Nyxoria: Echoes of the Abyss

A PROJECT REPORT
for
Mini Project-II (ID201B)
Session (2024-25)

Submitted by

Devansh Batta
(202410116100058)
Anubhav
(202410116100036)
Bhavna Rajput
(202410116100049)

Submitted in partial fulfilment of the
Requirements for the Degree of

MASTER OF COMPUTER APPLICATION

Under the Supervision of
Dr. Vipin Kumar
Assistant Professor

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206

(MAY- 2025)

CERTIFICATE

Certified that **Devansh Batta <202410116100058> Anubhav <202410116100036> Bhavana Rajput <202410116100049>** has carried out the project work having **"Nxyoria: Echoes of the Abyss" (Mini Project-II, ID201B)** for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Lucknow under my supervision.

The project report embodies original work, and studies are carried out by the student himself/herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Dr. Vipin Kumar
Associate Professor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Dr. Akash Rajak
Dean
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Nxyoria
Devansh Batta, Anubhav and Bhavana Rajput

ABSTRACT

Nyxoria: Echoes of the Abyss is a 2D choice-driven action RPG developed using RPG Maker VX Ace. Set in a dark fantasy world on the brink of collapse, the game immerses players in the role of the last surviving heir of the Velmoria Dynasty. Throughout the journey, players make morally complex decisions that dynamically shape the game's storyline, character relationships, and battle mechanics. A central feature is the **corruption system**, which tracks moral choices and alters gameplay by unlocking dark powers at the cost of ally trust and narrative consequences. The project integrates narrative depth with core gameplay mechanics such as turn-based combat, event-triggered interactions, exploration, and multiple branching paths.

Designed with mid-range PC compatibility in mind, the game makes effective use of RPG Maker's eventing system, database tools, and pixel asset management to build a responsive and immersive experience. The current version focuses on a playable demo that showcases essential systems including combat, dialogue branching, lore unlocking, and player choice consequences.

While RPG Maker VX Ace serves as the current development environment, the future plan is to rebuild and expand *Nyxoria* using **JavaFX technology**, allowing for greater customization, real-time systems, and platform scalability. This project demonstrates the feasibility and vision of a scalable indie RPG.

Keywords: Dark Fantasy RPG, JavaFX, RPG Maker VX Ace, Corruption System, Moral Choice, Turn-Based Combat, Branching Narrative, Indie Game Demo, Event-Driven Logic, Game Development

ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Dr. Vipin Kumar** for her guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to **Dr. Akash Rajak**, Professor and Dean, Department of Computer Applications, for his insightful comments and administrative help on various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Devansh Batta

Anubhav

Bhavana Rajput

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 Project Description

Nyxoria: Echoes of the Abyss is a 2D dark-fantasy role-playing game (RPG) that empowers players with agency over a shattered world teetering on the brink of ruin. Designed using RPG Maker VX Ace, this project blends narrative depth with tactical turn-based gameplay, and introduces a unique Corruption System—where moral decisions influence not just the storyline but the game's mechanics and outcomes. Players assume the role of Vaelin, the last surviving heir to the fallen Velmoria Dynasty, as they journey across five mysterious regions to reclaim their legacy or succumb to darkness. The game incorporates event-driven architecture, pixel-based assets, and adaptive systems to allow branching narratives, dynamic character arcs, and impactful quest resolutions. Each gameplay decision reverberates through regional politics, alliances, enemy behaviour, and skill progression.

1.2 Project Scope

The project aims to develop a playable MVP (Minimum Viable Product) demo featuring five richly designed zones—Burning Castle, Tower Town, Hollow Vale, Elderglen Forest, and the Abyss Temple. These zones include scripted NPCs, region-specific environmental effects, optional quests, hidden lore entries, and multiple branching story paths. Functional modules include a strategic turn-based battle system, a branching dialogue tree powered by variable switches, interactive quest tracking, a corruption meter with behavioural consequences, and conditional cutscenes. The MVP also introduces scalable UI elements and optimized visual rendering for mid-range devices. In the extended vision, the project is set to transition into JavaFX technology, enabling future enhancements in real-time input handling, custom shaders, and cross-platform deployment.

The scope of the project includes the following core components:

1. **Core Gameplay** – Turn-based battle mechanics, player movement, skill usage, and combat UI.

2. **Narrative Design** – Moral choice integration, dynamic story arcs, corruption tracking, and dialogue systems.
3. **Worldbuilding** – Regional lore, NPC behaviour, map exploration triggers, hidden events.
4. **System Design** – Event scripting, enemy AI, item mechanics, and variable-driven logic.
5. **UI/UX** – Health/corruption meters, menu navigation, feedback indicators, and scalable layout.
6. **Future Migration** – Structural readiness for JavaFX conversion with modular code logic.

1.3 Project Overview

This game project takes a modular approach to storytelling and gameplay logic. Core mechanics such as input parsing, skill execution, and dialogue branching are handled using RPG Maker VX Ace's event and scripting systems. Gameplay begins with basic movement and battle commands, progressing into dialogue decisions that influence corruption, quest flow, and eventual endings. The Corruption System is a standout feature: players who choose darker options gain powerful void abilities but may lose allies or trigger hostile takeovers in towns. Gameplay also features a lore unlocking system tied to map regions, rewarding exploration and creating a sense of discovery. Although the initial version is built in RPG Maker for rapid prototyping, all systems are modular and planned for eventual migration to JavaFX, allowing for a custom engine, enhanced visual effects, and expanded interaction models.

The overview includes:

- **Protagonist Journey**: Vaelin's transformation through player choices.
- **Choice-Driven Interactions**: Every action affects corruption, power, and alliances.
- **Event-Driven Engine**: Leveraging RPG Maker's backend for dynamic decisions.
- **Replayability**: Branching endings and region-specific quest arcs.
- **Technical Forward Planning**: Preparing for JavaFX version with flexible architecture.

Chapter 2

Feasibility Study

A feasibility study is a crucial component in assessing the viability of the *Nyxoria: Echoes of the Abyss* project from a technical, economic, operational, and scheduling perspective. It evaluates whether the project can be successfully completed with the resources, tools, skills, and timelines available.

2.1 Technical Feasibility

The technical feasibility of this project is well-supported by the accessibility and robustness of RPG Maker VX Ace, which provides the foundation for developing a structured, feature-rich RPG. The engine supports tile-based mapping, real-time event scripting, combat simulation, and sprite animations without requiring complex coding. This is ideal for student and indie projects.

Technologies and Tools Used:

- **Engine**: RPG Maker VX Ace (for event-based game design, scripting, and combat mechanics)
- **Scripting**: RGSS3 (Ruby-based scripting language for advanced customization)
- **Map Editing**: RPG Maker's in-built tile editor + optional Tiled Map Editor for modular custom regions
- **Art Tools**: Aseprite, LibreSprite (pixel art for characters, tiles, effects)
- **Audio Assets**: Freesound.org and RPG Maker's default SFX library
- **Optional Export Layer**: JavaFX (planned future technology for cross-platform real-time systems)

Scalability and Expansion:

- JavaFX allows the introduction of real-time combat, custom shaders, and visual effects.
- A modular code approach ensures core features like combat and UI can be refactored efficiently.

- Assets created in RPG Maker (e.g., characters, regions, quests) can be ported with minimal rework.

Risk Mitigation:

- Use of stable and documented tools with large support communities reduces the chance of development bottlenecks.
- Regular backups and milestone testing ensure rapid bug isolation and quick resolution.

