

Project Title: Online Recipe Management System

All group needs to finish both projects

Project Description:

The Online Recipe Management System is a Python-based application designed to help users manage and organize their recipes. It incorporates OOP principles and various Python modules to provide a user-friendly interface for users to add, edit, search, and share their favourite recipes. The system allows users to store recipe details such as ingredients, preparation instructions, cooking time, and dietary information.

Marking Scheme:

1. Class Design (10 marks):

- Create a **Recipe** class with attributes such as title, ingredients, instructions, cooking time, and dietary information. (3 marks)
- Implement methods for setting and getting recipe details. (3 marks)
- Create a **RecipeManager** class responsible for managing recipe records. (2 marks)
- Implement methods to add a new recipe, update recipe details, delete a recipe, and display recipe information. (2 marks)

2. Recipe Management (12 marks):

- Use the **RecipeManager** class to handle the management of recipe records. (2 marks)
- Implement methods to add a new recipe, update recipe details, delete a recipe, and display recipe information. (6 marks)
- Use file handling functions to save and load recipe records. (4 marks)

3. User Interface (12 marks):

- Implement a user interface module using a library like tkinter or PyQt. (4 marks)
- Design a user-friendly interface for users to interact with the recipe management system. (5 marks)
- Integrate the user interface with the **RecipeManager** class. (3 marks)

4. Documentation and Code Quality (4 marks):

- Provide clear and concise documentation for the project, including class descriptions, method explanations, and usage instructions. (2 marks)

- Ensure the code follows PEP 8 guidelines, including proper naming conventions, indentation, and comments. (2 marks)

5. Overall Project Functionality (4 marks):

- Evaluate the overall functionality of the project, including error handling, data validation, and usability. (4 marks)

Total: 50 marks


Project Title:Maze Runner Game

The "Maze Runner Game" is an exciting Python project that incorporates the OOP (Object-Oriented Programming) principles and provides an engaging experience with different levels of difficulty.


In this game, players navigate through a maze, encountering obstacles and challenges while trying to reach the exit. The project emphasizes the use of classes, inheritance, encapsulation, and polymorphism.

Step-by-Step Tasks:


1. Create a Maze class to represent the game board.

-  Marking: 5 points
 - Design the class with attributes such as width, height, maze layout, start position, and exit position.
 - Implement methods to generate the maze, display the maze, and check if the player has reached the exit.

2. Implement an base class (ABC) called GameEntity to represent game entities.


-  Marking: 5 points
 - Define common attributes and methods that game entities will inherit.
 - Include attributes like position (x, y coordinates) and symbol (character representation).

3. Create derived classes for different game entities such as Player, Wall, and Exit.


-  Marking: 8 points
 - Inherit from the GameEntity class and define specific attributes and behaviors for each entity.
 - Player: Allow movement within the maze, track player's current position.

- Wall: Obstacles that block the player's path, preventing movement through certain positions.
- Exit: The goal for the player to reach and successfully complete the level.


4. Implement a GameEngine class to manage the game flow and interactions.

-  Marking: 10 points
 - Include methods for starting the game, handling player input, updating the game state, and checking win/lose conditions.
 - Initialize the maze, place game entities (player, walls, exit), and display the initial state.
 - Process user input for player movement and update the player's position accordingly.
 - Check for collision with walls or reaching the exit, displaying appropriate messages.
 - Provide options to play again or quit the game.

5. Add multiple levels of increasing difficulty.

-  Marking: 5 points
 - Create additional maze layouts with varying complexities, such as different sizes, more walls, or longer paths.
 - Modify the GameEngine class to support level progression and track the player's progress.

6. Optional Enhancements (Bonus):

-  Marking: Up to 7 points
 - Add additional game entities (e.g., keys, locked doors) with specific interactions.
 - Implement a scoring system based on player performance (e.g., time taken, number of moves).
 - Include visual enhancements using external libraries like Pygame.

Deadline: 24th July 2023 by 23:59