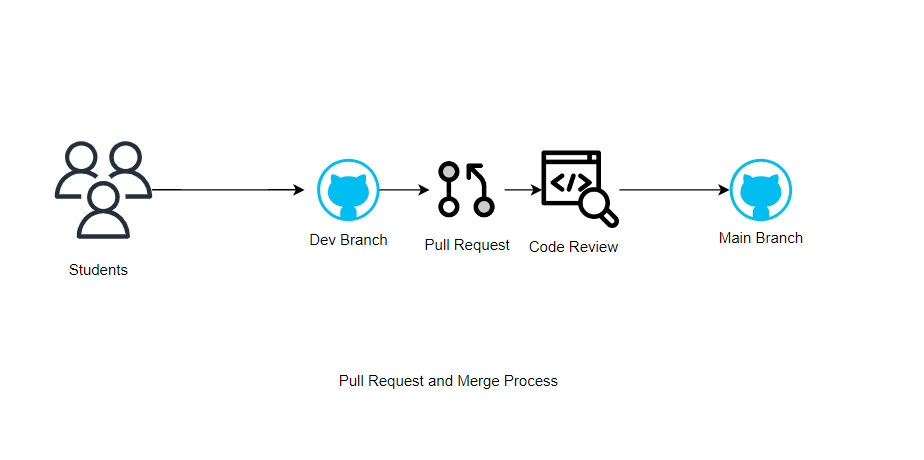
Here's a diagram showing the process where students first push their code to a branch, followed by a code review, and finally merging into the main branch. This workflow ensures that changes are reviewed before being integrated into the main project. [](https://private-user-images.githubusercontent.com/62823467/367557548-ee8569fc-baa9-4354-9fcb-97d48b8edc88.png?jwt=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..bKb07_GKWfXb4WxMFtEe1ZCV4dCc2NgsgzbxH_QsFTc)

**Question No. 1**

**Tinker Application**

This Python program creates an image classification application using the **Tkinter** library for the graphical interface and **TensorFlow** for machine learning. The app allows users to upload images, which are then classified using a pre-trained **MobileNetV2** model. Key features include image display, classification results, and the use of object-oriented programming (OOP) principles like encapsulation and method overriding. The app ensures that an image is uploaded before classification, and it outputs the top prediction along with a confidence score.

**Main Features:**

1. **BaseWindow Class**: This sets up the main application window using Tkinter.
2. **ImageLoader Class**: Handles loading and storing the uploaded image.
3. **ImageClassifierApp Class**: Inherits from both BaseWindow and ImageLoader, allowing the app to display images and classify them using the MobileNetV2 model.
4. **Decorator**: A check\_image\_loaded decorator ensures an image is uploaded before classification.
5. **Model**: The app uses **MobileNetV2**, a pre-trained model, to predict the image's class and display the top result with a confidence score.

**Steps:**

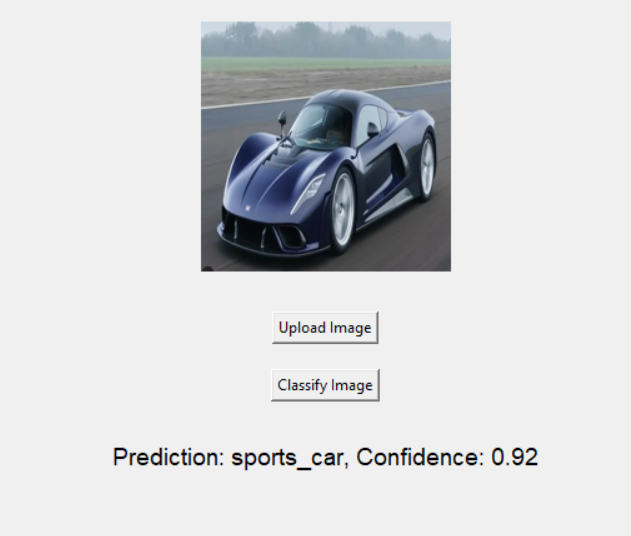
* The user uploads an image, which is resized and displayed.
* Then Classify the Image
* The image is processed, resized to 224x224 pixels, and normalized.
* The MobileNetV2 model classifies the image, and the top prediction is shown with its confidence level.