2.2 Economic Feasibility

The economic feasibility of *Nyxoria* is strong, given that the project avoids recurring costs by relying on student-developed resources and free or one-time-payment software.

Estimated Cost Breakdown:

- **RPG Maker VX Ace license**: ~$30 (one-time cost)
- **Aseprite License (optional)**: ~$20
- **Tiled Map Editor, Freesound.org**: Free
- **Marketing assets (optional trailer, art design)**: In-house
- **Testing and bug tracking**: Managed by the team using free tools (e.g., Trello, GitHub Issues)

Cost Optimization Strategies:

- Avoid external hires by skill-sharing within the team.
- Repurpose free and open-source content legally for placeholder or development phases.
- Modular build allows only partial rework during JavaFX migration, preserving previous investment.

2.3 Operational Feasibility

Operationally, the project is well-positioned for completion within its academic constraints due to the division of labor, clearly defined roles, and tool familiarity.

Team Composition:

- **Programmer**: Handles RPG Maker scripting, event logic, and JavaFX planning.
- **Game Designer**: Oversees region layout, enemy balancing, and encounter pacing.
- **Narrative Designer**: Develops lore, quest dialogue, branching narratives, and visual storytelling.

Workflow Efficiency:

- RPG Maker enables rapid prototyping without the need for separate build and compile stages.
- Events, switches, and variables provide a visual logic interface, increasing productivity.
- Built-in playtesting reduces the debugging burden.

Support & Maintenance:

- Each module (combat, dialogue, regions, corruption) is independently testable.
- Frequent internal reviews ensure feature completion stays on schedule.

2.4 Schedule Feasibility

A well-structured timeline has been created to ensure the project reaches a demo-ready state within the 8-week academic project window.

Phase Breakdown:

- **Week 1–2**: Environment prototyping, tile import, lore planning
- **Week 3–4**: Dialogue writing, corruption engine scripting, NPC creation
- **Week 5–6**: Combat system tuning, item/spell integration, boss scripting
- **Week 7**: Final region events, dynamic decisions, cutscene handling
- **Week 8**: Testing, debugging, polish, UI alignment

Contingency Planning:

- Buffer time of 3–5 days for unresolved bugs or gameplay balancing.
- JavaFX implementation begins post-demo, reusing asset pipeline and logic schema.

Chapter 3

Project Objective

The primary objective of *Nyxoria: Echoes of the Abyss* is to build a compelling and technically sound RPG demo that demonstrates the fusion of narrative complexity and strategic gameplay. The project emphasizes player-driven storytelling, immersive exploration, and the introduction of the Corruption System, which ties player choices to mechanical and narrative consequences. Below are the key objectives of the project, structured with corresponding goals and impacts:

1. Integrate Core RPG Systems

- **Goal**: To implement essential gameplay systems such as player controls, turn-based combat, skill casting, and inventory handling.

- **Impact**: Provides a polished RPG foundation that reflects tactical depth and ensures stable, intuitive gameplay.

2. Implement a Morality-Driven Corruption System

- **Goal**: To design a corruption meter that evolves based on player decisions, affecting relationships, abilities, and event triggers.

- **Impact**: Introduces consequence-driven mechanics that enhance narrative replay ability and strategic decision-making.

3. Enable Interactive and Branching Storytelling

- **Goal**: To use variables and conditional branches to offer branching paths, character reactions, and multiple endings.

- **Impact**: Empowers the player to shape the storyline and engage in immersive, evolving narratives.

4. Design Five Thematic and Explorable Regions

- **Goal**: To construct regions like Burning Castle and Elderglen Forest, filled with quests, lore scrolls, secrets, and region-specific effects.

- **Impact**: Enhances the sense of world-building and supports a diverse range of interactions and discoveries.

5. Develop Modular and Scalable Architecture

- **Goal**: To create isolated systems (combat, dialogue, exploration) that can be adapted for the planned JavaFX version.

- **Impact**: Ensures future scalability and faster migration to a more advanced platform.

6. Ensure Responsive UI/UX and Cross-Device Optimization

- **Goal**: To implement HUD elements, menus, and indicators that are clear, minimalistic, and adaptive to varying resolutions.

- **Impact**: Improves accessibility and visual clarity, increasing overall player satisfaction.

7. Establish Foundation for JavaFX Expansion

- **Goal**: To document system logic and establish feature parity for future real-time and graphical enhancements in JavaFX.

- **Impact**: Future-proofs the game for continued development beyond the scope of RPG Maker VX Ace.

8. Foster Player Engagement with Lore and Exploration Incentives

- **Goal**: To include lore entries, murals, hidden items, and vision cutscenes triggered through exploration.
- **Impact**: Deepens immersion and encourages investigative gameplay.

9. Enable Custom Quest and Event Systems

- **Goal**: To build side quests with conditional outcomes and align them with character arcs and faction tensions.
- **Impact**: Adds depth to game content and narrative flexibility through parallel storylines.

10. Create a Demo That Demonstrates Full Game Vision

- **Goal**: To present a high-quality demo that includes all major systems, a boss fight, narrative branches, and region mechanics.
- **Impact**: Serves as a tangible prototype to guide the full-scale development roadmap.

Chapter 4

Hardware and Software Requirement

To ensure the smooth development and operation of *Nyxoria: Echoes of the Abyss*, the following hardware and software requirements have been identified. These specifications are categorized based on the needs of both developers and end-users, ensuring that the game runs efficiently across a variety of systems.

4.1 Hardware Requirements

Development System Requirements:

Component	Minimum Specification	Recommended Specification
CPU	Intel Core i3 / AMD Ryzen 3	Intel Core i7 / AMD Ryzen 5 or higher
RAM	4 GB	8 GB or higher
GPU	Integrated Graphics	Dedicated GPU (GTX 1050 / RX 560 or higher)
Storage	1 GB free space	SSD with 10 GB free space
Display	1280x720 resolution	1920x1080 (Full HD) or higher
OS	Windows 10 or later	Windows 11 or macOS/Linux with Wine

End-User System Requirements:

Component	Minimum Specification	Recommended Specification
CPU	Dual-Core Processor	Quad-Core or higher
RAM	2–4 GB	8 GB
GPU	Integrated Graphics	Entry-level gaming GPU
Storage	1 GB available	SSD preferred for faster load times
Input Devices	Keyboard and Mouse	Keyboard, Mouse, and Gamepad (Optional)

4.2 Software Requirements

Development Software Stack:

Software	Purpose
RPG Maker VX Ace	Primary game development engine with scripting support
RGSS3 (Ruby)	Custom event logic and scripting
Aseprite / LibreSprite	Pixel art creation for tiles, characters, and UI assets
Tiled Map Editor	Advanced map structuring and modular tile usage (optional)
Audacity	Editing and converting sound effects (optional)
Git/GitHub	Version control and team collaboration
Trello / Notion	Task tracking and project management

Runtime Software for Players:

Software	Function
RPG Maker RTP (RunTime Package)	Required to run the VX Ace-based executable
DirectX or OpenGL	Graphics rendering support (usually built-in)
Java Runtime (for JavaFX build)	Required in future versions

4.3 Optional Tools and Assets

- **Freesound.org**: Free repository for sound effects
- **Itch.io Tools**: For future demo distribution and analytics
- **ScreenToGif**: For recording game progress or UI previews
- **FontForge**: To customize UI fonts if needed for accessibility

By selecting lightweight yet capable tools, the development team ensures efficient resource utilization while preserving the scalability needed for the game's future migration to JavaFX.

