# **Introduction**

Welcome to "Epic Adventures of Mr. Peanut"! In this game, you get to be Mr. Peanut. Your job is to go on cool adventures, find solutions to puzzles, and see many exciting places. You have some special buttons to use that make playing easier and more fun. Enjoy being Mr. Peanut and have a great adventure!

# **Game Description**

Mr. Peanut’s adventures take you on a magical journey through a mysterious and magical world. It’s full of fun, humor, and surprises! You’ll be part of his big adventure, and as you go, you’ll face off against powerful enemies and uncover hidden secrets. Mr. Peanut is not only brave, but he’s also smart and has a unique sense of humor that shines through every situation. The game has special buttons to help you play better, like a healing button, a pause button, and even one to quit the game. It's easy to play and fun! If you’re looking for a fun, quirky, and engaging game that’s packed with character and heart, “Epic Adventures” is the perfect choice!

# **User Stories – Explanation of Features**

This game is designed to keep you engaged with its variety of interactive features and dynamic gameplay, inspired by the following user stories:

1. **Advertisements Before Combat**: The game will present an advertisement to the user right before a combat sequence begins, potentially as a means of monetization or to showcase in-game promotions.
2. **Game Story Introduction**: Before the game starts, the user is presented with a narrative or backstory, setting the context for the game, and enhancing the immersive experience.
3. **Diverse Textures**: The game features a variety of textures for floors and walls, adding to the visual diversity and realism of the game environment.
4. **Item Interaction**: Users can see various items within the game world and can interact with them, which may include picking them up, using them, or combining them.
5. **Main Menu Access**: Upon starting the game or pausing during gameplay, the user can access the main menu page to navigate different options such as starting a new game, loading a save, adjusting settings, etc.
6. **Character Customization**: Users can personalize their experience by changing the skins or appearances of their characters, allowing for a degree of customization.
7. **Game Over Page**: After the player's character runs out of health or meets a specified failure condition, the user can create or is presented with a "game over" page, possibly with options to retry, quit, or save the game.
8. **Player Health Visibility**: Throughout gameplay, the user can consistently see a display of the player character's health, keeping them informed of their survival status and adding to the game's strategy and tension.

Balancing action-packed sequences with strategic gameplay and user-friendly features, each aspect of the game, from the health mechanics to the interactive main menu, is thoughtfully integrated to enhance your playing experience.

User Stories

* Textured Fools and Walls
* Item Creation and Interaction

**NANDINI**

# **Feature 1 – Adding textured floors and walls in the game.**

Using a list of three possibilities in the wall texture from FrmCharater.cs form, players can select their preferred gaming style and introduce a new theme to the game with the different textured walls and floors feature. The wall feature will automatically appear in the FrmBattle.cs form and seamlessly interact with the current codebase in Frmlevel.cs as the player selects it from the list.

#### **Step-by-Step Implementation:**

**Texture Creation:**

Designs with different textures added to the Frmcharater.cs form, derived from the data for the walls and floors. Verify that the player will be able to adapt to the dimensions and art style of the game.

**Integration:**

Imported the textures into the game from the created library.

**Material Creation:**

Utilizing imported textures, create materials for the flooring and walls. To get the desired look to the game, adjust the parameters such as wall colour, size, and thickness.

**Object Arrangement:**

Use these materials on the corresponding game models that depict the walls and flooring.

**Texture Selection System:**

Texture Selection System: Using a user menu created from the Frmlevel.cs form, users can choose from a variety of wall and floor textures.

**Texture Application:**

Based on the player's selection, a mechanism was put in place to apply the chosen textures dynamically to the walls.

**Testing:**

Extensive review and gameplay were conducted to verify that the textures in the FrmBattle.cs game are displayed correctly, and that different textured walls and floors interact with one another in different gaming environments. made sure the integration with the current codebase went well.

#### **Code Implementation:**

A computer screen shot of a program code

Description automatically generated

A computer screen shot of a program code

Description automatically generated

#### **Screenshots of Game**

A screenshot of a wall texture

Description automatically generated

# **Feature 2 – Item Creation and Interaction**

The purpose of this feature is to add objects to the FrmBattle.cs game so that users can engage with created several different game options. These are items that players can pick up, drop, and use. With an option setting, this enhancement seeks to offer a novel and engaging gaming experience.

#### **Step-by-Step Implementation:**

**Item Creation:**

To design and create different items to fill the game, implement the code in the Frmlevel.cs form.

