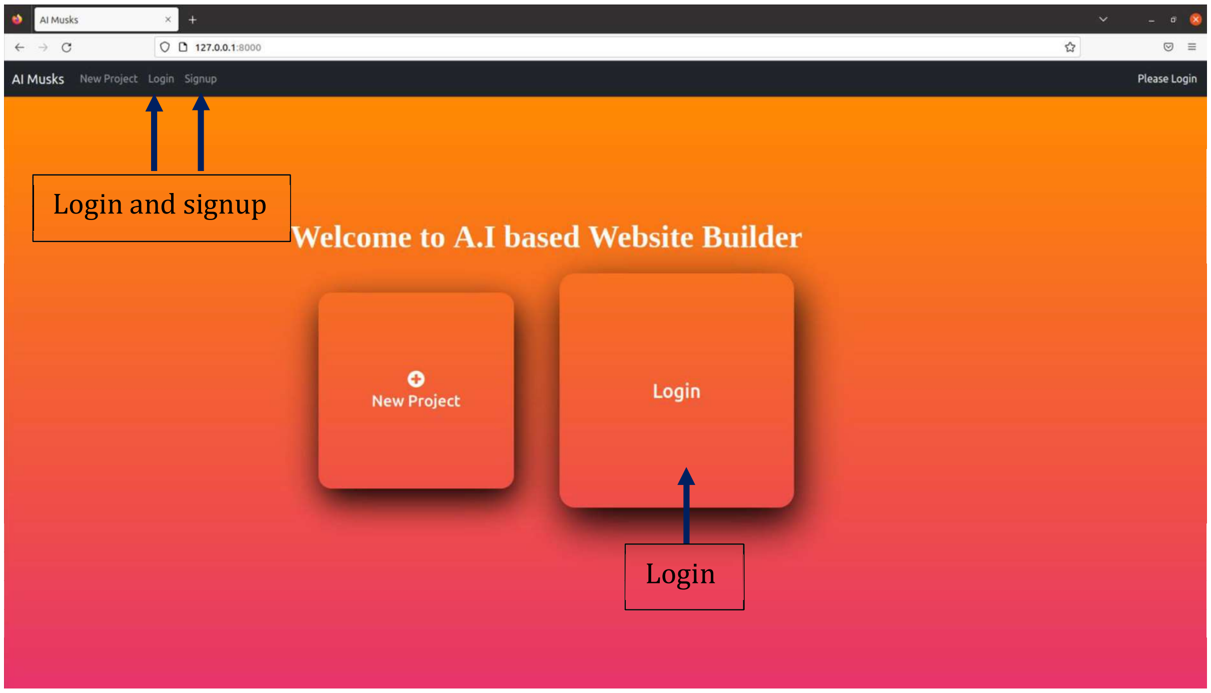
Accessing GUI:

The program can be accessed by users using an HTML5 browser. The Django server runs on TCP port 8000.

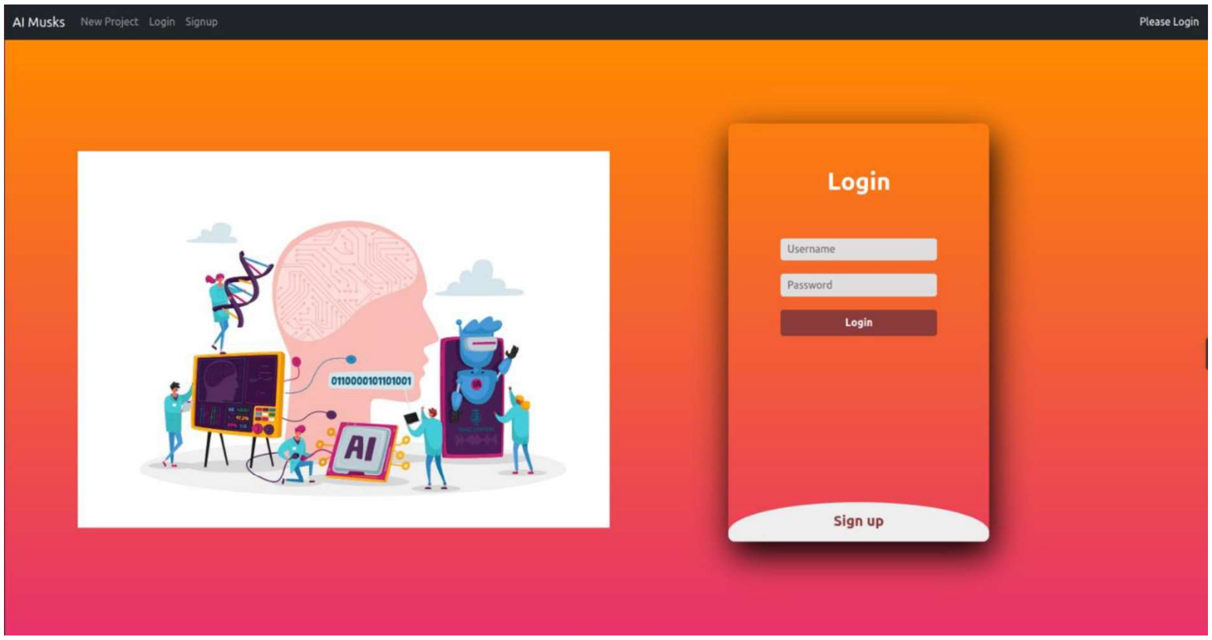


Login:

Users can log in if they are previously registered.

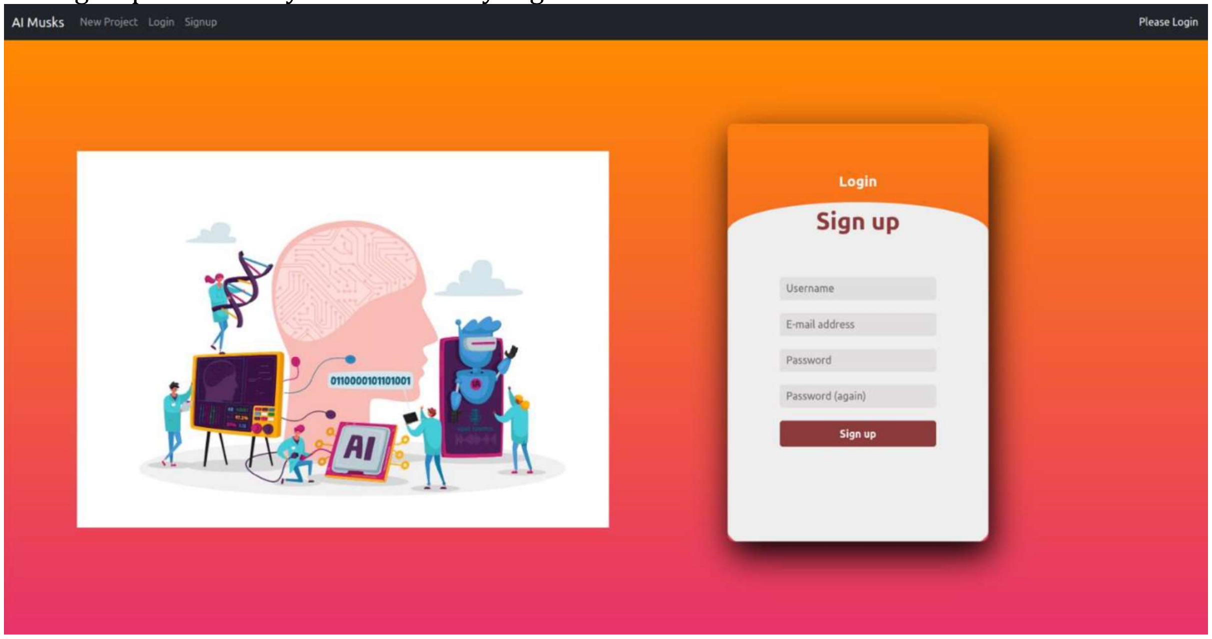


Once the user clicks on the login button, the user will be presented with Login.