Chapter 5

Project Flow

The Nyxoria game follows a structured flow to ensure systematic development and delivery. The flow consists of multiple phases, starting from project initiation to maintenance. Below is the detailed project flow:

5.1 Project Initiation

Requirement Gathering

- Define the narrative structure focusing on themes of justice vs. corruption.
- Identify core gameplay systems: player controls, combat, morality, quest tracking, and dialogue systems.
- Detail user needs: immersive lore, intuitive mechanics, multiple endings.
- Set system goals: turn-based combat, dynamic NPCs, scalable performance.

Feasibility Study

- **Technical Feasibility**: Confirm RPG Maker VX Ace supports required features; assess potential with JavaFX for future development.
- **Economic Feasibility**: Use free or open-source tools (Aseprite, Tiled, Freesound) and cost-efficient publishing on itch.io.
- **Schedule Feasibility**: Plan MVP completion in 8 weeks with later enhancements.

Project Scope Definition

- Deliver a functional MVP with:
    - 5 unique regions
    - A branching narrative
    - A corruption/morality mechanic
    - Core UI and save/load functionality
- Define boundaries (e.g., voice acting and advanced animation to be added post-release).

5.2 System Design

Architecture Design

- Use modular design in RPG Maker VX Ace (events, tiles, scripts).
- Plan migration to JavaFX using MVC pattern and object-oriented design.

UI/UX Design

- Create mock-ups of:
    - Main Menu
    - Dialogue Box
    - Combat UI
    - HUD Elements (corruption meter, health bar)
- Ensure accessibility (font readability, contrast).

Database Design

- RPG Maker VX Ace internal database:
    - Characters, Skills, Items, Enemies, Troops, States
- Plan future SQL database for JavaFX:
    - Tables: Players, NPCs, Skills, Quests, Regions, Inventory

Event & Script Planning

- Use switches and variables for tracking decisions and quest states.
- Use common events for cutscenes, skill cooldowns, and item use.
- Plan custom RGSS3 scripts for:
    - Advanced combat behaviours
    - Corruption system logic

5.3 Development Phase

Core Gameplay Development

- **Player Controls**: Implement via VX Ace's default movement with adjustments for terrain and interaction.
- **Combat System**: Skills like Void Slash, Purge, Celestial Chains—linked to player morality.
- **Corruption System**:
    - Levels: Pure, Neutral, Corrupted
    - Impacts: Skill tree changes, NPC trust, story divergence

Dialogue & Choice System

- Dialogue branches tied to switches and conditional branches
- Variable tracking for corruption impact
- Create influence system for major story paths

World Design & Exploration

- Regions: Burning Castle, Tower Town, Sunken Halls, Pale Grove, Twilight Citadel
- Environmental storytelling via custom tiles, lorebooks, and hidden dialogue
- Map transfer events for fast travel and region navigation

UI/HUD Integration

- Custom menu for inventory, status, quest log
- Real-time updates to HUD elements (e.g., corruption meter)

5.4 Testing Phase

Unit Testing

- Test movement, event triggers, and skill effects
- Validate save/load system integrity

Integration Testing

- Verify interlinked events and variables across maps and modules
- Test battle outcomes affecting story events

User Acceptance Testing (UAT)

- Internal playtesting for narrative pacing
- External testers to provide feedback on decision impact

Performance Testing

- Test on target hardware setups
- Monitor memory usage and frame rate during intense battles

Bug Fixes

- Categorize by severity (critical, moderate, cosmetic)
- Maintain a bug tracker (e.g., Trello or GitHub Projects)

5.5 Deployment

Initial Deployment

- Export and encrypt VX Ace game data
- Upload to itch.io with build versioning

Future Migration & Release

- Start JavaFX rebuild:
    - Recreate player controller, battle system, and dialogue engine
    - Use FXML for UI and Scene Builder for layout
- Prepare for Steam publishing: achievements, cloud saves

Final Launch

- Create patch notes and update log

- Prepare launch trailer and gameplay video

5.6 Maintenance and Updates

Monitoring

- Check feedback on itch.io, social platforms, and forums
- Use Google Forms or Discord bots to collect bug reports

Bug Resolution

- Address critical bugs with immediate hotfixes
- Schedule minor patches weekly or bi-weekly

Feature Enhancements

- Post-launch expansions:
    - New quests or regions (e.g., Abyss Depths)
    - Unlockable modes (Ironman, Speedrun)

Community Support

- Engage via devlogs, AMAs, and Discord Q&As
- Encourage modding community with guides post-launch

Flowchart

Flowchart is a diagrammatic representation of sequence of logical steps of a program. Flowcharts use simple geometric shapes to depict processes and arrows to show relationships and process/data flow.

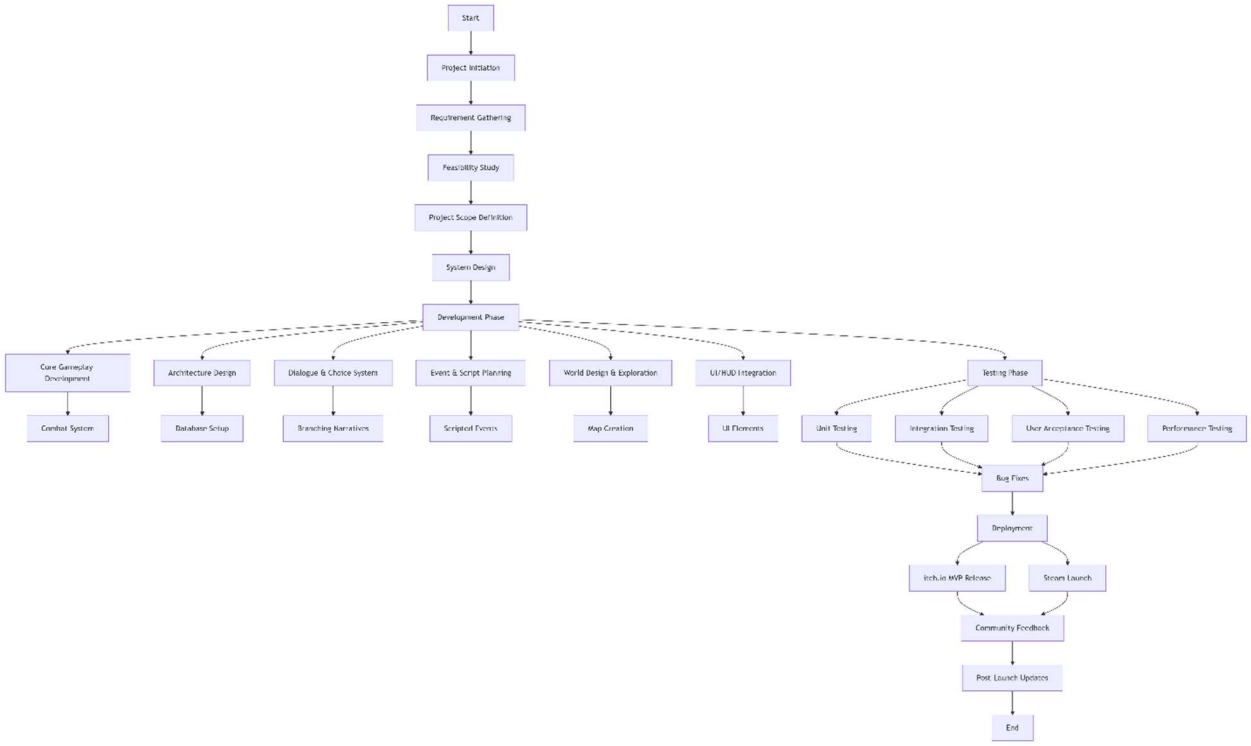

Fig 5.1: Shows a complete game development workflow

The provided flowchart illustrates the **end-to-end development lifecycle** of the Nyxoria project, showing how the game progresses from concept to release and post-launch maintenance. It visualizes both **technical** and **narrative design pipelines**, making it easier to understand how different modules and phases are interconnected.

