DESIGN PHASE AND PRESENTATION FORMAT (DPPF)

Prepared by: Bisual Bytes

USER STORIES

First Responder:

- *User Story:* As a player, I want a particular interaction for certain characters that could potentially lead to a different route on an ending, so that players may tend to be more alive when playing the game and won't expect a linear progression when playing.
- *Implementation:* These choices could lead to alternate endings, branching paths, or unexpected character alliances. Each interaction will trigger different flags in the game logic, which will influence later scenes or outcomes.

Second Responder:

- *User Story:* As a gamer, I want a progressive storyline with hidden events so that it is thrilling and exciting so the player would like to engage more time with the game.
- *Implementation:* Implement hidden events that are triggered by specific actions or exploration in certain areas of the game. These events could be triggered by factors like revisiting a location, obtaining certain items, or interacting with specific characters. Use scripting to ensure these events are not immediately obvious, adding layers of discovery for the player.

Third Responder:

- *User Story:* As a player, I want timed decision-making moments during key events so that I experience the thrill of urgency and the pressure of making quick choices.
- *Implementation:* Develop a system where during key narrative moments, a countdown timer appears. Players must make decisions before the timer runs out, with different outcomes depending on the decision made. This adds a layer of urgency and consequences, influencing character survival or progression.

Fourth Responder:

- *User Story:* As a player, I want an inventory system to collect and use items so that I can easily escape or unlock new areas, adding strategy to survival.
- *Implementation:* Design an inventory system that allows players to collect and manage items. Each item will have a specific purpose—whether to unlock doors, solve puzzles, or fend off enemies. The inventory system should include limited space or usage to encourage players to strategize about which items to keep or discard.

Fifth Responder:

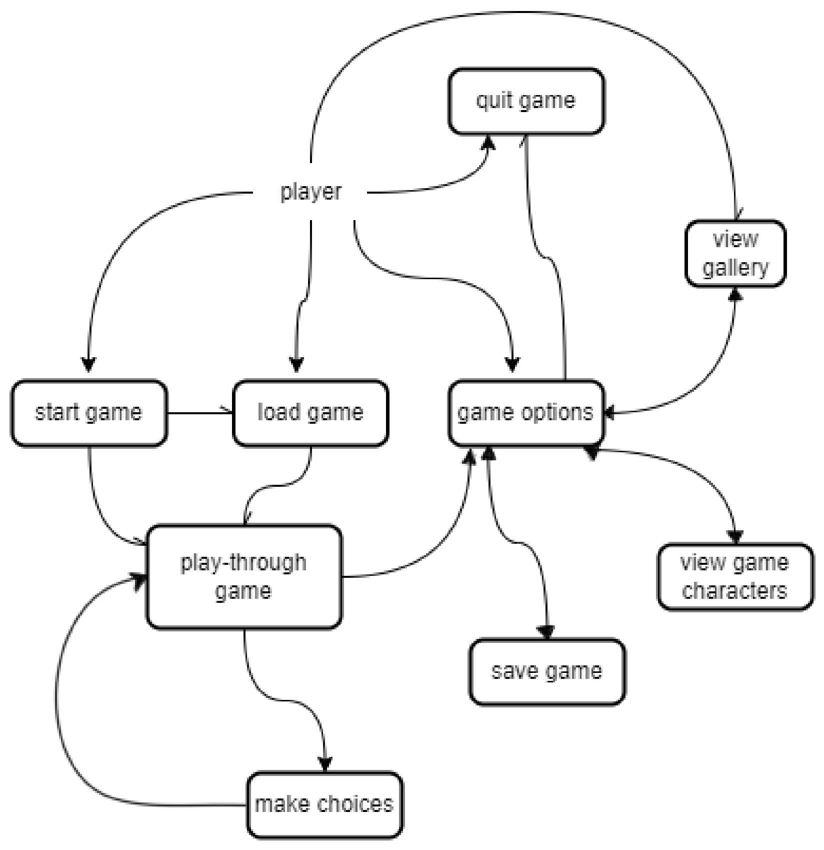
- *User Story:* As a user, I want the game to focus on the story so that I will completely enjoy the game.
- *Implementation:* Prioritize the narrative by balancing gameplay mechanics with deep, immersive storytelling. Implement frequent cutscenes, dialogue-driven interactions, and environmental storytelling to ensure the story remains at the forefront. The gameplay mechanics (like inventory or decisions) should support and enhance the narrative experience rather than distract from it.