**Image 1**

Top of Form



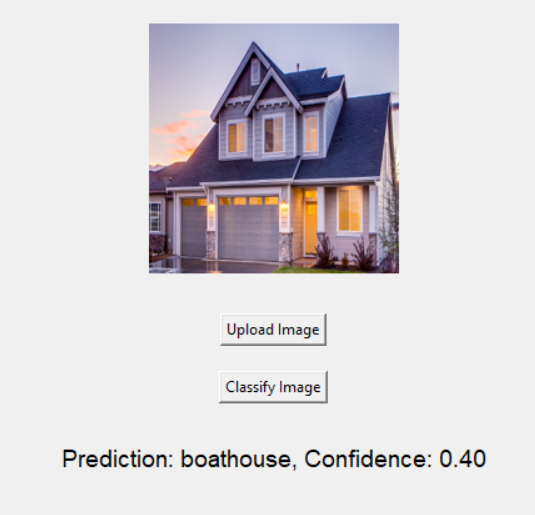
Result:



**Image 2:**



Result:



**Question 2:**

**Side Scrolling 2D Game**

**Game Overview:**

The game involves a **player character** with basic movements (run, jump, shoot projectiles), **enemies** to defeat, **collectibles**, and a camera that follows the player. The design includes **3 levels** with a **boss enemy** at the end of each. The player has health and lives, and the score is based on enemy defeats and items collected.

**Game Features Breakdown:**

**1. Player Class**

* **Attributes**:
  + **Speed** for running.
  + **Jump height** for upward movement.
  + **Health** and **lives** system.
* **Methods**:
  + Control player movements: run(), jump().
  + Player's ability to **shoot** projectiles: shoot().
  + **Take damage** and update health.

**2. Projectile Class**

* **Attributes**:
  + **Speed** and **direction** of the projectile.
  + **Damage** to be inflicted on enemies.
* **Methods**:
  + Movement of the projectile: move().
  + **Collision detection** with enemies: check\_collision().

**3. Enemy Class**

* Placeholder for enemy behavior, like **movement patterns** and **attack mechanisms**.
* **Health** and **damage-dealing** logic.
* Special **boss enemy** at the end of each level.

**4. Collectible Class**

* **Types**: Health boosts, extra lives, ammunition.
* Each collectible affects the player’s attributes (e.g., more health or ammo).

**5. Level Design**

* **3 distinct levels** with progressively harder enemies and challenges.
* Each level ends with a **boss fight**.
* The layout includes **obstacles** like water, mountains, and trees to make the environment visually appealing.

**6. Scoring System**

* Score increases based on:
  + **Enemies defeated**.
  + **Collectibles** picked up.

**7. Health Bar**

* Display both the **player’s health** and **enemy health** on the screen.

**8. Game Over Screen**

* When the player’s lives drop to zero, the game will show a **Game Over** screen with an option to **restart**.

**9. Bonus: Dynamic Camera**

* The camera should **smoothly follow** the player to keep them centered in the middle of the screen, creating a fluid gameplay experience.

**Game Controls:**

* **A** and **D** for movement left and right.
* **W** to jump.
* **Space** to shoot projectiles.

**Installation and Usage:**

1. **Clone the repository**:

<https://github.com/Bipuojha1/softwarenow_assignment3.git>

1. **Install the required dependencies**:

*pip install pygame*

1. **Run the game**:

*python main.py*

The game uses simple keys to control the character, making it user-friendly, and its dynamic elements like camera movement, smooth scrolling, and a variety of environments create an engaging experience.

Below there are Screenshots of game while we were playing.