Key Steps:

1. **Start**

- Initiates the project planning phase for *Nyxoria: Echoes of the Abyss*.

2. **Project Initiation**

- Define the game's core concept: a 2D, choice-driven action RPG.
- Align the development team on goals, themes, and tools (JavaFX in future, RPG Maker currently).

3. **Requirement Gathering**

- Collect game features and technical needs:
    - Turn-based combat system
    - Dialogue and branching choices
    - Corruption mechanic
    - Multiple endings
    - UI, HUD, and music systems

4. **Feasibility Study**

- Analyze if the selected tools (currently RPG Maker VX Ace; potentially JavaFX later) can support desired features.
- Consider performance, scalability, and custom scripting needs.

5. **Project Scope Definition**

- Finalize what will be included in the game:
    - Five regions
    - Core characters and moral dilemmas
    - Combat abilities, AI, and quest systems
    - Dark vs. Light corruption system

6. **System Design**

- Plan the overall structure:
    - Game architecture (events, logic, database)
    - UI design
    - Data flow for choices and consequences

7. **Core Gameplay Development**

- Implement the main gameplay modules:
    - Player movement, interaction
    - Turn-based combat with special skills
    - Enemy behavior and AI patterns

8. **Dialogue & Story System**

- Build a branching dialogue and choice system.
- Store decisions and trigger consequences using conditions or future JavaFX state tracking.

9. **World Design & Exploration**

- Design maps, regions, and NPCs with unique quests.
- Add environmental effects (corruption visuals, weather, etc.).

10. **UI & HUD Implementation**

- Create status bars (HP, corruption meter, skill cooldowns).
- Design intuitive interfaces for inventory, choices, and combat.

11. **Testing Phase**

- **Unit Testing**: Test each game module (combat, dialogue, UI).
- **Integration Testing**: Ensure all systems work smoothly together.
- **User Testing**: Gather player feedback and adjust systems.
- **Performance Testing**: Ensure stability and responsiveness at 60 FPS.

12. **End**

- Close the current development cycle (excluding deployment).
- Prepare for future enhancements or a JavaFX-based reimplementation.

Entity-Relationship (ER) Diagram

The **Entity-Relationship (ER) Diagram** describes the **database structure for a game management system**, specifically designed for an RPG (Role-Playing Game) like *Nyxoria:*

Echoes of the Abyss. Here's a breakdown of its purpose and components:

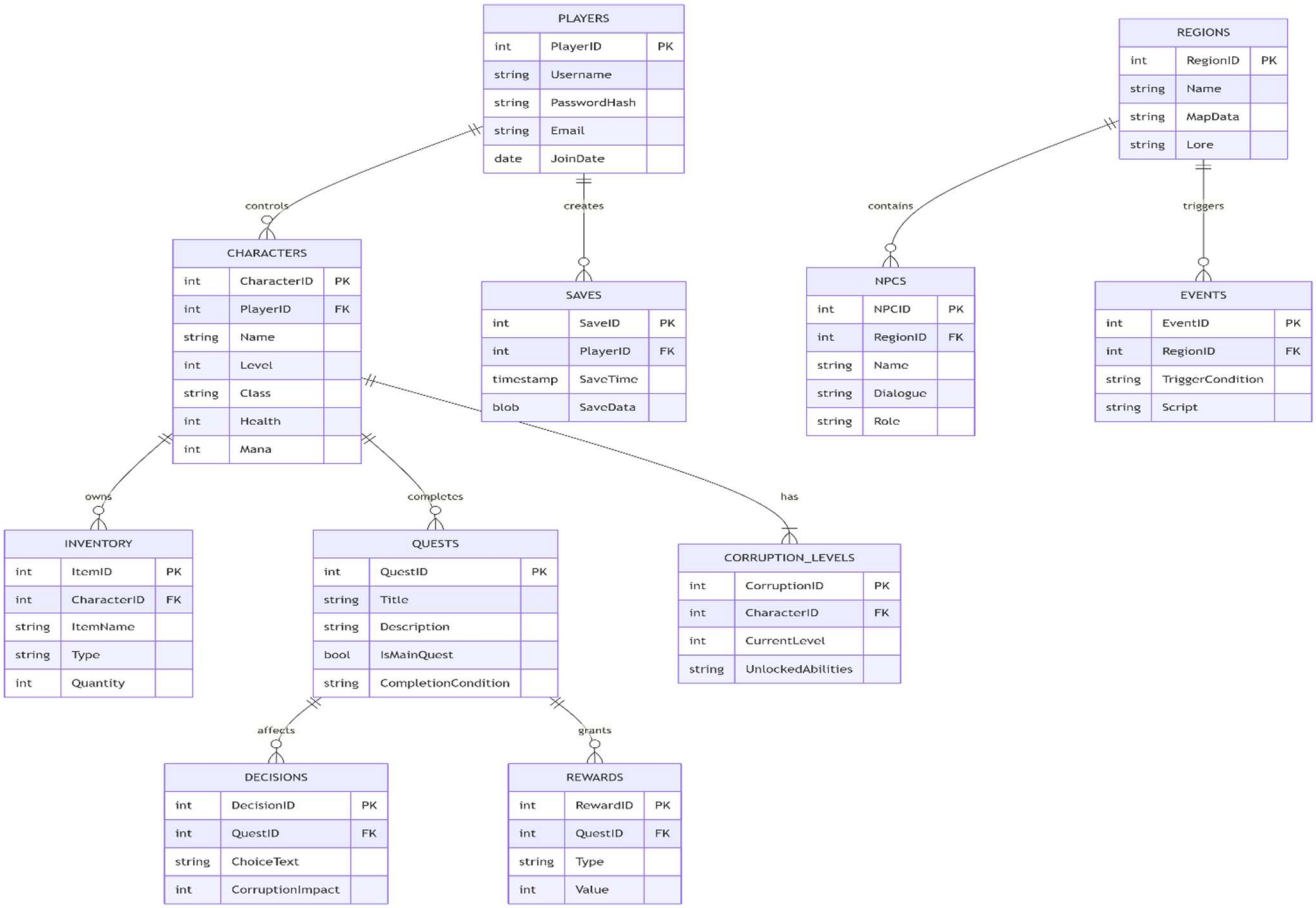

Fig 5.2: Database Structure Entity-Relationship Diagram (ERD)

Entities and Relationships

1. Players

Attributes:

- **PlayerID** (Primary Key): Unique identifier for each player.
- **Username**: Player's display name.
- **Email**: Account email address.
- **PasswordHash**: Encrypted password for security.
- **JoinDate**: Date when the player registered.

2. Characters

Attributes:

- **CharacterID** (Primary Key): Unique identifier for each character.
- **PlayerID** (Foreign Key referencing *Players.PlayerID*): Links to the player who owns the character.
- **Name**: Character's in-game name.
- **Level**: Current progression level.
- **Class**: Character's class (e.g., Warrior, Mage).
- **Health/Mana**: Vital stats for gameplay.