Based on our responders, the system implementation will focus on creating an engaging, story-driven game with interactive features. A branching storyline will allow player choices and character interactions to influence the plot and lead to multiple endings. Hidden events will be placed throughout the game, encouraging exploration and adding depth to the narrative.

Timed decision-making moments will add urgency, where players must quickly choose actions that impact the outcome. An inventory system will also be included, letting players collect and use items for solving puzzles, escaping, or unlocking new areas, adding strategy to gameplay.

The focus remains on storytelling, with all gameplay mechanics supporting the narrative. The game will use a flexible engine, which is Ren'Py, with custom scripts for dialogue choices, decision-making, inventory management, and hidden events, ensuring a seamless experience for the players.

USE CASE DIAGRAM



Key Use Cases

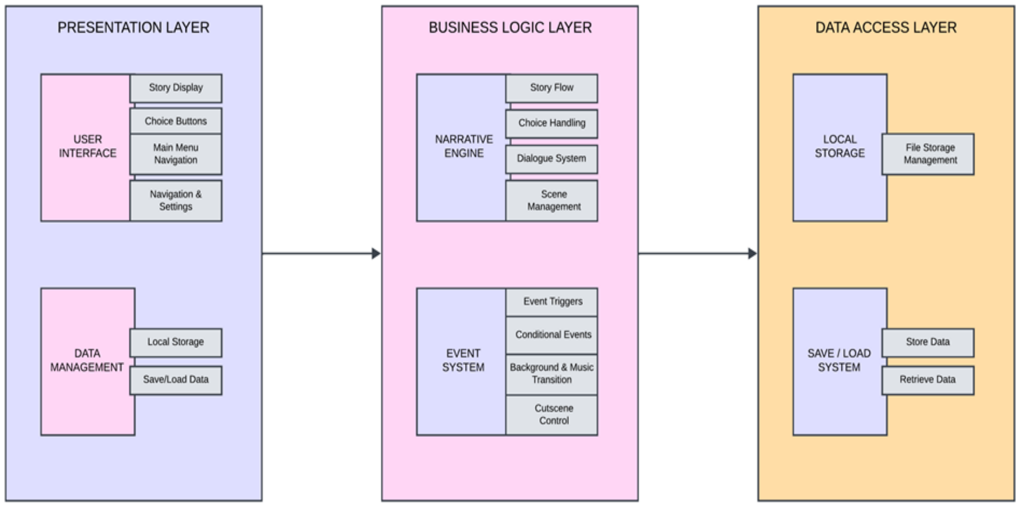
- **Start Game:** Players begin a new story from the main menu.
- **Load Game:** Players load a saved game to continue from where they left off.
- **Play-through Game: The** main gameplay is where players follow the story and make decisions.
- **Make Choices:** Players select options that affect the story and characters.
- **Game Options:** Access settings such as save or view other options during gameplay.
- **Save Game:** Players save their progress to return to it later.
- **View Game Characters:** Displays profiles of the characters encountered.
- **View Gallery:** Shows unlocked artwork images or special content from the game
- **Quit Game:** Players will exit the game.

How different users will interact with it:

- **First-time Players:** Start a new game, make choices, and save progress.

- **Returning Players:** Load previous saves and resume gameplay.
- **Completionist Players:** Explore all choices, and character details, and unlock gallery content.
- **Casual Players:** Play through the story with occasional saves.

SOFTWARE ARCHITECTURE



The Visual Novel **"Beneath the Vines"** follows a 3-Tier Architecture system. This type of architecture organizes the application into three (3) layers. These layers are the Presentation Layer, Business Logic Layer, and Data Access Layer. Each layer has specific roles and responsibilities and is designed especially for a simple student-developed project where data is managed locally without database or cloud integration.

Presentation layer

The Presentation Layer (UI Layer) is the front-end of the application. It handles all interactions between the player and the game. It also manages the display of game content, menus, and other use choices. It provides interfaces that make users interact with the story and make choices along the way.