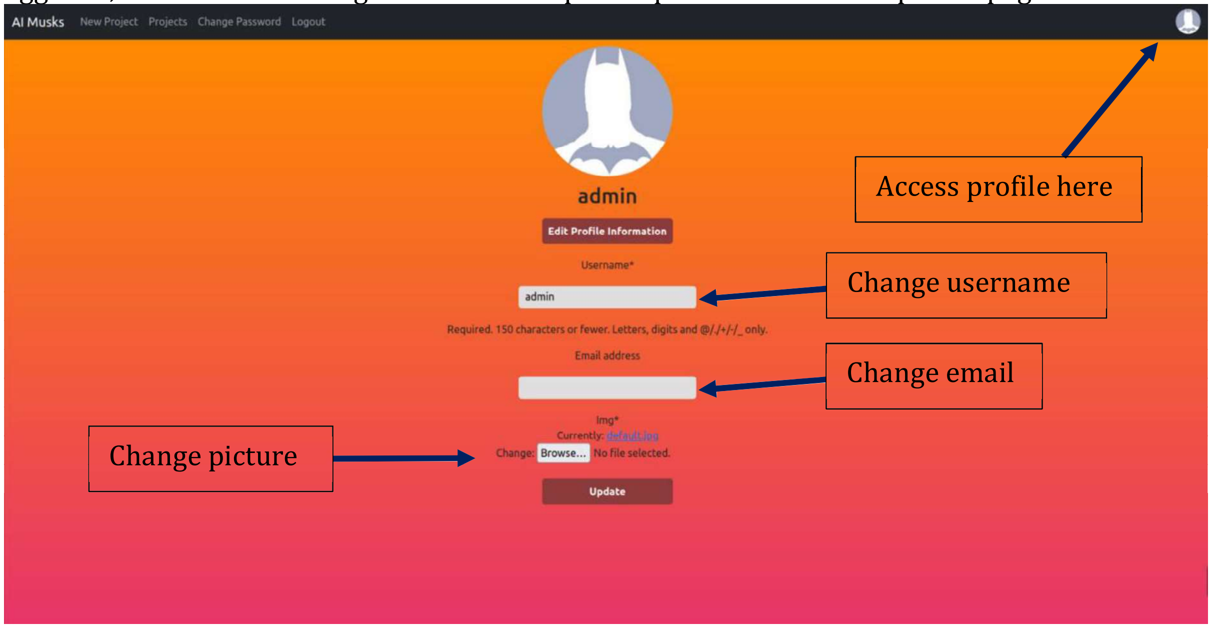
Sign-Up:

Users can sign-up in case they are not already registered.



User Profile Customization:

Once logged in, the users can change their info and profile pictures from their profile page



Username*

admin

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email address

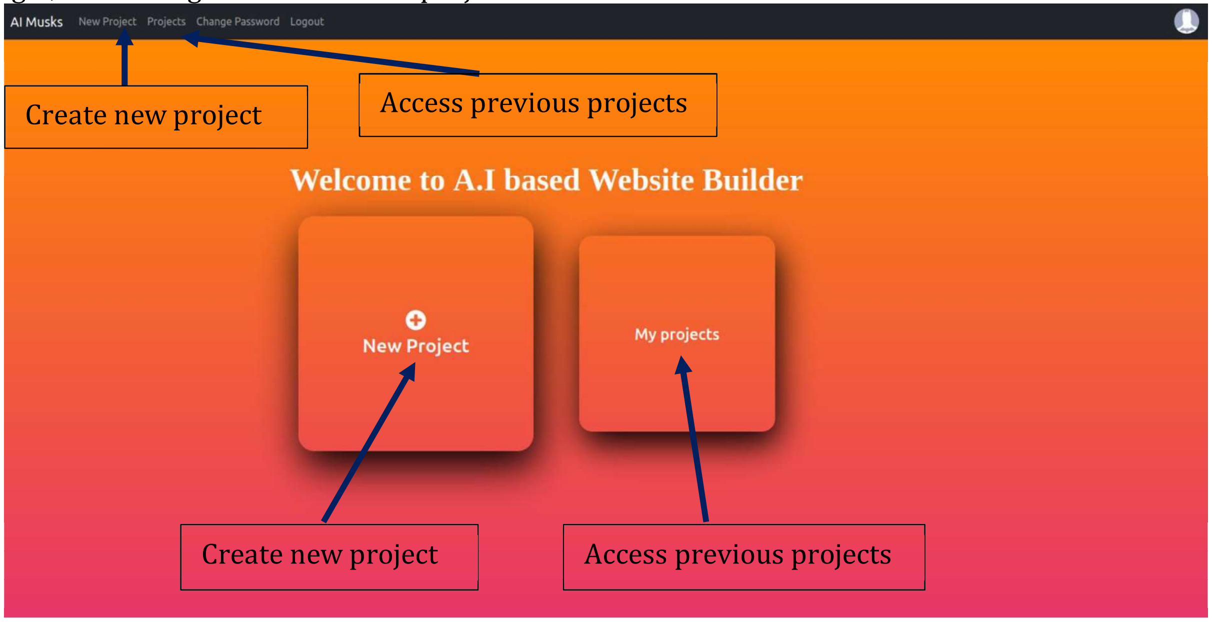
Img*

Currently: default.jpg

Change: Browse... No file selected.

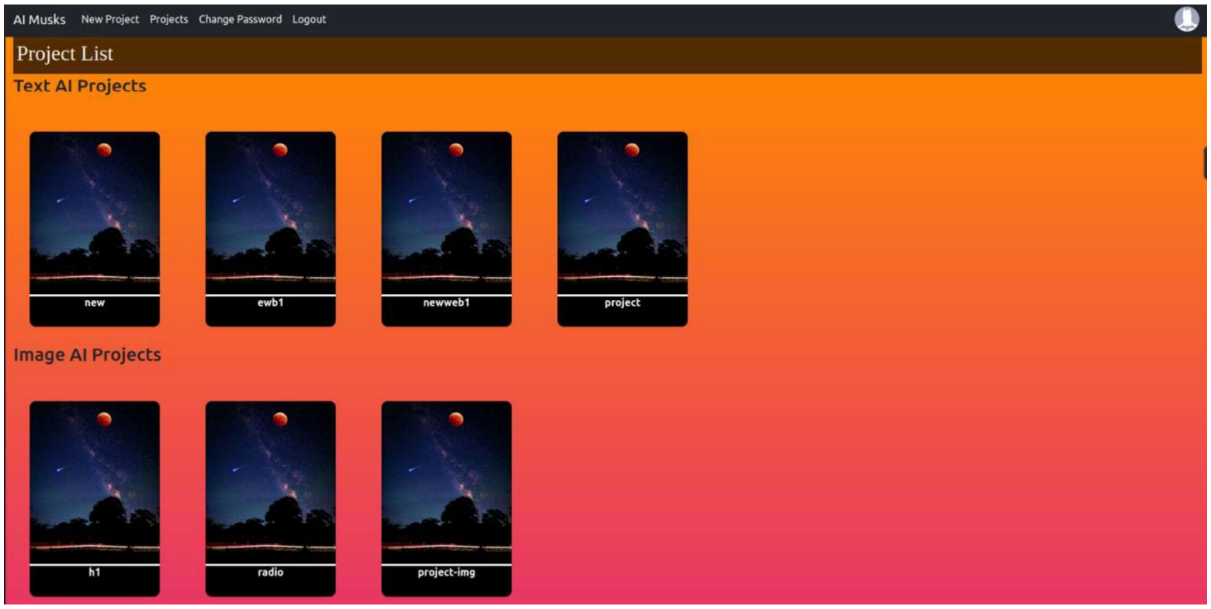
User Home Page:

Post login, users can go to create a new project from the links shown below



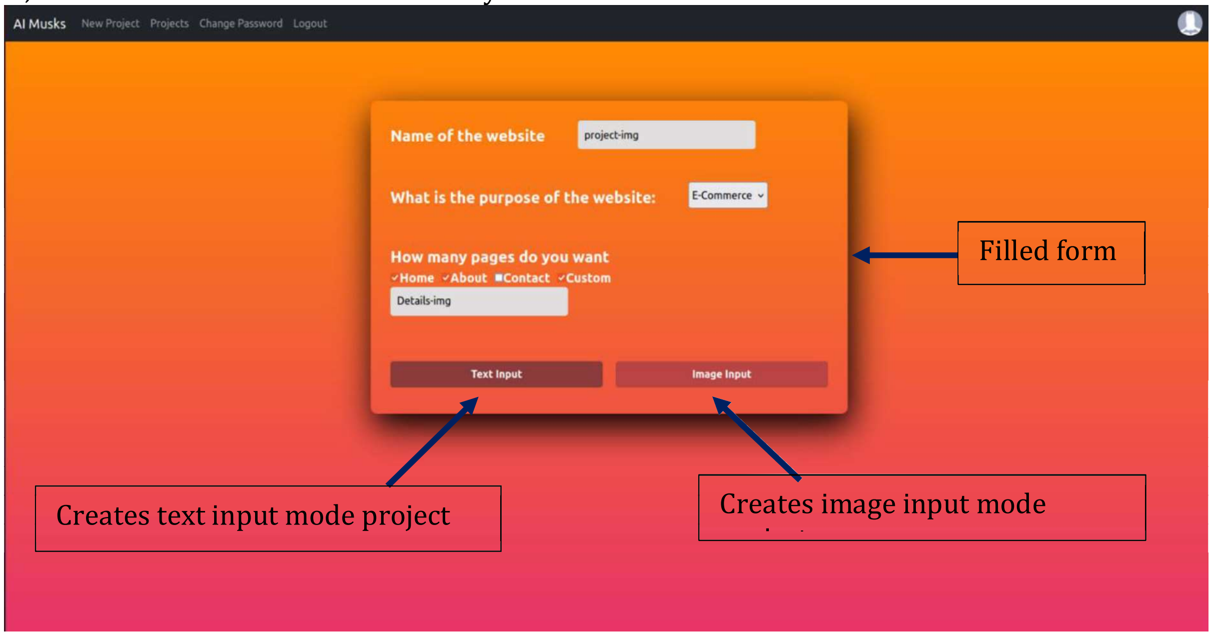
My Projects:

The "My Projects" page takes the users to the projects page, where they're shown their previously created projects



Creating New Project:

The “New Projects” page takes them to the purpose page, where they give general information about their website, user can also select which mode they want to use.



Name of the website: project-img

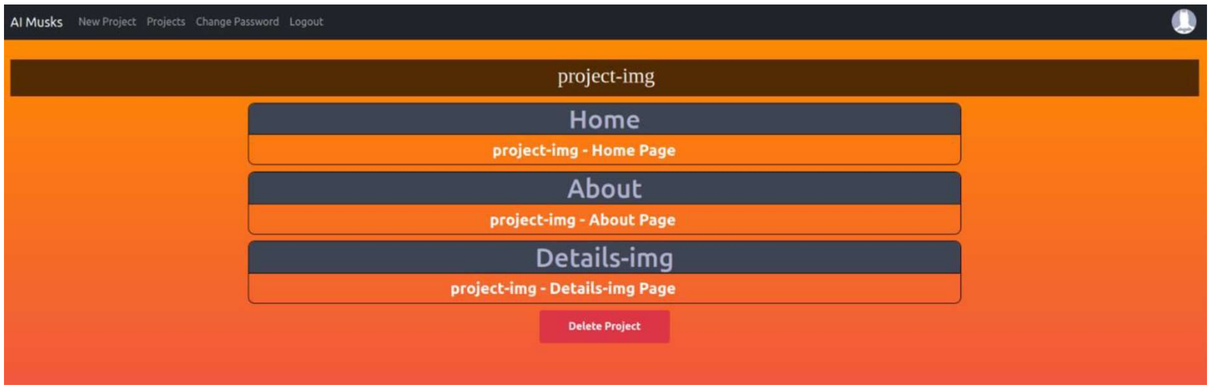
What is the purpose of the website: E-Commerce

How many pages do you want: ✓Home ✓About ■Contact ✓Custom

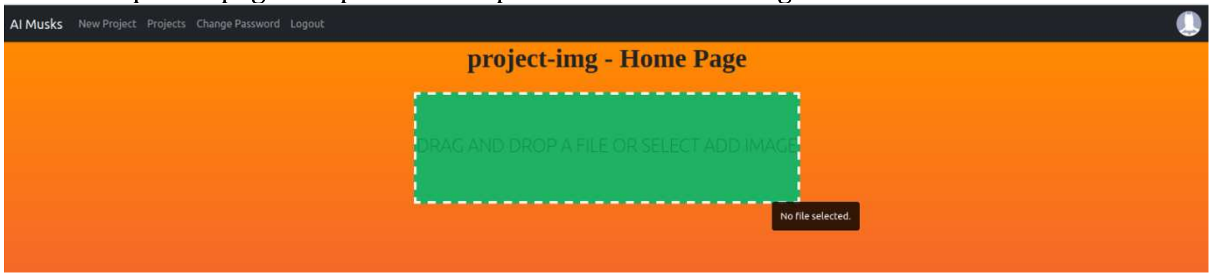
Details-img

Image Input Mode:

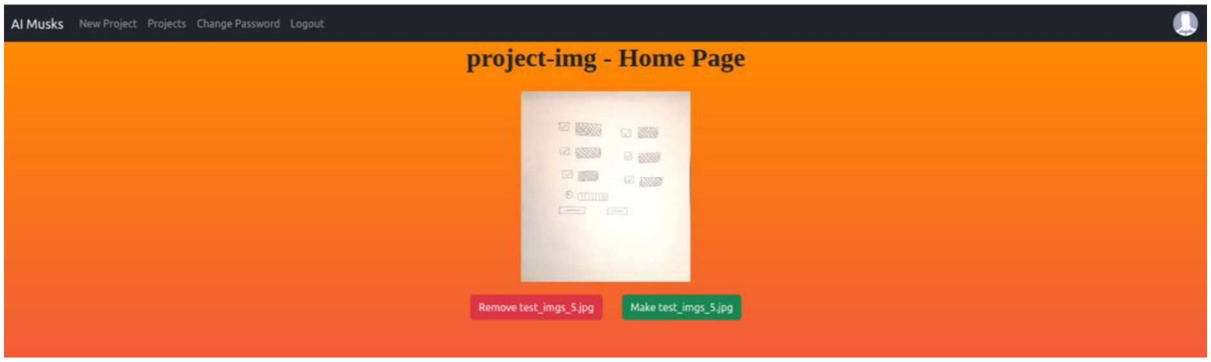
Users can use image input mode to upload hand-drawn images and generate HTML code. Once the image input mode is selected, the pages the user chose to make are listed, where they can then choose which to make first.