Relationship:

- **Players → Characters**: One-to-Many (One player can create multiple characters).

3. Inventory

Attributes:

- **ItemID** (Primary Key): Unique identifier for each item.
- **CharacterID** (Foreign Key referencing *Characters.CharacterID*): Links to the character owning the item.
- **ItemName**: Name of the item (e.g., "Celestial Sword").
- **Type**: Category (e.g., Weapon, Potion).
- **Quantity**: Number of items held.

Relationship:

- **Characters → Inventory**: One-to-Many (One character can own multiple items).

4. Quests

Attributes:

- **QuestID** (Primary Key): Unique identifier for each quest.
- **Title**: Quest name (e.g., "The Velmoria's Trial").
- **Description**: Summary of objectives.
- **IsMainQuest**: Flag for main vs. side quests.
- **CompletionCondition**: Requirements to finish the quest.

Relationships:

- **Characters → Quests**: One-to-Many (A character can complete multiple quests).
- **Quests → Decisions**: One-to-Many (Each quest has multiple branching choices).
- **Quests → Rewards**: One-to-Many (Completion grants rewards).

5. Decisions

Attributes:

- **DecisionID** (Primary Key): Unique identifier for each choice.
- **QuestID** (Foreign Key referencing *Quests.QuestID*): Links to the quest where the choice occurs.
- **ChoiceText**: Narrative description of the option (e.g., "Spare the traitor").
- **CorruptionImpact**: Moral alignment shift (+/- points).

Purpose:

- Tracks player choices affecting the story and corruption level.

6. Corruption Levels

Attributes:

- **CorruptionID** (Primary Key): Unique identifier.
- **CharacterID** (Foreign Key referencing *Characters.CharacterID*): Links to the character.
- **CurrentLevel**: Numeric value (0 = Pure, 100 = Corrupt).
- **UnlockedAbilities**: Dark powers unlocked at high corruption.

Relationship:

- **Characters → Corruption Levels**: One-to-One (Each character has one corruption tracker).

7. Regions

Attributes:

- **RegionID** (Primary Key): Unique identifier for each area.
- **Name**: Region name (e.g., "Burning Castle").
- **MapData**: Tile layout or coordinates.
- **Lore**: Backstory of the region.

Relationships:

- **Regions → NPCs**: One-to-Many (A region contains multiple NPCs).
- **Regions → Events**: One-to-Many (Triggers scripted events).

8. NPCs

Attributes:

- **NPCID** (Primary Key): Unique identifier.
- **RegionID** (Foreign Key referencing *Regions.RegionID*): Links to the region.
- **Name**: NPC's name (e.g., "High Arbiter").
- **Dialogue**: Default interaction text.
- **Role**: Function (e.g., Merchant, Quest Giver).

9. Events

Attributes:

- **EventID** (Primary Key): Unique identifier.
- **RegionID** (Foreign Key referencing *Regions.RegionID*): Links to the region.
- **TriggerCondition**: How the event activates (e.g., "On item use").
- **Script**: RPG Maker VX Ace event commands.

10. Saves

Attributes:

- **SaveID** (Primary Key): Unique identifier.
- **PlayerID** (Foreign Key referencing *Players.PlayerID*): Links to the player.
- **SaveTime**: Timestamp of the save.

- **SaveData**: Serialized game state (e.g., JSON/BLOB).

Relationship:

- **Players → Saves**: One-to-Many (A player can have multiple save files).

Key Relationships Summary

Entity A	Entity B	Type	Description
Players	Characters	One-to-Many	A player controls multiple characters.
Characters	Corruption Levels	One-to-One	Each character has a moral alignment tracker.
Quests	Decisions	One-to-Many	Quest choices affect the story.
Regions	NPCs/Events	One-to-Many	Regions host interactive elements.

Data Flow Diagram (DFD)

Components of Nxyoria DFD

1. External Entity: Player

- **Role**: The game user interacting with the system.
- **Actions**:
    - Inputs credentials (login).
    - Makes in-game choices (dialogue, combat).
    - Requests game data (load/save).

2. Processes (Key Systems)

Process	Function	Input	Output
Authentication System	Verifies player identity.	Username/Password	Access Grant/Error Message
Game Engine	Manages core gameplay (rendering, physics).	Player Commands	Updated World State
Decision System	Processes narrative choices.	Player's Dialogue Selection	Updated Story Branch

3. Data Stores

- **Database**:
    - Stores player profiles, character stats, and save files.
    - Connected to the Game Engine for save/load operations.

4. Data Flows (Arrows)

Flow	Description
Player → Authentication System	Sends login credentials.
Authentication System → Player	Returns success/failure.
Player → Game Engine	Sends movement/combat commands.
Game Engine → Player	Renders visuals (UI, world).
Player → Decision System	Submits story choices (e.g., moral decisions).
Decision System → Game Engine	Updates quests/corruption levels.
Game Engine → Database	Auto-saves progress.
Database → Game Engine	Loads saved data on request.

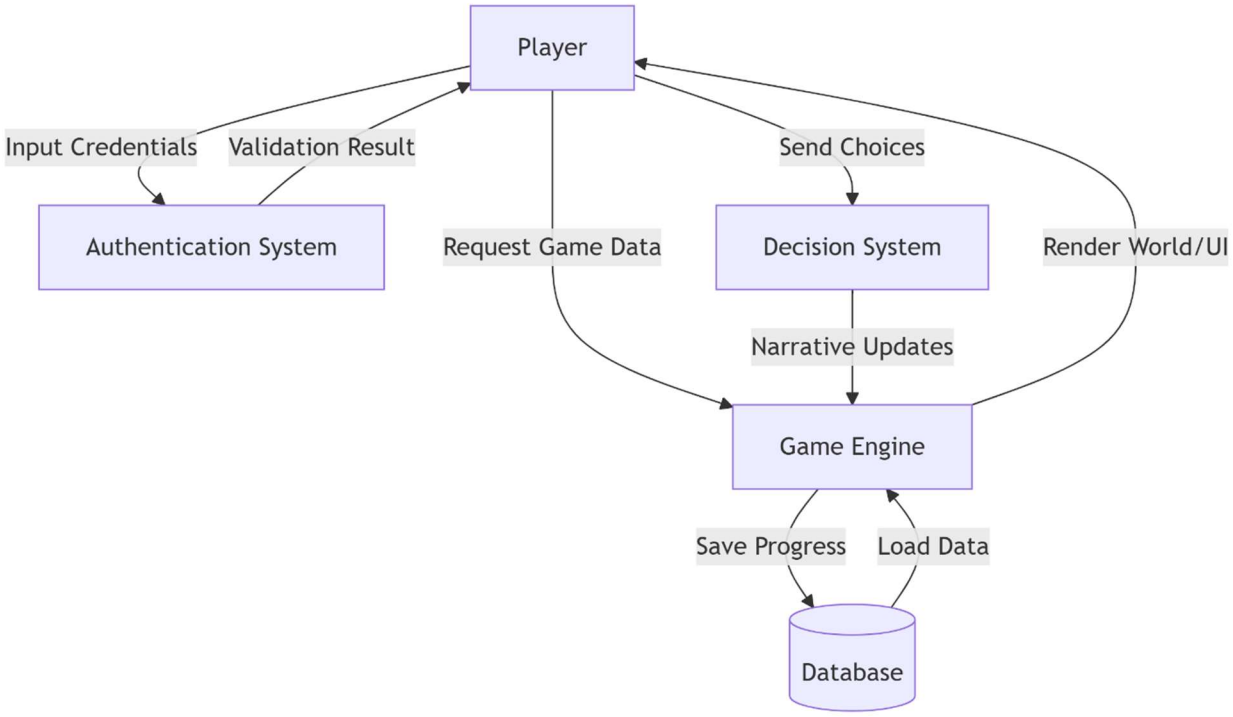

Fig 5.3: Functional Data Flow Diagram (DFD)

Sequence Diagram

Purpose of a Sequence Diagram To model high-level interaction among active objects within a system. To model interaction among objects inside a collaboration realizing a use case. It either models' generic interactions or some certain instances of interaction.

