<u>Name</u>: CMU Rush

<u>Overview</u>:
CMU Rush is a 2D runner game developed using Python and the CMU Graphics library. The game features three distinct modes, each offering unique gameplay mechanics and challenges. Players must navigate various obstacles, collect power-ups, and aim for the highest score or survival. Each mode (Hassan Mode, Harras Mode, and Riley Mode) presents different themes and difficulty levels, providing a versatile gameplay experience.

<u>Competitive Analysis:</u>

My inspiration for this game is a game called Gotham CIty Rush which I used to play on friv.com. My game is also comparable to games such as red ball and bull run. These games feature simple controls (jumping, sliding, swinging, moving right) and a progression system based on distance traveled and obstacles avoided. However, CMU Rush differentiates itself through its unique thematic elements based on university life at CMU. For instance, obstacles like flying quizzes, attacking professors, and time management challenges are inspired by real academic pressures.

Moreover my game will also have three different game modes themed off of some of my university professors.

> **FreePlay (Hassan Mode):** A relaxed mode without time constraints, where obstacles appear at a moderate pace.
>
> **Time is Money (Harras Mode):** A timed mode where players must reach checkpoints before time runs out. Obstacles are more frequent, and players must manage time efficiently.
>
> **Challenge Mode (Riley Mode):** A difficult mode with complex obstacle patterns and faster gameplay. Players must survive as long as possible while facing challenging obstacles.

<u>Structural Plan:</u>

For now I intend to do all of my work in a single file and if required I can build separate files for certain pieces of code. However, within my main file I will have classes for my player, my obstacles as well as power ups. I intend to use the screen functionality of cmu_graphics to implement the different screens in the game.

Algorithmic Plan :

The trickiest part for me in this game would probably be designing and implementing different levels. I intend to come up with a strategy so that I do not have to write the code for each level myself. I may create a 2-D list that contains certain variable values for each level and using those variable values I can implement all the levels. Here is a step by step breakdown of my current plan to implement this :

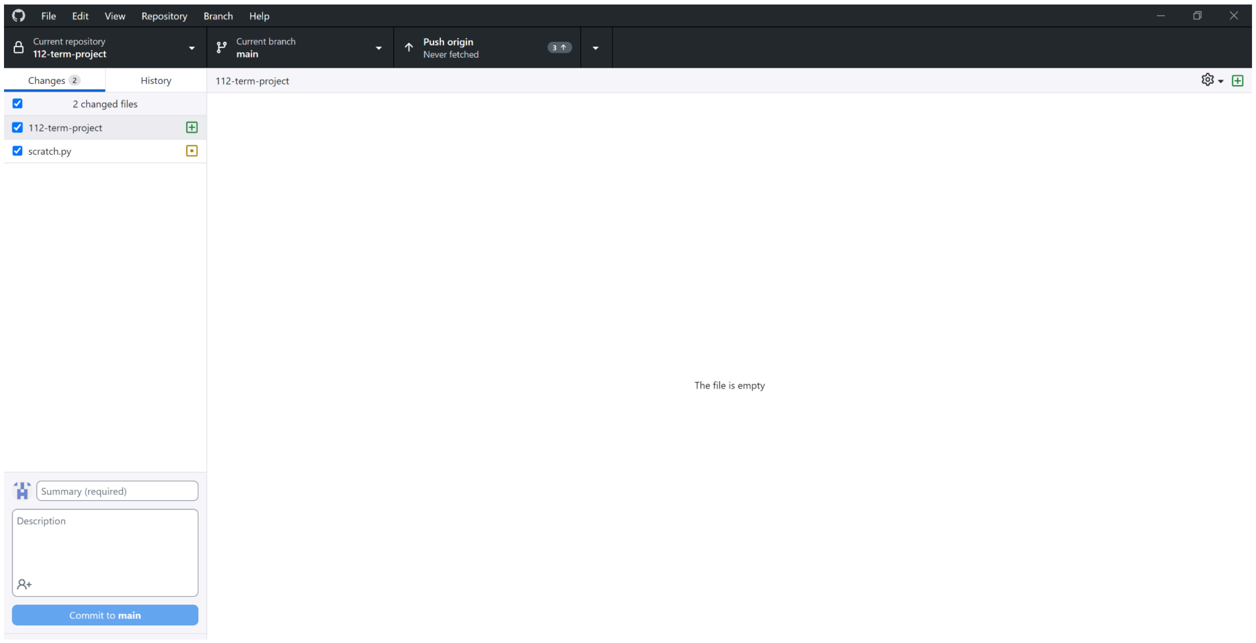
1. Define key variables such as obstacle frequency, speed, power-up frequency, level duration, background/theme, which control each level's characteristics.
2. Create a 2D list where each sublist represents one level's variable values.
3. Write a function that reads from this list and applies settings dynamically.
4. Implement my main program to adjust game mechanics based on these settings.
5. Manage transitions between levels using an index-based system.

Timeline Plan:

- By TP1, I plan to implement the basic functionality, including player movement (jumping, sliding), obstacle generation, a health bar, and the game-over condition. This will be done using simple shapes like rectangles and squares.
- By TP2 (MVP), I aim to have a fully functional game across all three modes, with sprites incorporated and at least one level for Challenge Mode. The menu should be visually decent by this stage.
- By TP3, I will focus on adding more levels, improving the graphical quality of the menus, and potentially introducing additional features such as sound effects, tutorials, or a storyline.

Version Control Plan:

For version control I am utilizing github desktop and github to commit all my changes to the repository for my 112 project on github. I have attached an image from my github desktop app below.



Module List:

I am not planning to use any other libraries other than cmu_graphics for now. Though, common modules which we have used throughout the course such as random, math etc… I will be using them.

TP2 Update :

- I have not made any major design changes

TP3 Update:

- I made some major design changes including the following :
  - I changed how I added the background and instead drew the frames myself to get smooth scrolling effects. And then animated using some infinite scrolling logic.
  - I added a new feature → Dean Trick Power Up.
  - The player can slide to kill attackers.
  - A new obstacle → boulder was used which harms the player no matter where it collides with it.
  - The hovering was also implemented here
  - The design mechanism for the gravity was also changed

- I split my game into two modes : free play and challenge mode:
The challenge mode consists of easy medium and difficult
And the levels get progressively harder
I store variables associated with it in a constant dictionary.
    - I also added sounds to which the files are in the same directory
    - The image files are also in the sam directory and the paths are now relative for both sounds and images.