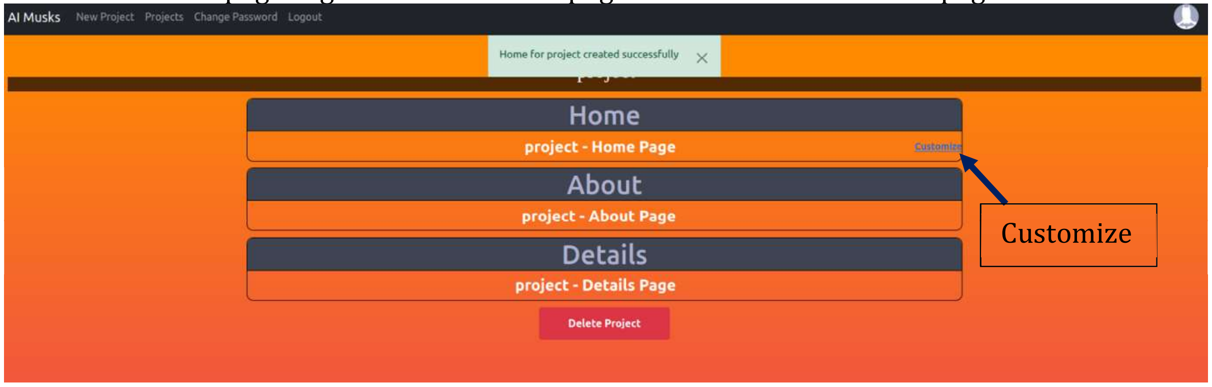
User can select specific page and proceed to upload hand drawn image.



They can then select the image they want to upload, which will be processed by the ML

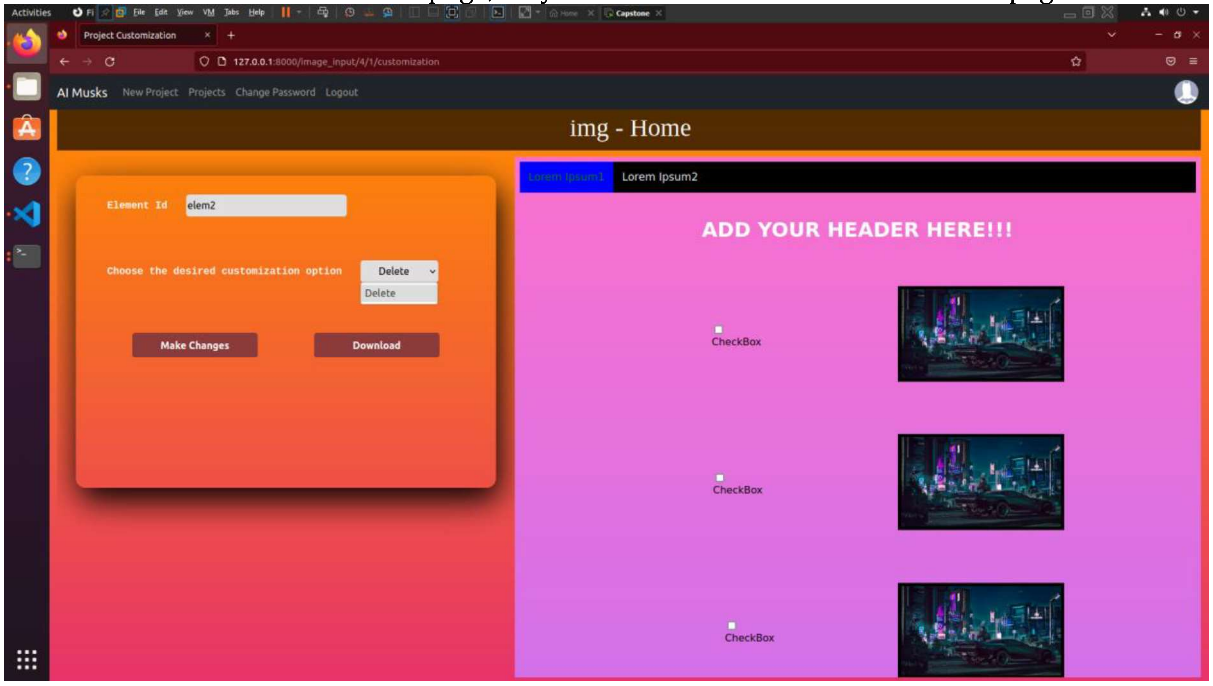


Once they choose to create the page, they are redirected back to the list of their project pages, where they can then either select a new page to generate additional pages or customize the HTML page which is made already.



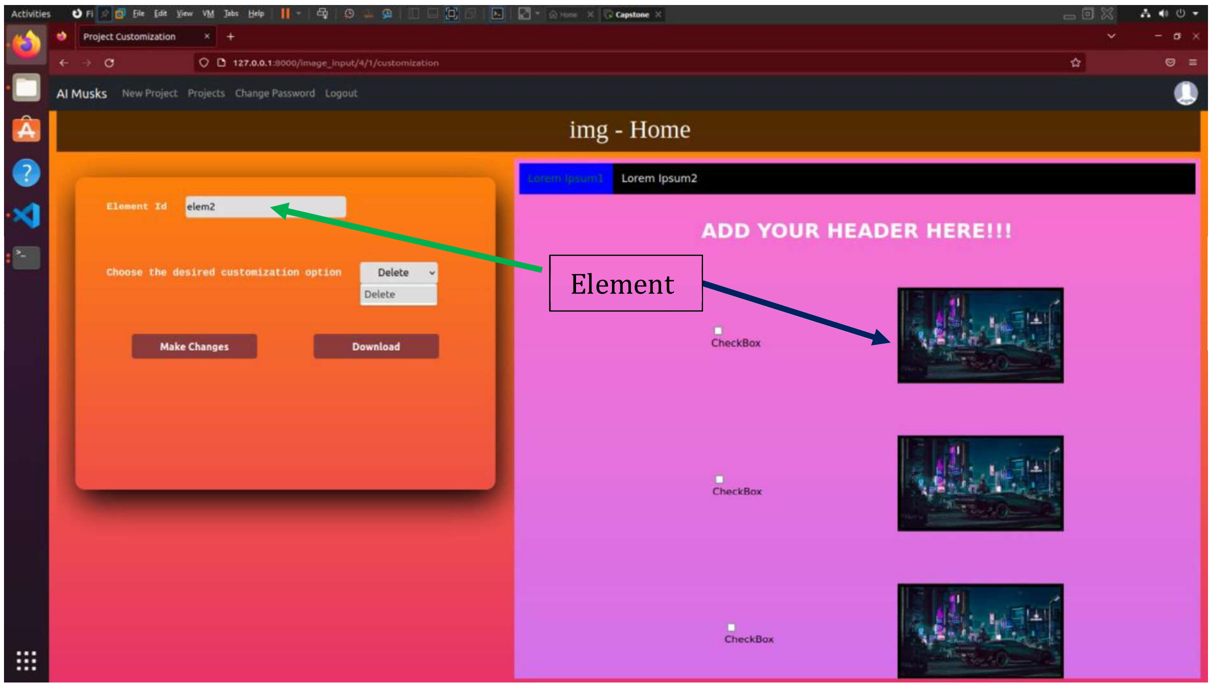
Page Customization - Image Input Mode:

Once the user selects to customize their page, they are redirected to the customization page



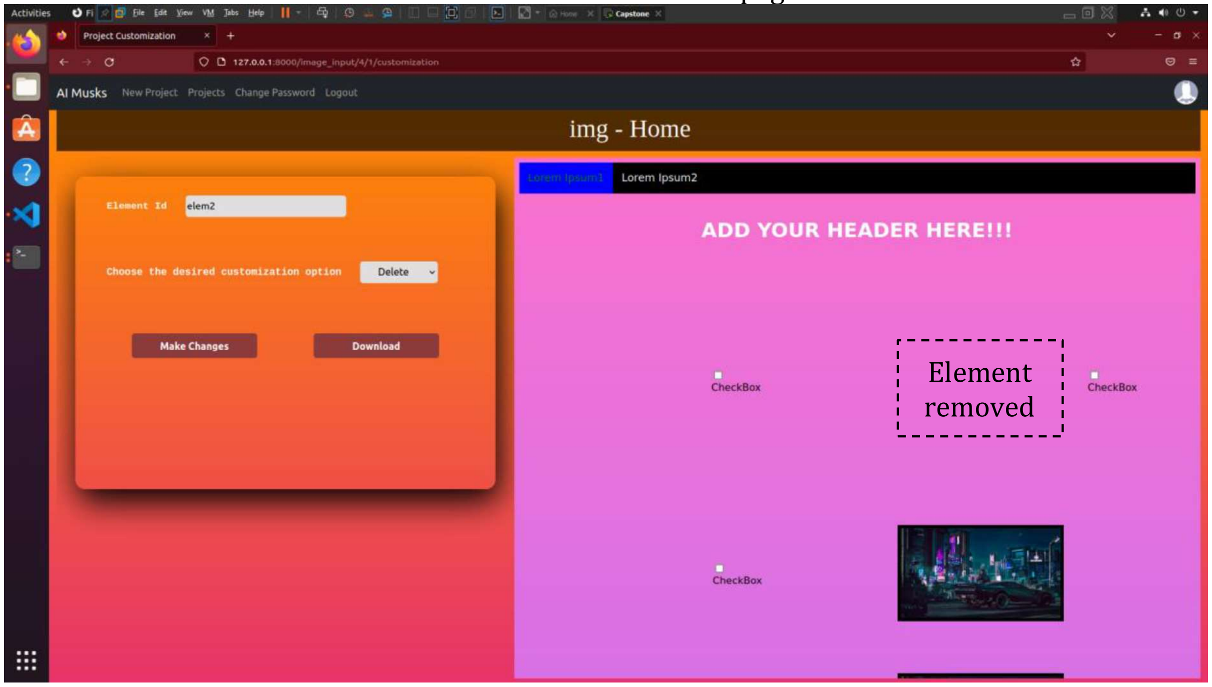
Page Customization – Element Editing:

Users can select the elements in real-time and make the changes.



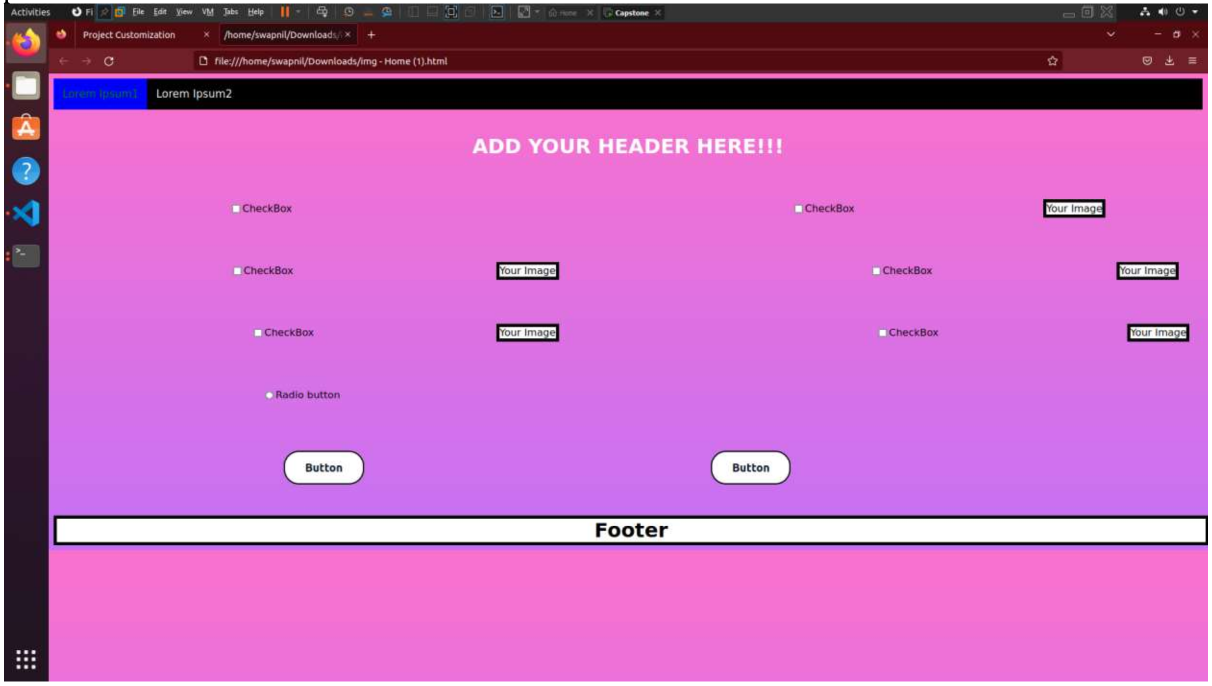
Page Customization – Page Download:

Users can click on the download button and download the HTML page.



