Requirements analysis

Version: 1.0

1. Introduction

The story of Steve is a story-driven psychological horror game that is as close to reality as possible, putting the player in the role of Steve - a man in the midst of a personal crisis. Accompanied through the day and his nightmarish experiences, the player delves deep into Steve's reality and his fragile psyche.
The working name of the game is “**GetOut**”

The points that haven’t been mentioned yet can’t be filled out at this given point of time in the development process.

1.1. Purpose of the system

Creating an immersive, terrifying experience for the player. The system encompasses various components, each serving a specific purpose to enhance the horror elements.

1.3. Objectives and success criteria of the project

Create a game using an engine and documenting the process of development.
In the end all participants should be able to understand the steps of game-specific software development processes and be able to organize themselves as a team.

1.4. Definitions, acronyms, and abbreviations

Steve: the working name of the main character who is played in first person (his using name in game will be clarified in the future)
Boss: Steve’s boss who will occur in the second scene of the game, at work. He will be a side character.
Wife: divorced wife of Steve, she will only occur in the forth (last) scene. She is the main antagonist and will be included in the main part of the story.
Smiley: a grinning face which looks very scary that will occur as an emoji in the last scene at home. This is a part of the scary illusions that the player will face during the game.

1.5. References

Trello
GitHub
StoryBoard
SCRUM-Protocol

Time Management (will be translated in future)

1. Phasenaufteilung

1. **Konzeptionelles Design**

   Das Spiel wird sowohl von der Story als auch vom ersten Aussehen beschrieben. Es wird ein Storyboard erstellt und erste Statemachine Diagramme werden erstellt um den Ablauf im Spiel zu beschreiben.

Zeitaufwand: 10h

2. **Technisches Design**

   Die Architektur sowie das genaue Aussehen vom Programmiercode werden hier beschrieben. Dabei werden Klassendiagramme, Sequenzdiagramme und ähnliche angelegt.

   Zeitaufwand: 14h

3. **Entwicklung**

   In dieser Phase wird GetOut schlussendlich umgesetzt. Mithilfe einer Engine werden nun alle Anforderungen übersetzt.

   Zeitaufwand: ~120h

4. **Tests**

   Nachdem GetOut implementiert wurde, gibt es mehrere große Testphasen, die möglichst viele Szenarien abdecken sollen.

   Zeitaufwand: 26h

5. **Fertigstellung**

   In dieser letzten Phase wird das Spiel für die verschiedenen Plattformen fertiggestellt und ein Report geschrieben. In diesem Report wird nochmals genauer auf die einzelnen Phasen eingegangen

   Zeitaufwand: 10h

2. Proposed system

2.1. Overview

- Game should be able to run on Windows/MAC/Linux
- All developers are using Windows
- Game will be programmed and designed with Unity

2.2. Functional requirements

ID: FR1	Title: Interact with surroundings
Description	User interacts with objects by using the left mouse button
Acceptance Criterion	1. User triggers event with mouse clicking and some action will follow

Notes	Will be the main function throughout the game

ID: FR2	Title: Move in World
Description	User can move by using W,A,S,D
Acceptance Criterion	1. User must be able to move in all directions except for jumping 2. Movement is developed for FP (First person) 3. Movement speed is variable 4. Movement is independent from Head up and down rotation
Notes	

ID: FR3	Title: Enter Gameplay
Description	User must be able to load gameplay at any part of the story and from the main menu
Acceptance Criterion	1. User must be able to start or resume gameplay 2. If a checkpoint is given, user should be able to spawn there
Notes	Requires functional memory management

ID: FR4	Title: Save Gameplay at any point
Description	User is able to save at any given point throughout the game
Acceptance Criterion	1. After saving the game, there is a file containing the current state of the story.
Notes	Requires functional memory management

2.3 Nonfunctional Requirements

ID: NFR1	Title: Performance
Description	Make sure that game runs smoothly at all times

Acceptance Criterion	1. The game should have a smooth gameplay, even during complex scenes and effects
Notes	