The key components of the Presentation Layer include Story Display, Choice Buttons, Main Menu Navigation, Navigation and Settings, and more.

In User Interfaces includes:

- **Story Display:** Presents the ongoing narrative to the player.
- **Choice Buttons:** Displays choices during decision points.

- **Main Menu Navigation:** Provides access to game functions like starting a new game, loading saved data, or adjusting settings.
- **Navigation and Settings:** Handles player interaction with in-game settings and overall navigation

For Data Management:

- **Local Storage:** Manages the player's game data locally on their device.
- **Save/Load Data:** Allows players to save their progress and reload it later.

The purpose of this layer facilitate interaction between the user and the core game mechanics, providing a clean and intuitive interface for players to engage with the narrative.

Business logic layer

The business logic layer (game logic) contains the logic and core mechanics of the game. It controls how the game progresses and how choices affect the narrative. It also controls how the game events are triggered and the flow of the game. It determines the story flow, decision consequences, and game states based on the player's input.

The key components of the business logic layer include story flow, choice handling, dialogue system, and scene management—as well as Event Triggers, Conditional Events, Background and Music Transition, and Cutscene Control.

Narrative Engine:

- **Story Flow:** Controls the progression of the narrative.
- **Choice Handling:** Manages player decisions and their consequences.
- **Dialogue System:** Displays character dialogue and synchronizes it with story elements.
- **Scene Management:** Oversees transitions between scenes and environments.

Event System:

- **Event Triggers:** Determines when specific in-game events occur based on player actions or story progression.
- **Conditional Events:** Ensures events occur only under specific conditions.
- **Background & Music Transition:** Changes background visuals and music based on story context.
- **Cutscene Control:** Manages the display of animated or cinematic cutscenes.

The purpose of the Business Layer Logic is to drive the narrative progression and manage the consequences of player choices. It ensures that game events, visuals, and transitions happen based on player actions.

Data access layer

The Data Access Layer is responsible for managing the local storage of player progress. This layer stores and retrieves game data ensuring that the progress of the players can be saved and reloaded at any time.

The key components of the Data Access Layer include File Storage Management and Save / Load System.

Local Storage:

- **File Storage Management:** Organizes and stores game files on the player's local device.

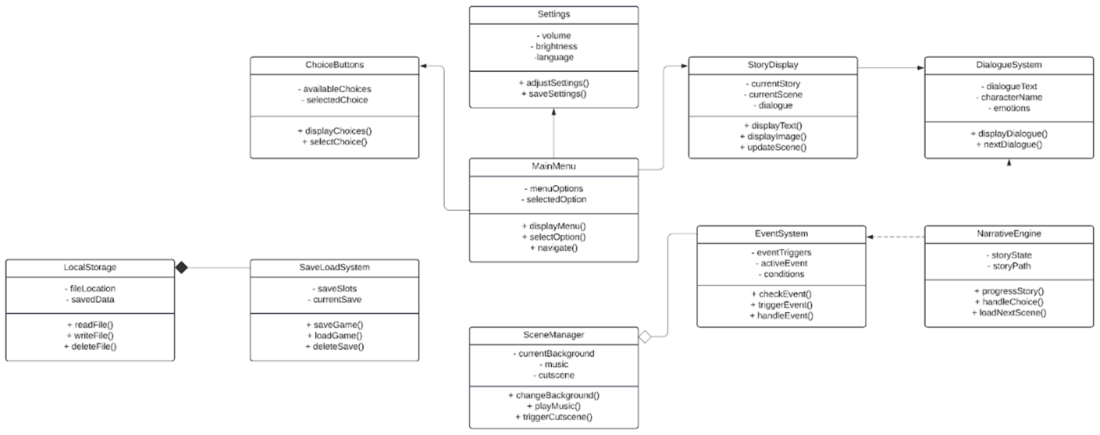
Save/Load System:

- **Store Data:** Saves player progress and game state.
- **Retrieve Data:** Loads stored data to restore the player's game state.

The purpose of the Data Management Layer provides a mechanism for saving and loading player progress locally. It ensures that players can save their game at any point and continue from where they left.