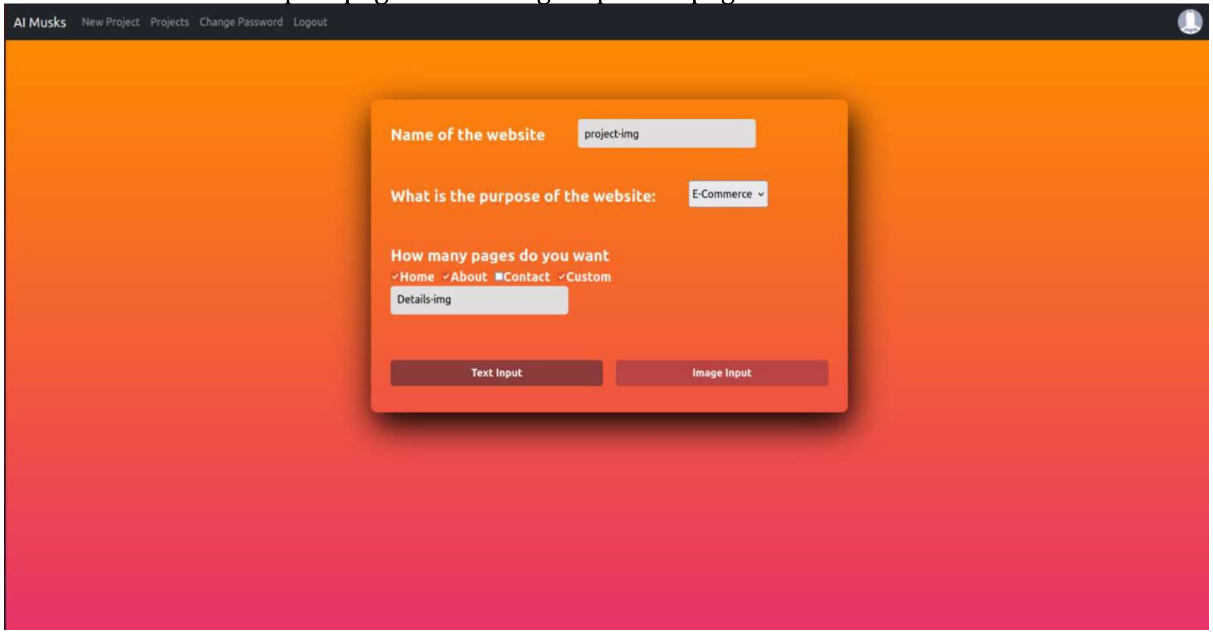
Page Customization – Post Download:

After they download their page, the default images will not show, but the alternate text will display, letting the users know that the image elements are still present on the page. The changes made during customization are made permanent



Text Input Mode:

Alternative to Image Input mode, the user can select text Input mode Once an image input project is made, it takes the user to the counterpart page as the image input list page

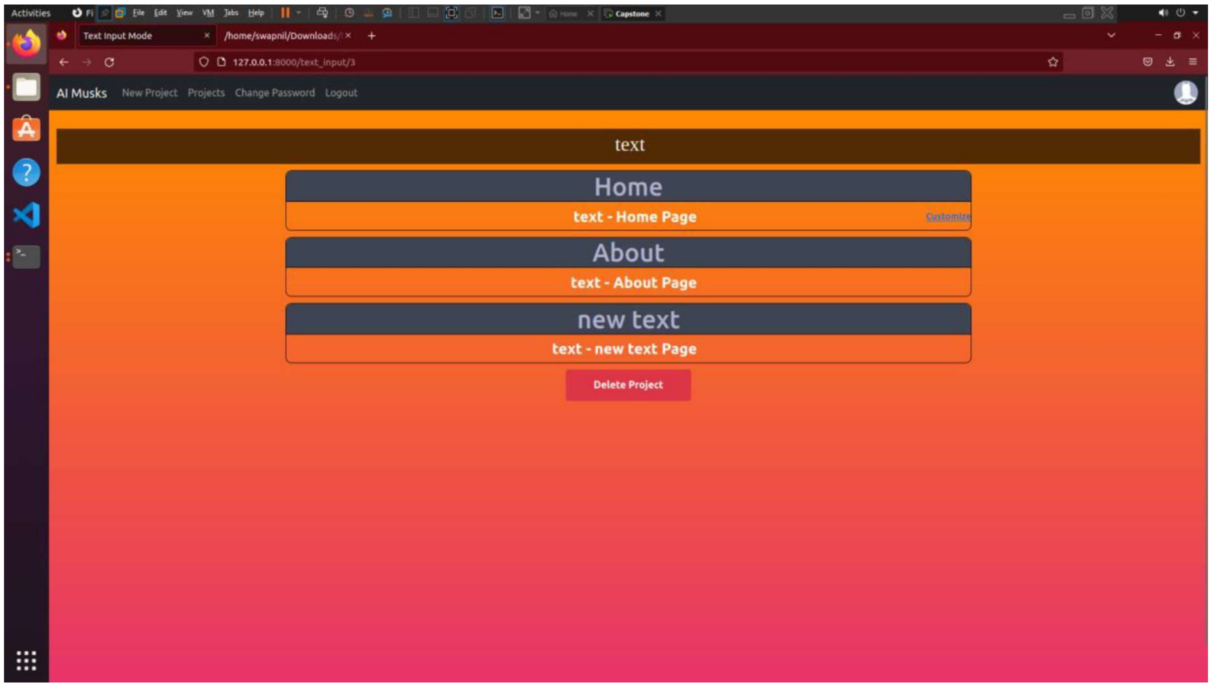


Name of the website: project-img

What is the purpose of the website: E-Commerce

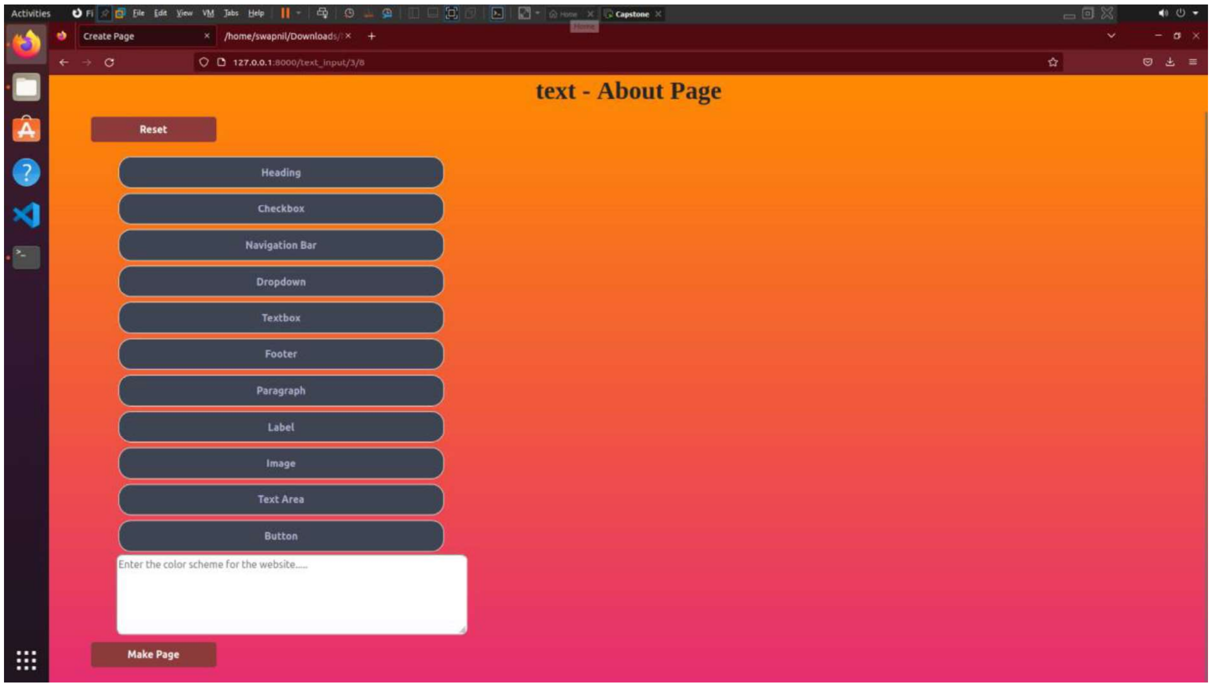
How many pages do you want
✓Home ✓About ■Contact ✓Custom
Details-img