**Tasks:**

Created a variety of objects in Frmbattle.cs, such as characters, wall textures, and skins, to provide a variety of gameplay options.

**Item Integration:**

Including objects for display and interaction in the game environment Imported the item assets (models, textures, etc.) from data into the game's library. created a system that lets players in the Frmbattle.cs game pick up items.

User Interface: While playing the game, create user interfaces that show pertinent information about the items you've selected.

**Item Use/Drop**: Enable functionality for using or dropping items in the game developed in Frmlevel.code.

**User Interface:**

While playing the game, create user interfaces that show pertinent information about the items user selected.

#### **Code Implementation:**

A computer screen shot of a computer code

Description automatically generated

A screen shot of a computer screen

Description automatically generated

#### **Screenshots of Game**

A screenshot of a computer

Description automatically generated

User Stories

* Main Menu
* Character Switching System

**VAISHNAVI**

# **Feature 3 – Character Switching System**

The character's costume changes: When a player selects a character from the list of options in the Frmcharacter.cs form, the added feature presents a new theme to them while they are playing the game. The FrmBattle.cs game displays the character skin that the user has selected. It appears that Frmlevel.cs's current code works without introducing any errors.

#### **Step-by-Step Implementation:**

**Changing character skin:**

To handle skin selection, a character skin was created using Frmcharacter.cs. Every character option in Frmlevel.cs is visually represented by integrated image boxes that are imported from data in the code.

**Tasks:**

Created a system that lets players easily swap between different characters' skins. This could consist of an interface or menu where users of the FrmBattle.cs game can choose their favourite character.

**Character Variety:**

Introduced a wide variety of characters from which players could select in FrmBattle.cs. Characters in the game may differ from one another in terms of appearance, skills, or other attributes.

**Implementation and Testing:**

Created a feature that allows users to change a character's skin tone. Make sure there are no difficulties or mistakes in the Frmlevel.cs code by thoroughly testing to make sure the switching process operates smoothly.

Confirm that while playing FrmBattle.cs, every selected character loads properly and doesn't interfere with the game environment in any way.

Characterized the classes, modules, and functions that control character loading and selection in the Frmlevel.cs code, as well as the overall architecture of the system. Clear instructions or tooltips in the game interface can improve the user's comprehension of how to switch between characters without any errors in the Frmlevel.cs code.

#### **Code Implementation:**

A screenshot of a computer program

Description automatically generated

#### **Screenshots of the Game:**

A screenshot of a game

Description automatically generated

A screenshot of a video game

Description automatically generated

# **Feature 4 – Creating Main Menu**

Creating a main menu screen feature requires working on several game aspects, such as Frmlevel.cs from interface design to functional implementation. perfectly integrating without creating any errors with the current codebase.

#### **Steps for Implementation:**

**Access on Game Launch:**

Verify that the main menu screen appears as the first interface the player sees when they launch the game in Frmbattle.cs form. This might involve making it the active view by default or creating a navigational logic to get there.

**Menu Options:**

Developed Include sections on game settings in the main menu options, such as the sections on new game, quit, user FAQs, and game story. The options are visible to players as soon as they start the game. With buttons or tabs making it simple to navigate, each section should have a clear label in the game.

**Visual Design and Layout:**

Created a visually appealing layout by importing buttons and pertinent graphics from the Frmlevel.cs form data. Using suitable color schemes in the code, a clear hierarchy for menu options was created, and symbols were used to direct the player's attention throughout the game.

**Functionality:**

Every button was tested to make sure it worked and pointed to the correct screens and features. As a result, players can grasp the game with ease. added animations or transitions to improve the user's experience when navigating Frmbattle.cs's menu. Described in detail how each button in the game works, along with any associated conditional logic or event triggers in Frmlevel.cs code.

#### **Code Implementation:**

Evaluated the main menu's code in the Frmlevel.cs form to make sure it complies with best practices and coding standards. Make sure everything is optimized, properly structured, and readable. provided an explanation of the code's structure and the architecture utilized in any frameworks built using the created libraries and verified that the modifications didn't introduce any new bugs or interfered with the game's existing functionality.

A screen shot of a computer program

Description automatically generated

#### **Screenshots of Game:**

A screenshot of a game

Description automatically generated

A screenshot of a computer

Description automatically generated

User Stories

* Advertisement
* Game Story

**SAHANA**

# **Feature 5 - To view advertisement, before the combat**

According to the feature I’m here to create an advertisement before a combat in game resources, whenever Mr.Peanut attacks enemy there generates an advertisement pictiure. And there is a text box to close the advertisement after certain amount of time. The player clicks on the text box and the advertisement box closes.