UML DIAGRAMS

1. Class Diagram



MainMenu: Manages the user interface for navigation and game options.

Relationships:

- StoryDisplay (association) - Displays story content as part of the main menu interface.
- ChoiceButtons (association) - Provides choices for player interactions in the menu.
- Settings (association) - Allows access to player preference adjustments from the menu.

StoryDisplay: Displays the current story text, images, and scenes.

Relationships:

- DialogueSystem (association) - Integrates dialogues into the story display.
- NarrativeEngine (association) - Receives updates on the story to show current scenes.

ChoiceButtons: Presents player choices at decision points in the game.

Relationships:

- NarrativeEngine (association) - Updates options based on the story's current state.

Settings: Manages player preferences like volume and brightness.

Relationships:

- MainMenu (association) - Accessed from the main menu to modify player settings.

NarrativeEngine: Controls the flow of the story and player choices.

Relationships:

- StoryDisplay (association) - Sends updates to display the current scene in the story.
- ChoiceButtons (association) - Provides the current choices to the player based on the narrative.
- EventSystem (dependency) - Triggers events based on story progression.

DialogueSystem: Manages and displays dialogues between characters.

Relationships:

- StoryDisplay (association) - Works with the display to show dialogues at the right time.

EventSystem: Manages in-game events and their triggers.

Relationships:

- NarrativeEngine (association) - Triggers events in response to narrative changes.
- SceneManager (aggregation) - Changes visuals and audio during events.

SceneManager: Handles the visual presentation of scenes and audio.

Relationships:

- EventSystem (aggregation) - Uses events to update backgrounds and soundtracks.

LocalStorage: Manages file operations for saving and loading data.

Relationships:

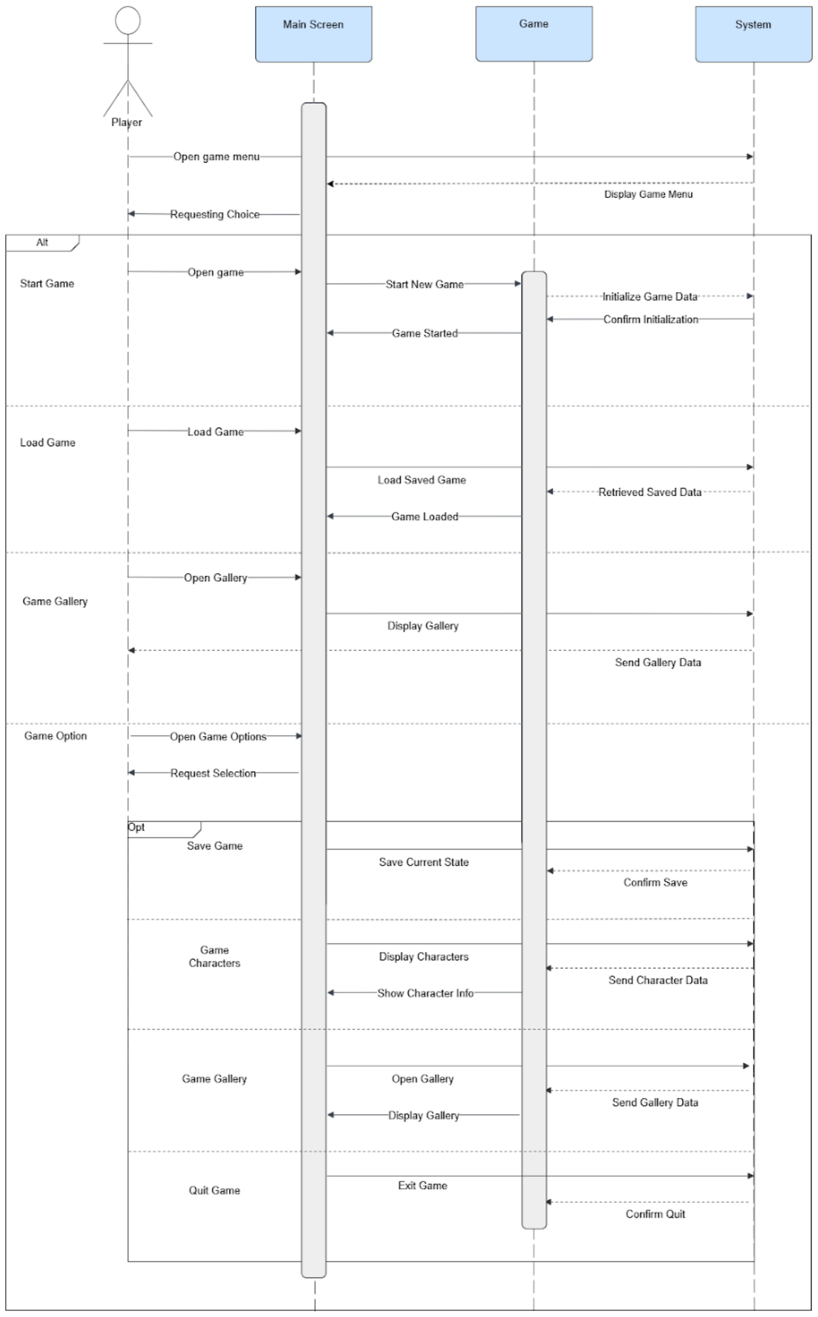
- SaveLoadSystem (association) - Provides storage functionality for saving and retrieving game data.