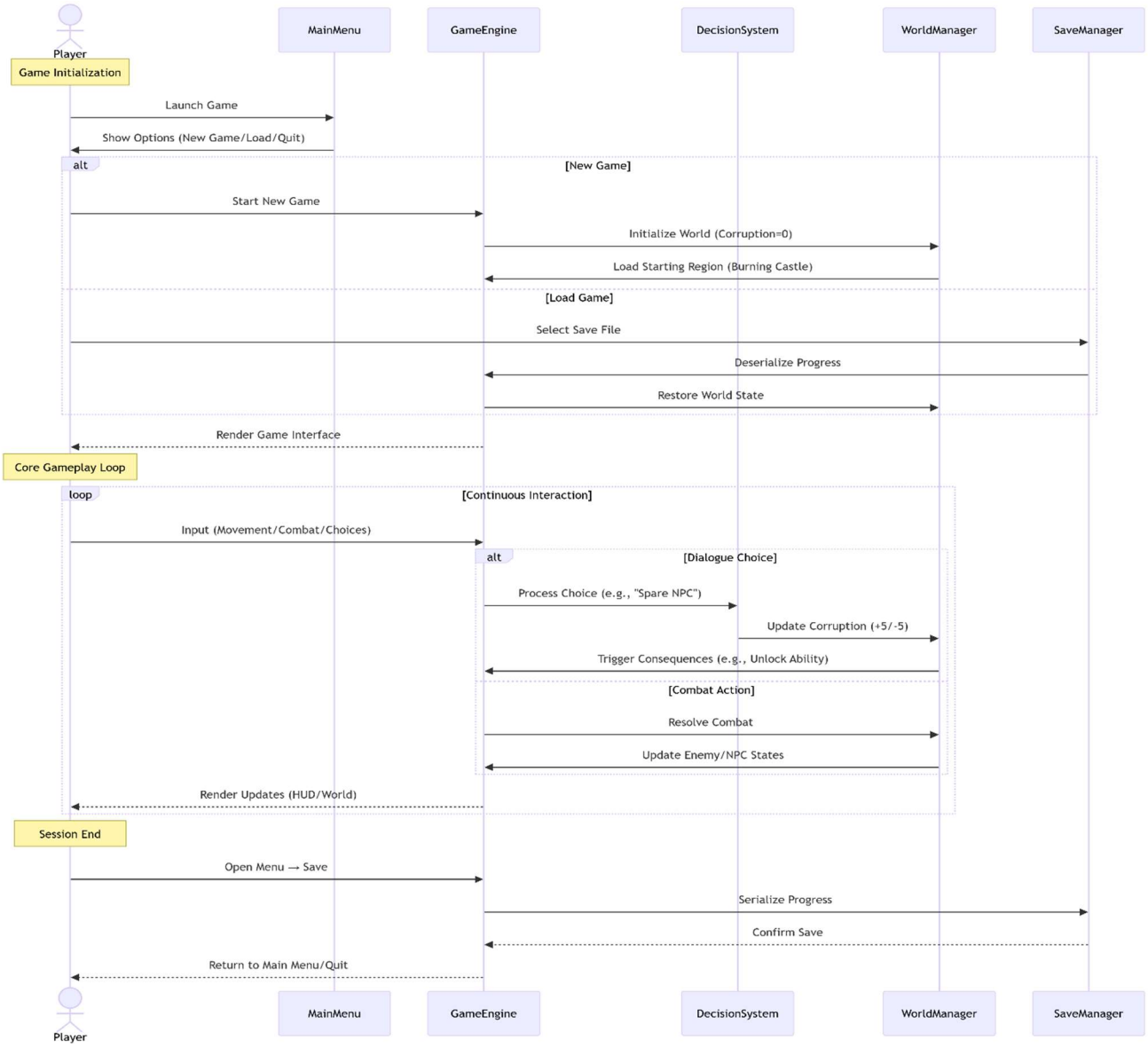

Figure 5.4: Game Initialization and Core Gameplay Sequence

The uploaded diagram is a **sequence diagram** representing the interactions between the **Player** and various core subsystems of the game. It demonstrates how the game engine responds to user actions such as launching the game, making decisions, saving progress, and engaging in gameplay.

Nxyoria Sequence Diagram Details:

1. Launch & Game Initialization

- **User Action**: Player starts the game application.

- **System Action**:
    - MainMenu displays options: **New Game**, **Load Game**, or **Quit**.

2. Start New Game

- **User Action**: Player selects **New Game**.
- **System Action**:
    - GameEngine initializes the game world with default values (e.g., Corruption = 0).
    - WorldManager loads the starting region (e.g., **Burning Castle**).
    - GameEngine renders the game interface and transitions into gameplay.

3. Load Existing Game

- **User Action**: Player selects **Load Game** and chooses a save file.
- **System Action**:
    - SaveManager deserializes saved progress.
    - WorldManager restores the world state to match the saved data.
    - GameEngine renders the restored game interface.

4. Core Gameplay Loop (Continuous Interaction)

- **Loop begins**: Player interacts with the world continuously through inputs.

a. Input Handling

- **User Action**: Player performs actions like movement, combat, or making choices.
- **System Action**:
    - GameEngine processes the input and delegates the logic.

b. Dialogue Choice

- **User Action**: Player makes a decision in dialogue (e.g., *Spare NPC*).
- **System Action**:
    - DecisionSystem processes the choice.
    - Updates the **corruption level** (e.g., +5 for evil, -5 for good).

- Triggers **consequences** such as unlocking new abilities or changing NPC behavior.

c. Combat Action

- **User Action**: Player engages in combat.
- **System Action**:
  - GameEngine processes the combat round.
  - Resolves enemy actions, health deductions, or ability activations.
  - WorldManager updates the states of enemies or NPCs after combat.

d. Render Updates

- **System Action**:
  - GameEngine continuously updates the interface to reflect:
    - New HUD values (e.g., health, corruption)
    - Changed world visuals (NPC reactions, area effects)

5. Session End – Save Game

- **User Action**: Player opens the pause/menu and selects **Save**.
- **System Action**:
  - GameEngine triggers the save process.
  - SaveManager serializes the current game state.
  - Player is returned to the main menu or can choose to quit.

6. End Session

- **User Action**: Player quits to main menu or closes the game.
- **System Action**:
  - MainMenu reappears for the player to restart or exit.
  - The session ends gracefully.

Chapter 6

Project Outcome

The **Nyxoria: Echoes of the Abyss** project represents a substantial step forward in the design and development of indie role-playing games. It successfully brings together fundamental gameplay elements such as player exploration, turn-based combat, and an immersive fantasy world, all built within a modular and expandable structure. The game is crafted to deliver a satisfying and engaging experience for players interested in action-oriented RPG mechanics within a rich visual and thematic setting.