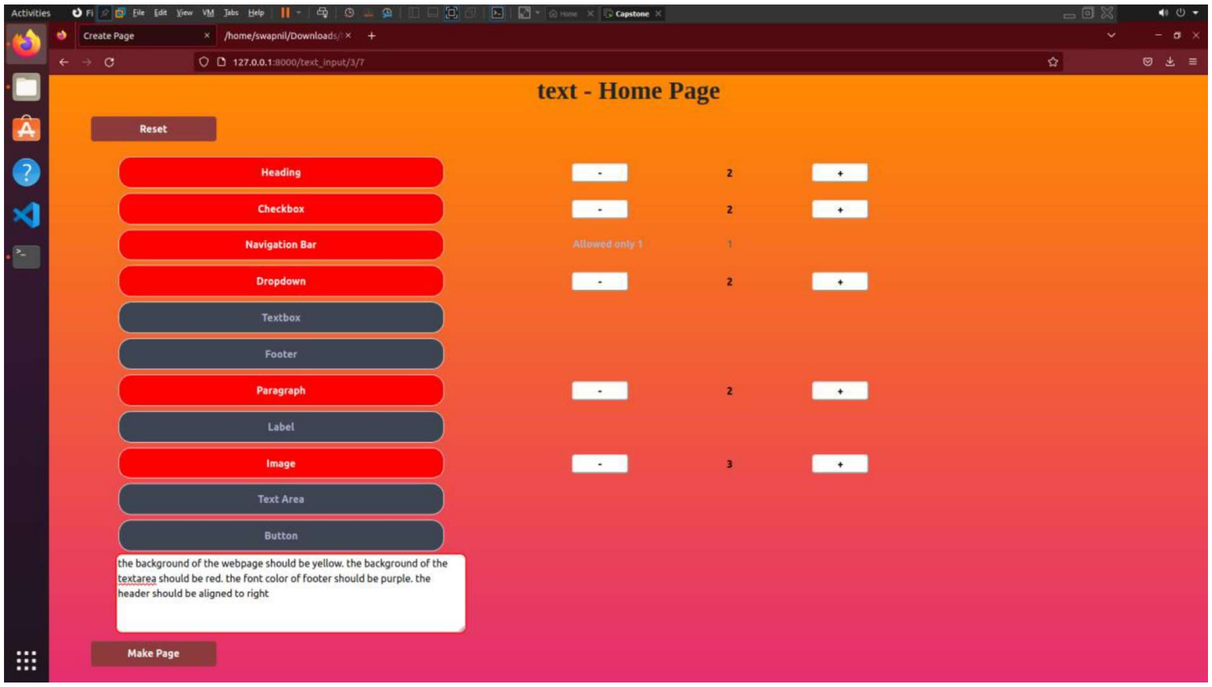
Similar to the image input mode flow of events, the user can select the page they want to create first



Text Input Mode – Element Customization:

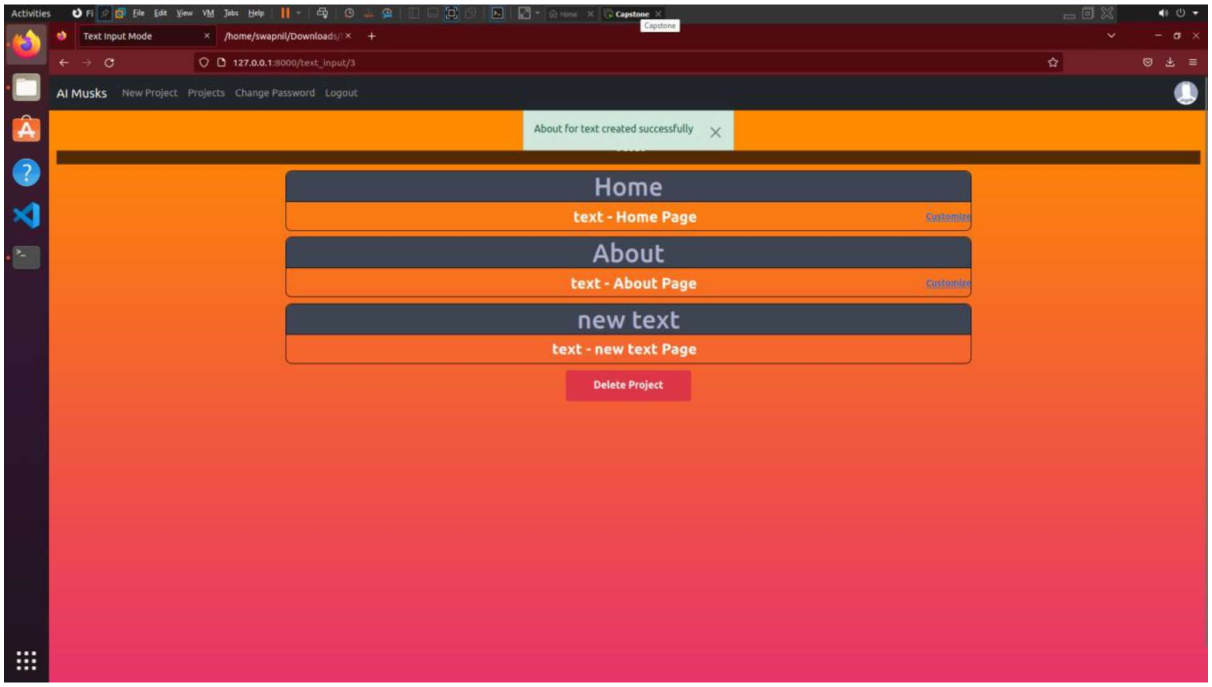
The user can then select the elements they want for their page, and choose the number of each element they want.





the background of the webpage should be yellow. the background of the textarea should be red. the font color of footer should be purple. the header should be aligned to right

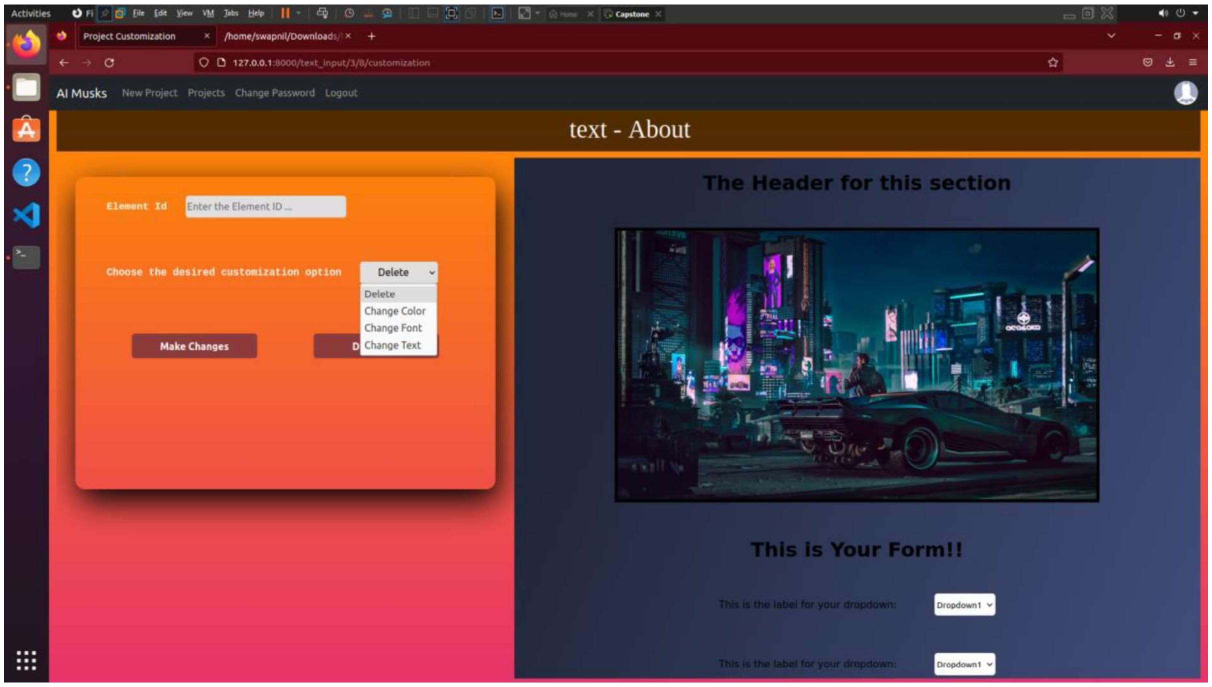
Similar to the image input mode, the user is redirected to the page list page for the project. Here they can select to go to the customization page for the page they made, or create another one