SaveLoadSystem: Manages save and load operations for game progress.

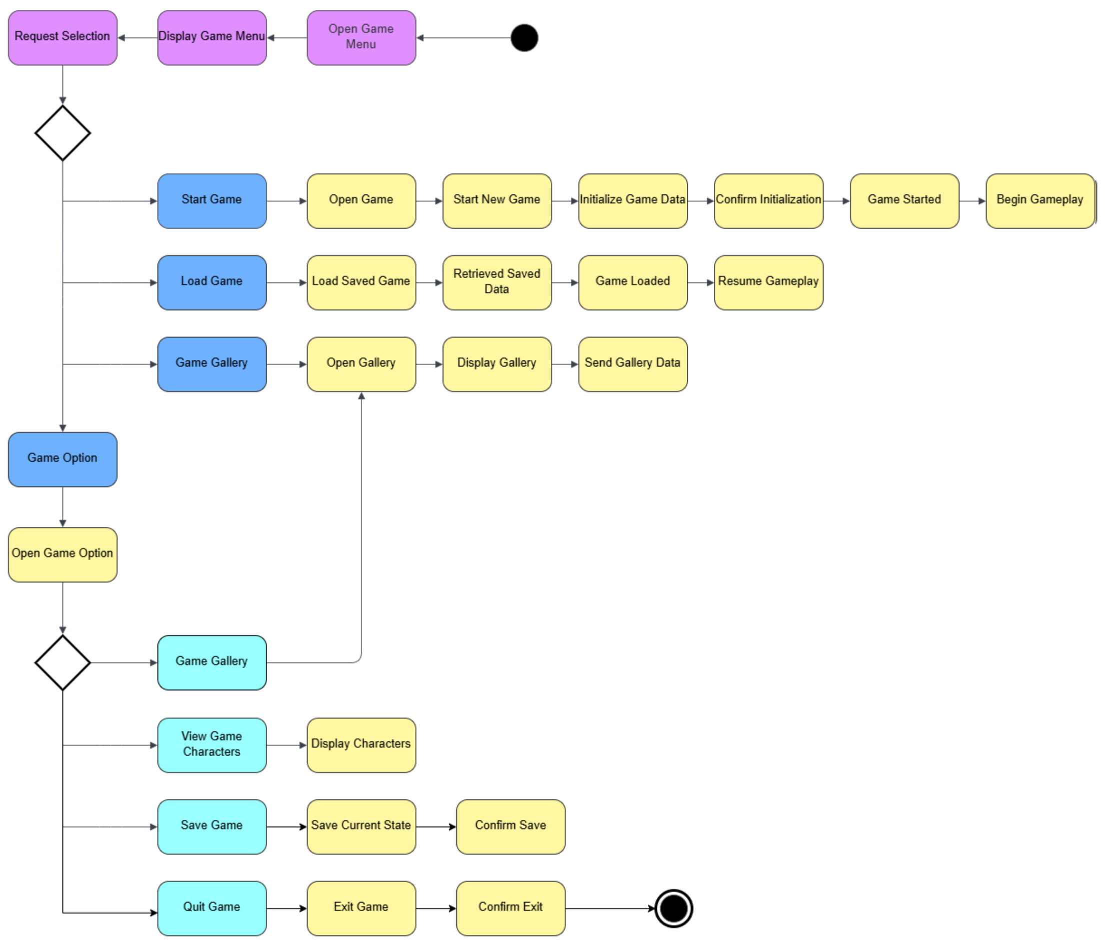
Relationships:

- LocalStorage (association) - Interacts directly to read and write saved game data

2. Sequence Diagram

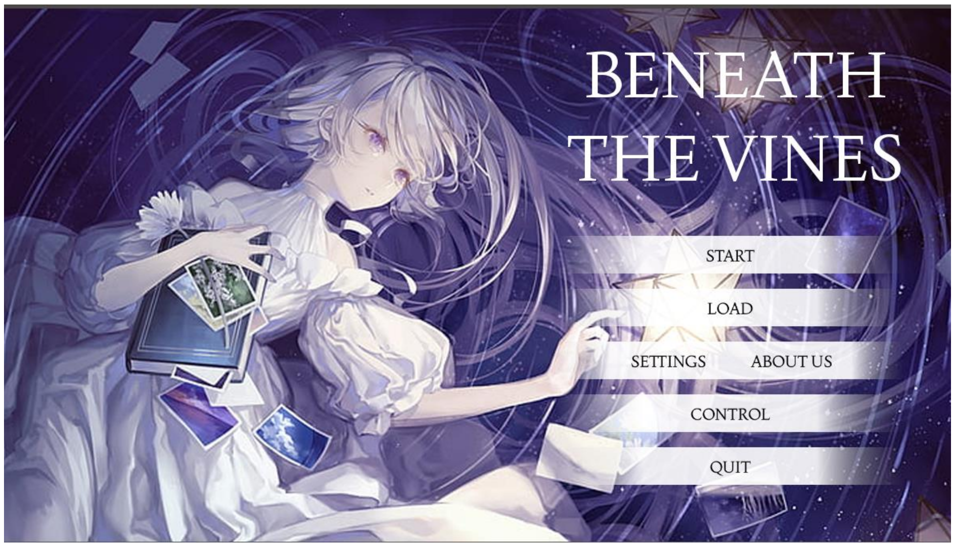


3. Activity Diagram



WIREFRAMES

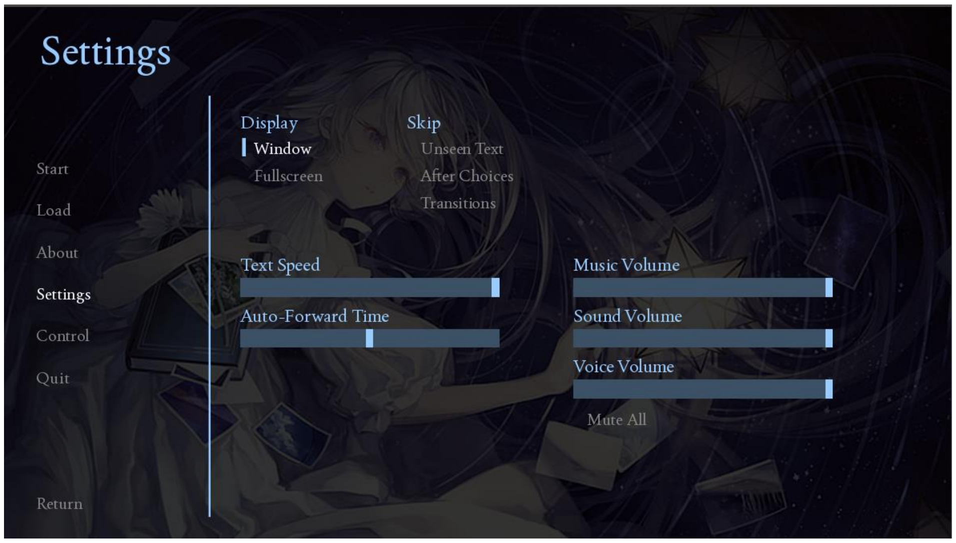
Main Menu



Layout:

- **Start:** Initiates a new game.
- **Load:** Opens a list of saved game slots to resume from.
- **Settings:** Access to audio and video.
- **About Us:** Information about the developers and production.
- **Control:** Brief guide on how to navigate the game.
- **Quit:** Exits the game.