ID: NFR2	Title: Load time
Description	Make sure that user doesn't have to wait too long to load into the game
Acceptance Criterion	1. The game should load within 10 seconds when starting a new session
Notes	

ID: NFR3	Title: Usability
Description	The game should have understandable instructions in general
Acceptance Criterion	1. Every button should have a suitable description or name and suitable design 2. The user is getting introduced to the game without a direct tutorial (Indirect explanation)
Notes	

ID: NFR4	Title: Reliability
Description	The game must not crash or freeze during gameplay session
Acceptance Criterion	1. The game must not crash more than once per hour under normal conditions 2. The game must not crash more than once every half hours under stress conditions (e.g. maximum settings, switching between scenes) 3. The game must auto-save progress at regular intervals 4. Auto-save points should be created at key moments (Change of scenes, interaction with specific objects)
Notes	All crashes and critical errors must be logged with detail information in order to debug

ID: NFR5	Title: Audio-Visual Quality
Description	The game should have a decent resolution and a pleasant sound quality

Acceptance Criterion	1. Game should work at low and high quality settings 2. Using a quality microphone
Notes	

2.3.2 Documentation

Documentation is done using Doxygen

Short guide on how to use Doxygen:

apt install Doxygen

Nach dem Programmieren muss man ein neues Verzeichnis machen namens docs und dort drin dann doxygen -g machen, das generiert ein DoxygenFile wo die ganzen configs drin sind

2.3.5 Error handling and extreme conditions

2.3.6. Quality issues

2.3.7. System modifications

2.3.8. Physical environment

2.3.9. Security issues

2.3.10. Resource issues

2.4. Pseudo requirements

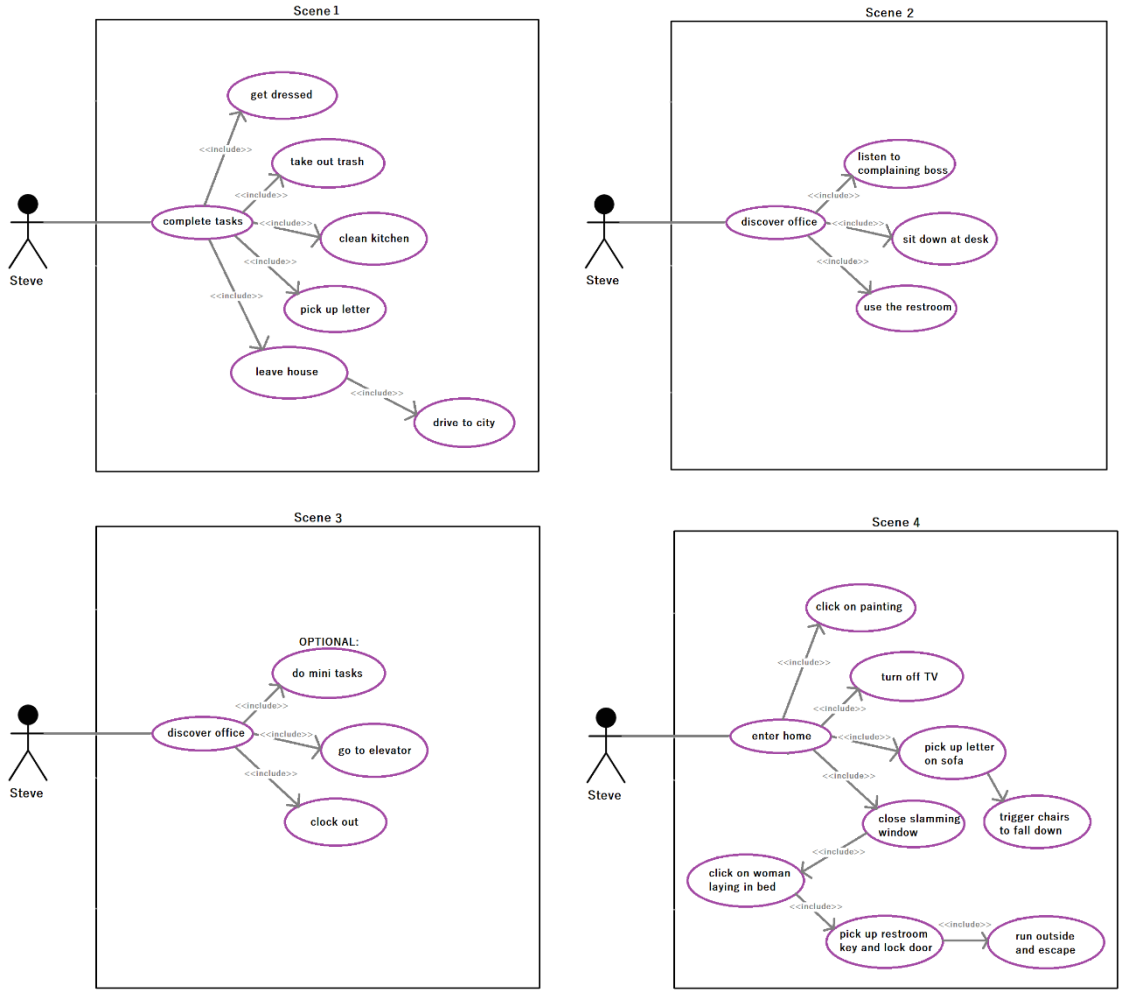
Using Unity instead of Unreal Engine 5:

We had many problems with using Unreal Engine. There were a lot of questions about how to use it and we were not able to get answers from the internet which forced us to change the engine.
In the long run we totally benefited from it because we understand the functional processes much better.

2.5. System models

2.5.1. Scenarios

2.5.2. Use case model



The main character is called Steve and he's a divorced man who has psychological problems.
In the first scene he wakes up on his sofa and the player can interact in first person with items inside the house.
The player has to accomplish tasks before being able to continue with the story. He has to get dressed, take out the
trash, clean the kitchen, pick up a letter laying on the floor and finally leave the house and drive to the city to work
by clicking on the car.

For the second scene the player will be teleported into the office and again is able to accomplish optional tasks such as
listening to complaining boss by stepping in front of the boss's door, using the restroom or sitting down at his desk,
which allows the story to continue.

As the third scene starts the player is now in the dream world but still in his office where he again has to do some mini
tasks that have not been specified yet, but definitely clocking out and leaving the office via the elevator.

In the fourth scene the player first has to enter his house and follow a mostly strict order of tasks which trigger new events.
First he has to click on a painting hanging on the wall which will turn on the TV. The TV has to be turned off and after that the
player has to pick up a letter laying on the sofa which triggers chairs, that have been hanging on the ceiling all the time, to fall
down and a window to start slamming loudly. The player has to close the slamming window and is somehow led to the
bedroom where a women is laying in bed. He has to click on the women which will trigger an event where a women is slamming
on the window and the player has to escape into the restroom and pick up the restroom key and lock the door.
After that he has to run outside of the house which will lead to scene 5.

The fifth scene only consists of a cut scene where it is shown that Steve wakes up on his desk while his boss is yelling at
him. The cut scene slowly blacks out and the game is finished.

2.5.3. Object model
2.5.3.1. Data dictionary

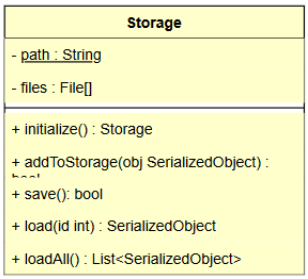
2.5.3.2. Class diagrams

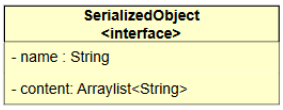
Backend for Getout

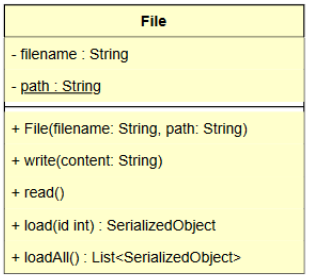
This document will describe how the game is designed in code.

Storage/Backend-System

implemented as
ngleton







2.5.4. Dynamic models

2.5.5. User-interface

3. Glossary