#### **Step-by-Step Implementation:**

**Identify Ad spaces:**

Identify suitable spaces within the game where ads can be placed without disrupting the player experience. This could include in-game billboards, loading screens, interstitial spaces between levels, or even within the game environment.

**Choose Ad types:**

Select the types of ads that will be integrated into the game, such as static ads, dynamic ads, video ads, or a combination. Consider the nature of your game and the preferences of your target audience.

**Cofigure Ad delivery:**

Configure how ads will be delivered, such as frequency capping (limiting the number of ads a user sees within a certain timeframe), targeting options, and any other relevant settings.

**Test Ad integration:**

Test the ad integration thoroughly to ensure that ads are displaying correctly, and the user experience is not negatively impacted. Check different devices and platforms to ensure compatibility.

**Optimise Ad performance**:

Monitor the performance of ads in your game and use analytics provided by the ad platform to optimize ad placements, formats, and targeting. This helps maximize revenue and enhance the player experience.

#### **Screenshots of Game**

A computer screen shot

Description automatically generated

# **Feature 6 – Game Story**

The game story includes both back story and story that unfolds during the course of the game. Players found conflicts that drive the story forward and motivate their actions. The game story respond to the player's decisions. The story serves the gaming experience.

#### **Step-by-Step Implementation:**

**Establish Narrative goals:**

The purpose of the story is to create emotions that players want to feel, and experiences the story of the game. Create outliers that story want to convey.

**Word building:**

Consider the rules of the game technology to build the technological words including its history.

**Supporting Characters**:

Support the characters with distinct personality and roles in the game story. Plan the characters that evolve and change througout the game.

**Gameplay Integration:**

Game story integrates with gameplay mechanics and consequences.

**Testing:**

Test the games narrative elements with actual players. And gather the feedback to identify any issues.

#### **Code Implementation:**

To be implemented in **‘Frmlevel.cs’** file.

C#

//Adding Game story:

A computer screen shot of a program

Description automatically generated

#### **Screenshots of Game:**

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

# **Feature 7 – A Game Over page.**

Using Record the modifications made to the code for implementing the game over a page.Supply guidelines for adjusting the game over the message as required. Incorporate information on customizing the visual aspects of the game over a page. Clearly outline any prerequisites or special considerations related to the game over logic.

#### **Step-by-Step Implementation:**

**Texture Creation:**

Choose a color scheme and imagery that align with the game's genre and narrative. Design a visually striking yet unobtrusive background texture, incorporating subtle patterns or gradients. Experiment with fonts and styles for the game over text, ensuring readability and adding visual effects. Optimize textures for performance, iterate based on feedback, and document design decisions for future reference.

**Integration:**

Imported the textures into the game from the created library.

**Texture Selection System:**

Texture Selection System: Using a user created from the Frmlevel.cs form, users can choose from a variety of wall and floor textures.

**Testing:**

Verify the dependability and functionality of the game over page. Perform comprehensive testing to confirm the validity of the game over conditions. Validate that the restart functionality operates as planned. Rectify any identified bugs or issues encountered during the testing phase.

#### **Code Implementation:**

A computer screen shot of a black background

Description automatically generated

#### **Screenshots of Game**

A screenshot of a computer

Description automatically generated

# **Feature 8 – Players Health bar**

Establish a health bar mechanism to monitor and showcase the player's health. Create a visual representation for the health bar (such as a graphical bar or numerical display). Develop code to set up the health bar with the player's maximum health value. Define the data structure responsible for storing and dynamically updating the player's health.

#### **Step-by-Step Implementation:**

**Texture Creation:**

Create an aesthetically pleasing health bar texture with distinct visual components symbolizing various health conditions. Optimize the texture for efficient performance and ensure its smooth integration into the game's user interface.

**Integration:**

Incorporate the health bar into the game's user interface (UI) by determining its screen placement. Ensure the health bar is seamlessly integrated, visible, and does not disrupt the overall user experience.

**Texture Selection System:**

Texture Selection System: Using a user created from the Frmlevel.cs form, users can choose from a variety of wall and floor textures.

#### **Code Implementation:**

A computer screen with text

Description automatically generated

#### **Screenshots of Game**

A blue line on a brick wall

Description automatically generated