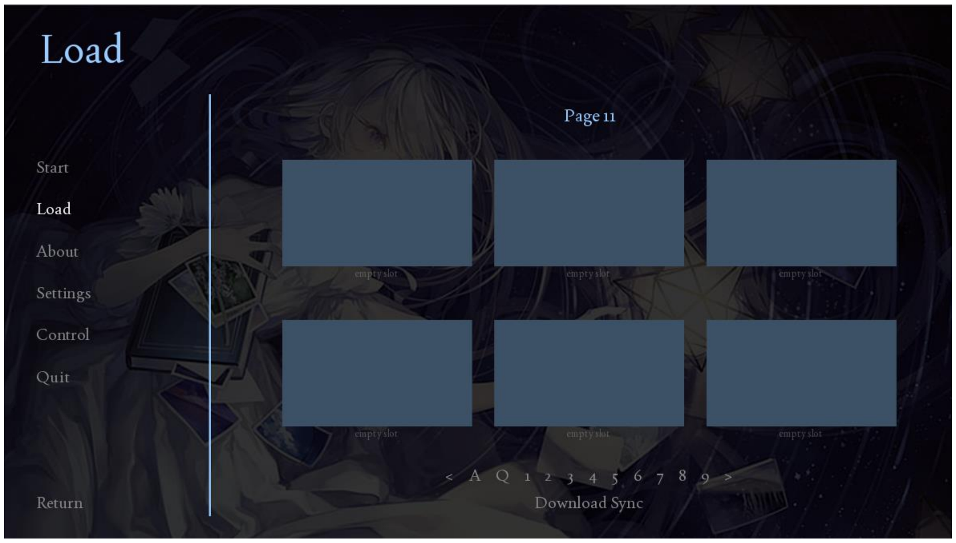
Settings Page



Layout:

- **Audio Settings:** Sliders for BGM, SFX, and Voice volume.
- **Mute option:** Video Settings:
- **Resolution selection:** Fullscreen toggle.
- **Text Speed Setting:** Slider or Toggle for Text Speed: This allows players to adjust how fast text appears during dialogue scenes.

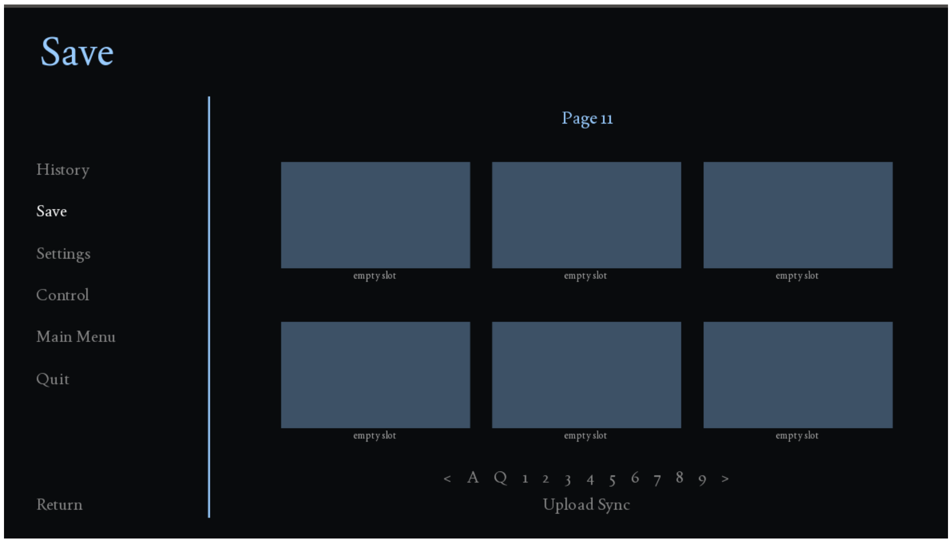
Load Screen



Layout:

- **Save Slots (Grid):** Display saved thumbnails with timestamps, organized in a grid.
- **Navigation:** A scrollable list or multiple pages.
- **Return Button:** Returns to the main menu or previous screen.