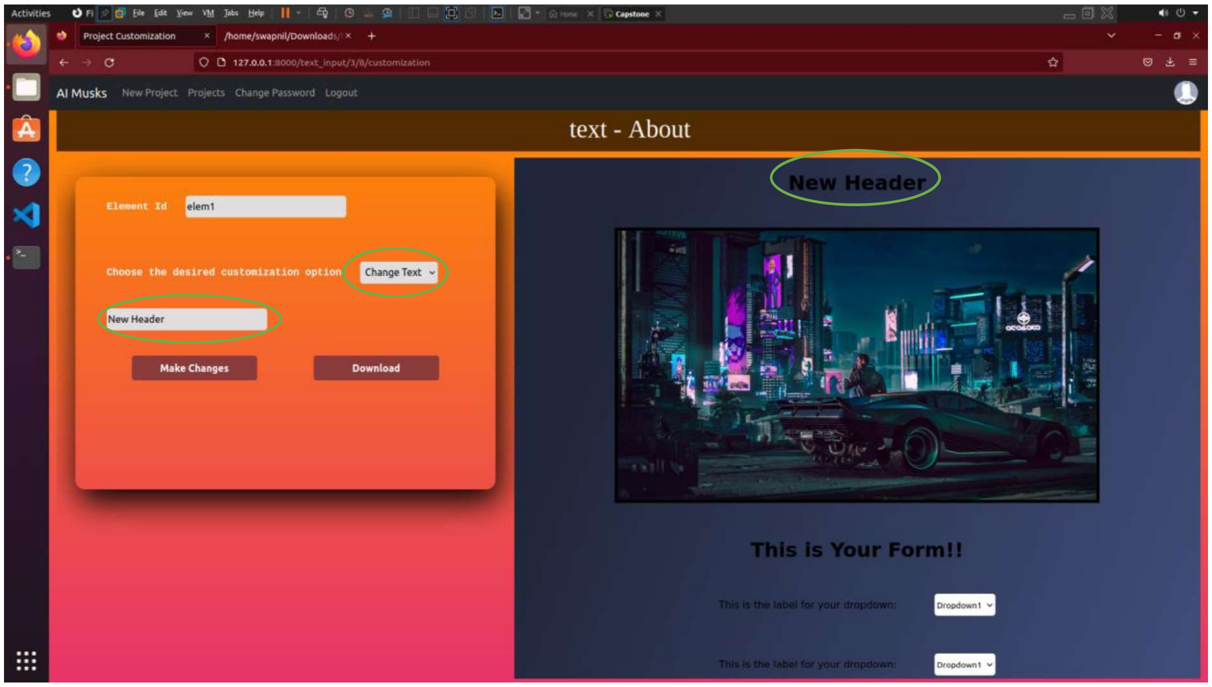


About for text created successfully

Text Input Mode – Page Customization:

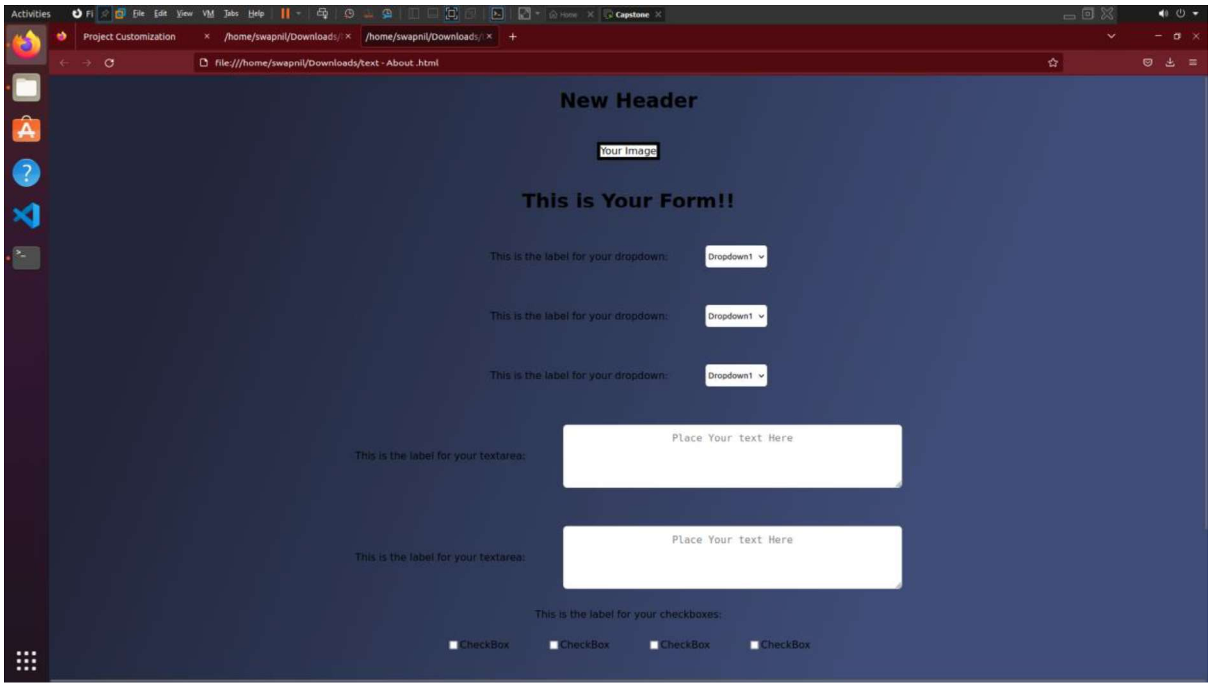
Once on the customization page, the process is consistent with the image customization page, where the page is displayed and elements can be clicked to be selected and changed





Text Input Mode – Page Download:

The page can be downloaded by clicking the download button.


file:///home/swapnil/Downloads/text - About .html
New Header
Your Image
This is Your Form!!
This is the label for your dropdown:
Dropdown1
This is the label for your dropdown:
Dropdown1
This is the label for your dropdown:
Dropdown1
This is the label for your textarea:
Place Your text Here
This is the label for your textarea:
Place Your text Here
This is the label for your checkboxes:
CheckBox
CheckBox
CheckBox
CheckBox

Meetings Report:

The meetings the group conducted throughout the whole capstone 2 project are listed below, with their times. This toggl report only includes the group meetings that took place during the sprints. For an in-depth look at the individual progress of the group members, refer to the final individual reports.

Toggl report starts from next page