Metropolia - University of Applied Sciences

Group 06

Zombiator

Requirement Specification

Nhi Dinh	2314582
Lihini Hewage	2315328
Kamonnun Silarat	2322365
Jiayue Zheng	2315316

Version history

Version	Date	Author	Description
1.0	13.06.2023	Nhi Dinh	Finalize the document
1.1	09.12.2023	Nhi Dinh	Update Section 2.2 Game Idea, 2.3 Game Info, 2.4 Game Flow Chart
			Update Section 3 Functional Requirements
			Update Section 4 Execution of Technical Requirements
			Update Section 5 Quality Requirements
1.2	13.12.2023	Nhi Dinh	Update Section 6 Further Development Ideas Finalized the document

Contents

1 Introduction

1.1 Purpose of the Document

This document serves as a comprehensive requirements specification for **Zombiator**. Its main purpose is to provide a clear and structured outline of the project's objectives, current state, visions, functions, and quality requirements, as well as constraints. It serves as a foundational reference for all stakeholders involved in the game development process, including project developers, project team leaders and teachers.

1.2 Target Audience

The intended audience for this document includes:

- **Project Developers**: The individuals are responsible for designing, coding, and implementing functions, features of the product.
- **Project Team Leaders**: The individuals who oversee and manage the development process, ensuring that it aligns with the defined requirements and project goals.
- **Teachers (Testers and Graders)**: Educators who actively participate in testing **Zombiator**, assess and grade the game based on the predefined criteria as well as providing valuable feedback on the product.

1.3 Document Structure

To ensure clarity and facilitate efficient use of this document, it is organized into the following key sections:

- **Vision**: This section provides an overall vision statement that represents the goals, and core concepts of the product. In this section, we also provide a flow chart to show how the program plays out and the stages players will go through.
- **Functional Requirements**: includes the specific functions and features of the game. It also provides a detailed overview of the game concept and outlines various use cases.

- **Execution of Technical Requirements**: This section provides an overview of the technical requirements executed in both the backend and frontend components of **Zombiator**, emphasizing the technologies employed and the communication architecture for real-time updates.
- **Quality Requirements**: includes the quality standards and criteria that **Zombiator** must meet such as performance, reliability, usability, and other factors that contribute to a high-quality player experience.
- **Further Development Ideas**: This section will briefly summarize some ideas for further development of the product.

2 Vision

2.1 Overview

The vision section provides an overall understanding of our product, providing insight into the game concept, and game mechanics in the second phase of this project.

2.2 Game Idea

In 2025, Earth's environment is in crisis, causing a pandemic. A strange virus has turned people into zombies, starting in **Spain** and spreading worldwide. In **Zombiator**, player will play as a hero from **Finland**, the last safe place on Earth during a zombie outbreak. The mission is to flight around the world,defeat the zombies, rescue the survivors, and stop the pandemic.

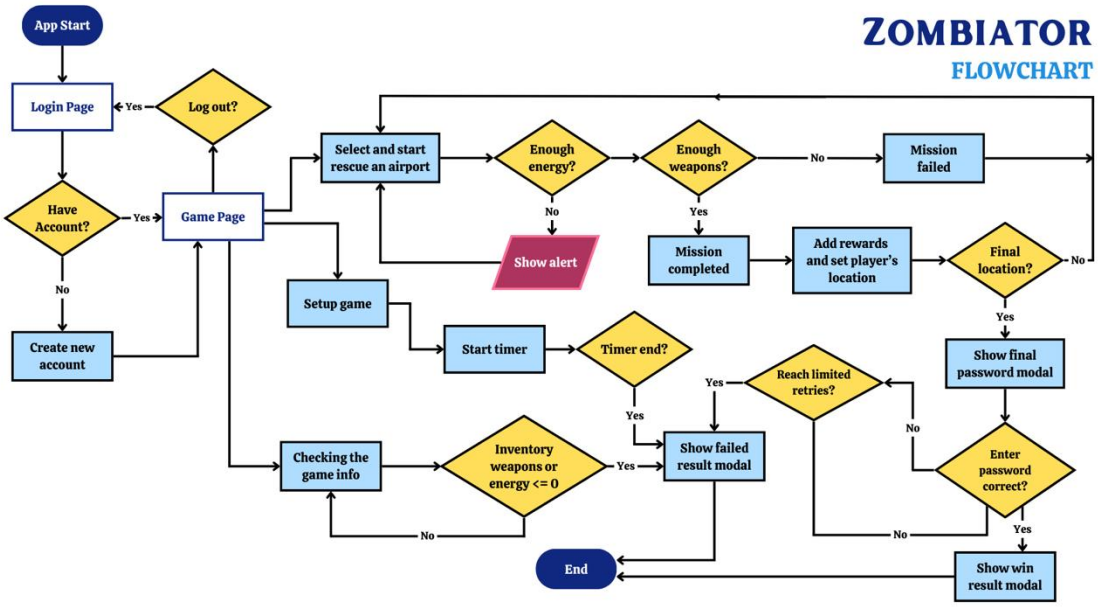
Here's what the game is about:

- **Rescue the World**: Player’s mission is to save survivors in different countries. The journey starts at Helsinki, Finland, ends at Madrid, Spain.
- **Manage Resources**: Player will need **energy**, and **weapons** to complete the mission. **Energy** lets the player fly to other countries, and weapons help the player defeat zombies.

- **Timer Management**: The game will initiate a timer at the beginning, and if the game is not successfully completed by the time it runs out, it will result in a failure.
- **Face Challenges**: The challenge level varies in each city, determined by evaluating the current weather conditions and air pollution data associated with the provided latitude and longitude information.
- **Earn Rewards**: Completing mission gives the player energy and weapons. Harder missions mean bigger rewards.
- **The Difficulty Level and Earning Rewards**: The default display does not reveal the difficulty level and earned rewards. These details become visible once the player initiates the airport rescue mission.
- **The Final Challenge**: The last mission is in **Marid**, and it demands a large supply of weapons. After overcoming the final challenge, players enter the collected **master passwords** to obtain the vaccine and save the world.

2.3 Game Flow Chart

The flow chart will provide insight into how the game will play out and what stages players will go through.



3 Functional Requirements

In this chapter, we break down the user stories to outline player interactions, gameplay mechanics, and goals to ensure that the product delivers a great gaming experience. Each user story identifies a specific action or ability that the player has, and a benefit the user can gain by completing the action. Functional requirements are an essential component of this document, complementing the overall vision and quality requirements, and together they form a comprehensive blueprint for developing and successfully implementing **Zombiator**.

#	As a/an	I want to / I can	So that
1	Player	Read the game information on the authentication page	I can understand the game's purpose and mission.
2	Player	Watch the game play on the authentication page	I can understand how the game will unfold.
3	Player	Authenticate by providing Email and Password	I can access the game.
4	Player	See the error notification if I enter incorrect email or password	Helping me identify and modify authentication issues.
5	Player	Log out of the game	I can retry it afterward, maintaining the flexibility to resume gameplay at my convenience.
6	Player	View the Google Map with various pinned airport location on it	Enabling me to navigate and explore different location within the game.
7	Player	See the Timer counting down when entering the game page	I can track the time remaining before the game end.
8	Player	See the information about current inventory and password fragments collected	Allowing me to effectively manage my resources throughout the gameplay.
9	Player	Modify player's name	I can personalize and update it according to my preferences.

10	Player	Choose the airport I want to go	My airplane will move there, allowing me to do my rescue missions effectively.
11	Player	View the current information of the airport such current air pollution, temperature and wind speed	So that I can make informed decisions and strategize effectively within the game difficulty level.
12	Player	Start a rescue mission in a chosen country	I can do the core action of saving survivors and earning rewards.
13	Player	Enter the number of weapons that I want to use	I can have more interaction with the game and improve the gaming experience.
14	Player	See the error notification if I provide an invalid number of weapons	Helping me identify and then correct the issue.
15	Player	Complete a mission	I can receive rewards, including energy and weapons, and view the earned rewards prominently displayed within the game layout.
16	Player	See a notification if I don't complete a mission in a country.	I can know the result of my mission and what happens next.
17	Player	Can see a popup prompting me to enter the master password after rescuing the final location	I can proceed to the next stage and finish the game successfully.
18	Player	Can see the error notification if I provide an incorrect master password	I can correct my mistake and continue with the game smoothly.
19	Player	Can see the "**You Win"** popup when I complete the game successfully	I can receive acknowledgment and celebration for completing the game successfully.
20	Player	Can see the "**You Lose"** popup when I fail the game	I can know that I was unsuccessful in completing the game, which allows me to understand the results and encourages me to strategize, learn from

			mistakes, and make necessary adjustments next time.
21	Player	View the ranking list to see my scores	I can compare my performance with other players and aim for the top position
22	Player	Start a new game by clicking on the Retry button	Enabling me to begin a fresh gaming session and pursue improved outcomes.
23	Player	Listen the background sounds while playing game	Enhance my gaming experience
24	Player	Can pause the game time	Allowing me to temporarily halt the game progress when I need to attend to other tasks.
25	Player	Can resume the game time	Enabling me to continue the game from where it was paused

4 Execution of Technical Requirements

This section provides an overview of the technical requirements executed in both the backend and frontend components of **Zombiator**, emphasizing the technologies employed and the communication architecture for real-time updates. Additionally, it outlines the technical implementation of user navigation, highlighting the integration with Google Maps and data retrieval from external APIs and internal databases.