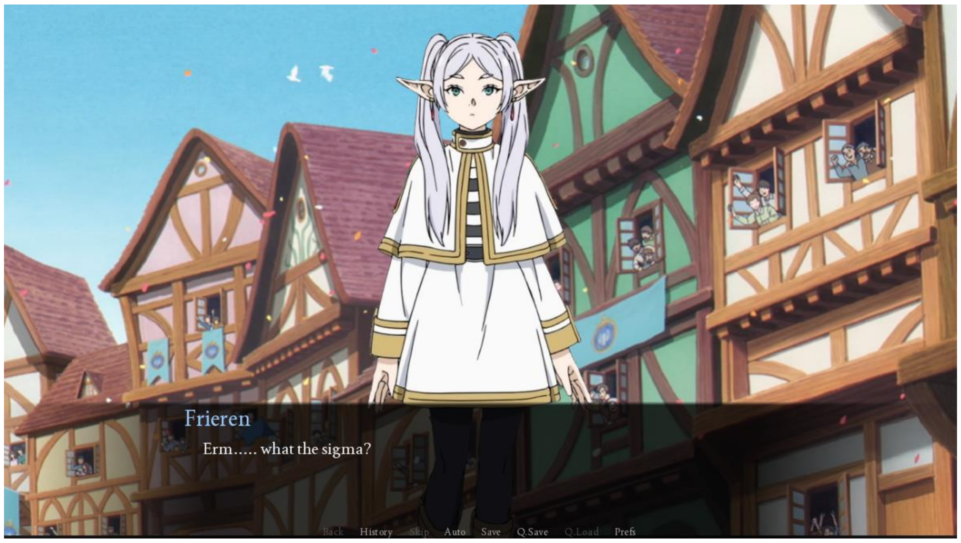
In-game Menu



Layout:

- **History:** Opens a scrollable log of previous dialogues, allowing players to review past text.
- **Save:** Allows players to save their progress in a slot, similar to the main menu's save functionality.
- **Load:** Provides access to previously saved games, so players can load a different state.
- **Settings:** Accesses in-game settings for audio, video, and text speed.
- **Main Menu:** Exits the current game and returns the player to the main menu.
- **Quit:** Exits the game entirely.
- **Return Button:** Closes the in-game menu and resumes gameplay.

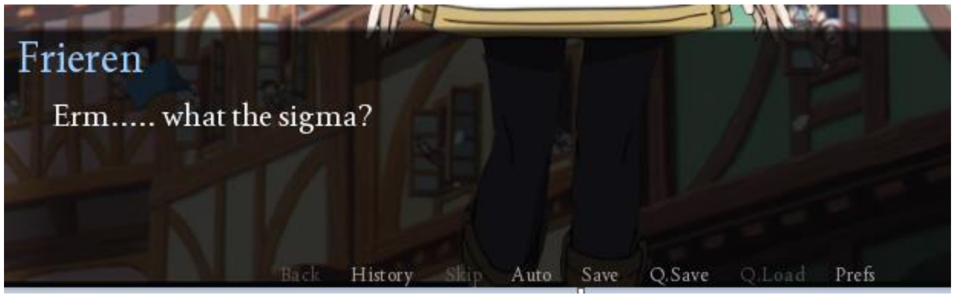
In-game Page



Layout:

- **Background Image**: The current scene of the story.
- **Character Sprites:** Displayed on top of the background image, positioned according to their role in the dialogue.
- **Dialogue Box (Bottom of the screen):**
    - <u>Text Area:</u> Shows the ongoing dialogue or narrative description.
    - <u>Character Name Label (Top or side of the dialogue box):</u> Displays the name of the speaking character.

Quick Menu (bottom of the page)



- **Back:** Rewinds to the previous dialogue line.
- **History:** Opens a scrollable dialogue history.
- **Auto:** Automatically advances dialogue at the set speed.
- **Quick Save:** Saves the current progress into a temporary slot.
- **Quick Load:** Loads the last quick save.
- **Settings:** Opens a mini-settings menu.