One of the most significant outcomes of the project is the creation of a **responsive and strategic combat system**, developed using RPG Maker VX Ace. The system supports turn-based mechanics, animated skill usage, and health/resource tracking, offering players a challenging and rewarding battle experience. Combat integrates intuitive controls and dynamic feedback to enhance usability and immersion, laying a solid foundation for more complex mechanics in future iterations.

The project also includes an extensively **designed game world** consisting of multiple regions, each with unique environmental themes and exploration opportunities. Players can navigate through diverse settings such as the *Burning Castle* and *Tower Town*, encountering NPCs, completing quests, and uncovering hidden items or lore. The level design encourages curiosity and supports non-linear progression, contributing to a more open and replayable experience.

The game interface includes a **heads-up display (HUD)** that provides real-time feedback on player health, skill usage, and status effects. A clean and accessible UI layout ensures that players can focus on gameplay without being overwhelmed, while thoughtful menu systems support inventory access, character stats, and save/load features.

From a technical standpoint, the project leverages **RPG Maker VX Ace's** built-in event system and database management tools to control character movement, enemy behavior, item logic, and UI transitions. The modular design allows for efficient testing, debugging, and future scalability. Performance optimization ensures that the game runs smoothly on mid-range hardware, with responsive input handling and consistent frame rates.

Although some features such as the **corruption system** and **dialogue branching engine** are still under development, the current version of the project establishes a strong foundation for a full-featured release. It reflects sound planning, effective modular design, and adherence to key game development principles.

In terms of **future development**, the game is intended to be **recreated and extended using JavaFX**. This transition will allow:

- Greater flexibility in UI/UX design
- More control over event handling and graphics rendering
- Enhanced scalability for multi-platform deployment
- Easier integration with custom data management systems (e.g., JSON/XML for save files, dynamic UI elements)

This future migration aims to take advantage of JavaFX's modern interface capabilities and open new paths for performance optimization and cross-platform support.

Overall, *Nyxoria* not only fulfills its initial design and development goals but also establishes itself as a prototype capable of evolving into a robust, fully customized RPG platform. It stands as a strong example of how indie projects can grow through iterative development and forward-thinking technology adoption.

6.1 Title Screen

Figure 6.1: Nyxoria Title Screen

- The cover screen serves as the **first point of immersion**, combining atmospheric background visuals, custom logo design, and an evocative title sequence.
- A brief introductory message or tagline (e.g., *"Restore Light or Embrace the Abyss"*) sets the thematic tone of the experience.
- Interactive options such as **New Game**, **Load Game**, and **Exit** are neatly presented to guide user navigation.
- Music and animated visual effects (e.g., fire embers, flickering torches) enhance immersion and convey a mood of urgency and decay.

6.2 New Game Flow

Figure 6.2: New Game Initialization

Selecting **New Game** initiates, a scripted sequence that sets up the game environment.

The game begins by:

- Resetting all key variables (e.g., player_level = 1, base stats)
- Loading the starting region (*Burning Castle*) via the built-in map engine
- Playing an introductory cutscene to establish setting and tone

The player is then placed into an active gameplay environment with freedom to explore, engage in basic quests, and encounter the game's core mechanics.

6.3 Load Game Functionality

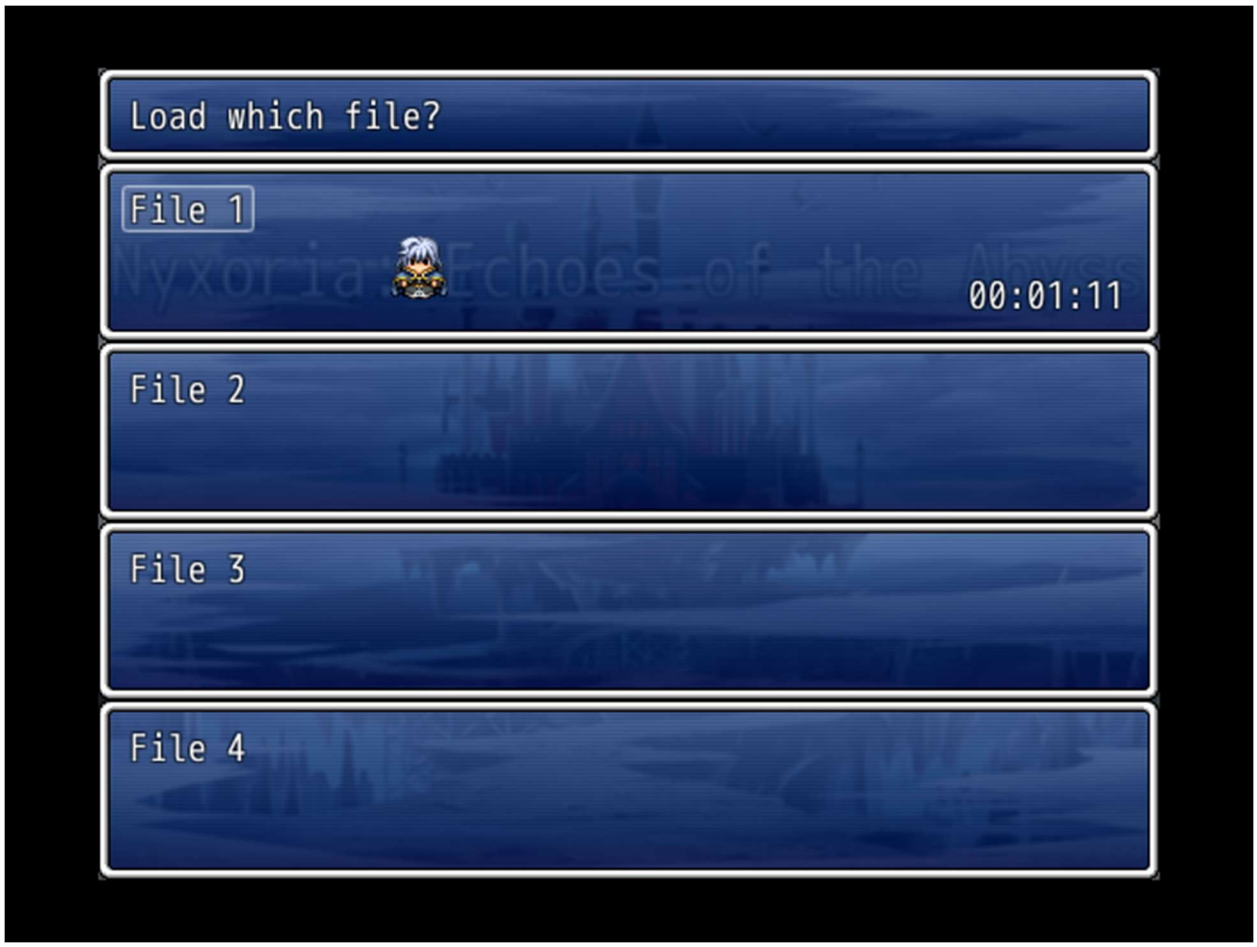

Figure 6.3: Load Game Screen

The **Load Game** screen allows users to resume previous progress by selecting from a list of available save files, which display:

- Save timestamps
- Player location
- Current level or status indicators

Upon selection:

- The SaveManager restores player data, inventory, and world state
- NPC positions, completed quests, and environment flags are also reloaded

This ensures a seamless transition back into the game world, preserving immersion and consistency.