4.1 Backend Technologies

- **Flask:** The backend of Zombiator is powered by Flask, a micro web framework for Python, providing a robust foundation for handling server-side functionalities.
- **Flask-SQLAlchemy**: To interact with a relational database, Flask-SQLAlchemy is employed, facilitating seamless integration and manipulation of game data.
- **Flask-Login**: User authentication and session management are implemented using Flask-Login, ensuring a secure and personalized gaming experience.

- **Python-Dotenv**: The python-dotenv library is employed to manage environment variables, enhancing security and configurability.
- **Object-Oriented Programming (OOP):** The backend architecture embraces Object-Oriented Programming principles to enhance code organization, modularity, and maintainability. Classes and objects are utilized to model various components of the game, promoting a structured and scalable development approach.

4.2 Frontend Technologies

The frontend of **Zombiator** is crafted with a combination of standard web development technologies to ensure an intuitive and visually appealing user interface.

- **HTML and CSS**: The foundation of the frontend is built upon HTML and CSS, providing the structure and styling for the game interface.
- **JavaScript (JS)**: JS is used for implementing dynamic and interactive elements within the frontend, enhancing user engagement.
- **Bootstrap:** The Bootstrap framework is incorporated to streamline the design process, ensuring responsiveness and a consistent look and feel across different devices.

4.3 Communication Between Backend and Frontend

The backbone of real-time communication between the backend and frontend is established through AJAX (Asynchronous JavaScript and XML) requests using the Fetch API. This approach allows the frontend to asynchronously make HTTP requests to the backend, enabling dynamic updates and interaction without requiring a full page reload.

4.4 Building Google Map Info and User Navigation

The technical implementation of user navigation involves a mechanism integrated with **Google Maps**. All marker information about the airports is constructed by retrieving real-time weather and air pollution data from the **Open Weather API**, supplemented by internal data stored in the database. This integration creates a comprehensive and dynamic navigation system, providing players with essential information about airports for efficient gameplay

5 Quality Requirements

In this chapter, we outline the specific quality requirements that **Zombiator** must meet to deliver a seamless and enjoyable gaming experience. These requirements set the standard for the game performance, ensuring the game operates efficiently and responds promptly to user actions.

Performance Requirements:

- **Fetching Data**: The communication between the Frontend and Backend to retrieve the airport list, along with weather and air pollution information from the **Open Weather API**, must be completed within a maximum of 2 minutes.
- **Instant Feedback**: The user should receive immediate feedback for every action they perform, not only during authentication but also throughout the progression of the game.

Reliability Requirements:

- **Stability**: Ensure the game does not crash or freeze during progression of the game.
- **Graceful Handling**: The game should handle unexpected situations gracefully.

Compatibility Requirements:

- **Platform Compatibility**: Ensuring a seamless gaming experience, the game is designed to be playable across a diverse range of web browsers. This commitment to platform compatibility aims to provide accessibility and enjoyment for players, regardless of their preferred browser, fostering a broader and more inclusive gaming community.

Usability Requirements:

- **User Experience (UX)**: the game instructions are clear and easy to understand.
- **User's age**: The game's content and gameplay must be suitable for young users (12+)

Gameplay Balance Requirements:

- **Difficulty Balance**: The difficulty levels should be balanced, offering a fair and enjoyable experience. The game should not be too easy to win, but it should also not be too difficult to accomplish.

Data Integrity and Maintenance: The game's relational database, derived from the airport database used in this course, must ensure data integrity through constraints and validation. At the same time, it must support the schema modifications for long-term maintenance and adaptability, allow for flexible schema extension while maintaining data accuracy and consistency.

These quality requirements will help ensure that **Zombiator** meets high standards of performance, reliability, usability, and player experience while maintaining balanced and enjoyable gameplay.

6 Further Development Ideas

In considering further development ideas for the game, several ideas can be explored to enrich the player experience.

- Improve the animations and audios of the game in each mission.
- Allow player to turn on/off the background music.
- Add more types of reward, e.g. mission support item and expand the rescue region to Asia with a new kind of mission.
- Research new techniques to reduce the map loading time.