6.4 Core Gameplay Screen

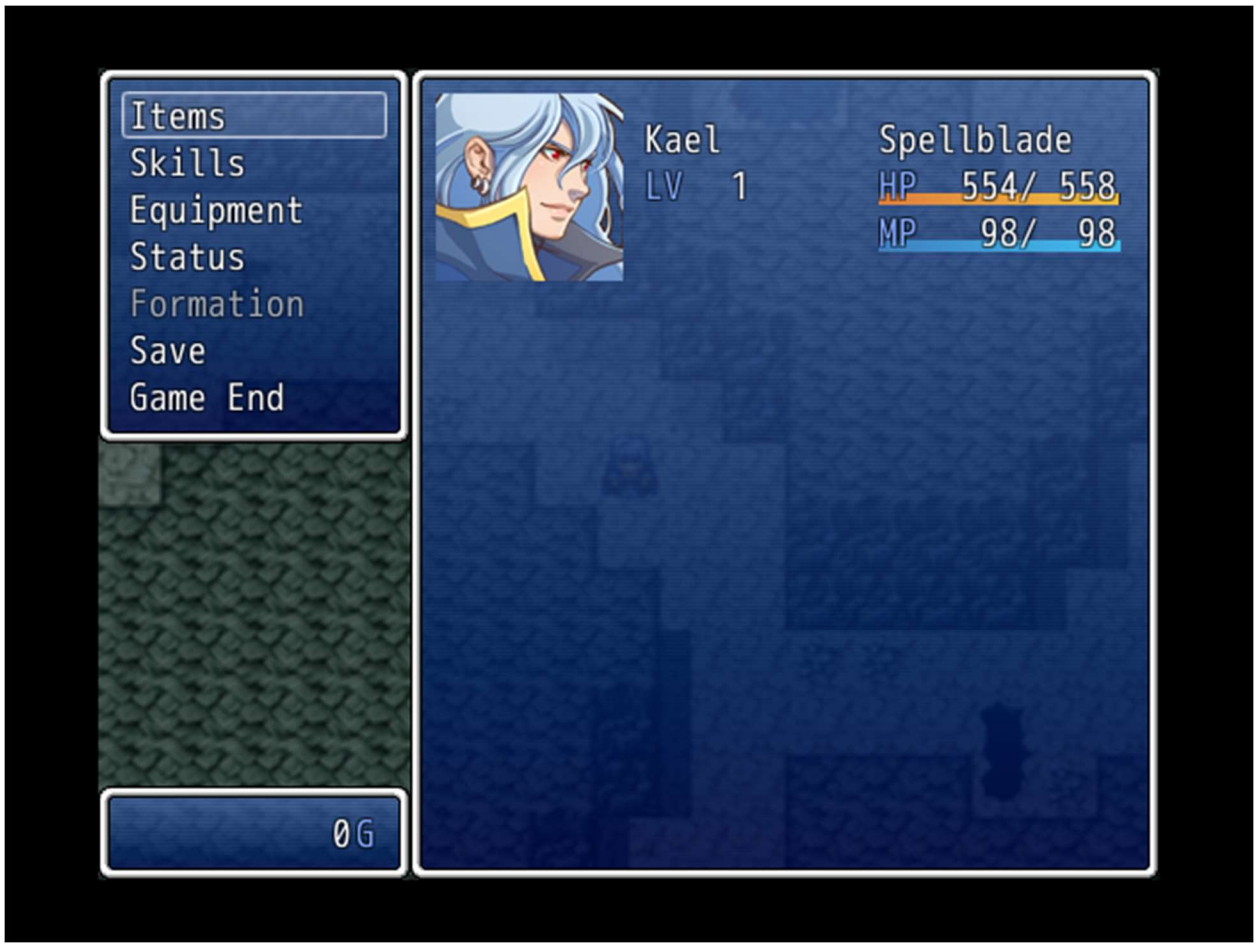

Figure 6.4: Main HUD and Exploration UI

The primary gameplay interface is designed for clarity, responsiveness, and immersion. It displays:

- Player health bar
- Skill cooldown timers
- Inventory and quest access via menu buttons

Exploration includes:

- Richly designed environments with unique lighting and weather effects
- Region-specific ambience that enhances the mood of each area

The UI ensures that players have quick access to all important stats and systems without breaking immersion.

Note: The corruption meter and dynamic narrative feedback systems are planned for future updates.

6.5 Combat System

Figure 6.5: Combat Interface

Combat in Nyxoria uses a **turn-based engine** with visual enhancements and tactical depth. Features include:

- Skill-based combat using custom animations
- Strategic turn-order and ability management
- Enemy AI behavior governed by conditional logic

Current available skills include:

- **Void Slash** – A high-damage dark-type attack

- **Celestial Chains** – Temporarily immobilizes the enemy
- **Purge** – A powerful area attack

The combat interface presents:

- Player/enemy HP bars
- Battle logs
- Special effect indicators

Future updates may expand on skill types and add elemental/resistance systems.

6.6 (Reserved for Future Development)

Planned: Dialogue and Morality System

This section is reserved for a future update to implement:

- A **choice-based dialogue system**
- A **corruption/morality tracker** that impacts gameplay and world state

These systems will allow players to make story-altering decisions, influencing allies, enemy behavior, and unlocking different narrative paths.

6.7 Save System

Figure 6.7: Save Menu

The game features a reliable **save system** accessible from the in-game menu.

Key features:

- Allows players to save progress at any time (based on conditions set in events)
- Tracks and stores essential data such as:
    - Current map and coordinates
    - Player level and stats
    - Inventory and equipped items
    - Game progress flags (quests, events)

Saves can be loaded from the main menu to resume gameplay without loss of progression.

References

1. Enterbrain. (2011). *RPG Maker VX Ace Official Documentation*. Kadokawa Games.
    - *Covers event scripting, database management, and RGSS3 scripting for Nyxoria's core systems*.
2. Oracle. (2024). *JavaFX Documentation: Building Desktop Applications*. Retrieved from https://docs.oracle.com/javase/8/javafx/api/
    - *For UI rendering, animation, and event handling in future JavaFX implementation*.
3. SQLite. (2024). *Lightweight Database for Local Storage*. Retrieved from https://sqlite.org/docs.html
    - *Local save data management (characters, quests, inventory)*.
4. Mozilla Developer Network (MDN). (2024). *JavaScript Event Handling*. Retrieved from https://developer.mozilla.org/en-US/docs/Web/Events
    - *For UI interactivity in JavaFX (adapted from web paradigms)*.
5. Harvey, A. (2011). *The RPG Maker VX Ace Cookbook*. Packt Publishing.
    - *Practical recipes for quest systems and world design*.
6. Sicart, M. (2013). *Beyond Choices: The Design of Ethical Gameplay*. MIT Press.
    - *Ethical frameworks for Nyxoria's corruption system*.
7. Team Cherry. (2017). *Hollow Knight Design Notes*. Retrieved from https://www.teamcherry.com.au/
    - *Inspiration for metroidvania-style world exploration*.
8. Freesound. (2024). *CC0 Sound Effects Library*. Retrieved from https://freesound.org/
    - *SFX for combat, NPC interactions, and ambiance*.
9. RPG Maker Forums. (2024). *Community Scripts and Plugins*. Retrieved from https://forums.rpgmakerweb.com/
    - *Custom RGSS3 scripts for advanced features*.

Key Takeaways

- Academic: Juul, Salen, and Sicart provide theoretical backing for game design choices.
- Technical: RPG Maker VX Ace, JavaFX, and SQLite docs cover implementation.
- Assets: Aseprite and Freesound support art/sound development.