Requirements and Analysis Document for Adlez

Version:  
 Date   
 Michel Folkemark, Pontus Thomé, Viktor Engström, Martin So

[1 Introduction](#h.9tvg0y1rzbq)

[1.1 Purpose of application](#h.2uw8povj07kn)

[1.2 General characteristics of application](#h.21yy7mufwzqt)

[1.3 Scope of application](#h.nbr41nkxx67p)

[1.4 Objectives and success criteria of the project](#h.d2zz85bfnpx5)

[1.5 Definitions, acronyms and abbreviations](#h.32zsy3l5pgrz)

[2 Requirements](#h.9wa64jlaczj)

[2.1 Functional requirements](#h.9mbqwmefvrqc)

[2.2 Non-functional requirements](#h.xuu5y5gnl9z8)

[2.2.1 Usability](#h.w1aqdhe9g7bv)

[2.2.2 Reliability](#h.yd5xl449f4d0)

[2.2.3 Performance](#h.7xrif7i3nrrm)

[2.2.4 Supportability](#h.38e6hxib05z6)

[2.2.5 Implementation](#h.5jkbi3fbq1xq)

[2.2.6 Packaging and installation](#h.p7bkzc5ksat7)

[2.2.7 Legal](#h.b8x5tem5pe9a)

[2.3 Application models](#h.6f3shazdub5)

[2.3.1 Use cases](#h.z4wrb6nq66ze)

[2.3.2 Use cases priority](#h.w8k29oyx4lo)

[2.3.3 Domain model](#h.ryv78vwhz0r1)

[Adlez](#h.obi0yvtn1e8u)

[Character](#h.kwuf89bn6s9i)

[World](#h.xpnfxaiqmnnl)

[Object](#h.q4il60ml84jp)

[Item](#h.5e3k9mcqpkwr)

[2.3.4 User interface](#h.svc86kgqli2)

[2.4 References](#h.58l8tefurgso)

[APPENDIX](#h.ogvx309y804m)

[GUI](#h.gjpxxqhzq9sq)

[Use case diagram](#h.if6stmkvwsix)

[Domain model](#h.p9o6asnbw8ji)

[Use case scenarios](#h.fl50wo2301a4)

[Start new game](#h.jocvln2kxhmi)

[Move](#h.mm504j2zxwkp)

# 1 Introduction

Adlez is a project where the aim is to create a computer based 2D game which is inspired by the famous game, “Zelda: A link to the past”. The game will be a single player adventure RPG (role playing game) where you play as the main character and try to defeat your enemies.

The project is going to be implemented in the programming language, Java in an Object Oriented way.

## 1.1 Purpose of application

The application has to be simple to use and understand since it should be suitable for everyone. The game should be playable on almost all modern computers since it won’t require the latest hardwares. It will also be an offline game which means that you can play it even though you don’t have an internet connection, as long as the application already is installed on your platform.

## 1.2 General characteristics of application

The application will be compatible for desktop with the following platforms, Windows/Mac/Linux. No need for internet connection to play.

An adventure game in a fairly large world with one main hero. The hero fights enemies in his journey and gains money/experience and items from them which he then can use to gain new abilities and stats. The game will be very open, which means that you do not have to wait for any turns, you can freely walk wherever you want.

## 1.3 Scope of application

The application is a single player game with enemies and friendly characters that are controlled by the computer. It is possible to save the game so that the player later can continue from where he/she left of.

## 1.4 Objectives and success criteria of the project

The success criteria is for the user to be able to control the hero, explore a world, fight enemies, and gain new abilities and items.

1. The player should be able to control a character and move around in an area.
2. The player should be able to encounter non-friendly characters/creatures and fight them using an array of weapons/abilities.
3. The player should be able to visit several areas with varying appearances, such as caves, buildings etc.
4. The player should be able to encounter friendly characters and interact with them.

## 1.5 Definitions, acronyms and abbreviations

GUI, graphical user interface.

Java, platform independent programming language.

OOP, Object Oriented Programming, a very commonly used programming paradigm.

RPG, Role Playing Game.

NPC, Non Player Character.

...

# 2 Requirements

## 2.1 Functional requirements

**Game setup**

* A user should be able to:
  + start a new game
  + save the current game
  + load a saved game

**Character actions**

* A user should with their character be able to:
  + move in four directions
  + attack both melee and ranged
  + cast a spell or use a skill
  + pick up gold and items
  + equip items
  + open inventory
  + interact with NPCs, such as having a conversation, buy and sell items

**Enemies actions**

* An enemy should be able to:
  + locate the player’s character when the enemy is visible on the screen
  + attack and damage the player’s character
  + drop gold and items

## 2.2 Non-functional requirements

### 2.2.1 Usability

The starting screen of the game should be quite simple and allow the user to easily start a new game or continue from a previously saved game. It should also be possible to read instructions about the game’s controls as well as configure a few options for the game.

Inside the game, controls for moving and basic attacking will be implemented using the keyboard and should be easy to grasp. There will also be interaction using the mouse so that the player more efficiently can do things such as switch between different weapons/abilities as well as navigate in menus.

The game’s difficulty should be easy enough for most people to enjoy it, otherwise there should be an option to modify the difficulty.

### 2.2.2 Reliability

NA

### 2.2.3 Performance

The game should be able to run smoothly on most computers. It should react to input from the user rather instantly and only take time to load in cases such as starting the game or entering a new map area in the game.

### 2.2.4 Supportability

The game will be designed to run on a personal computer with Windows/Mac OS/Linux installed. Tests for all use cases should be available.

### 2.2.5 Implementation

The game will be implemented using the Java programming language as well as the game-development framework libGDX, which will allow the game to easily be deployed for different systems.

### 2.2.6 Packaging and installation

//Fyll i senare

### 2.2.7 Legal

//Fyll i senare

## 2.3 Application models

### 2.3.1 Use cases

What actors that can use which use cases can be seen in the use case diagram in the appendix, [Use case diagram](#h.if6stmkvwsix). Scenario of the use case **Start new game** can be seen in the appendix, [Start game](#h.jocvln2kxhmi). Scenario of the use case **Move** can be seen in the appendix, [Move](#h.mm504j2zxwkp). Sequence diagram of the use case **Move** can be seen in the appendix, [Move](#h.v50abi1qkvqd).

*Game setup*

1. Start new game
2. Save game
3. Load game

*Character*

1. Move
2. Attack
3. Melee attack
4. Range attack
5. Spell attack
6. Interact
7. Interact with object
8. Interact with NPC
9. Buy
10. Sell
11. Open inventory
12. Equip item
13. Use
14. Use item
15. Use ability

### 2.3.2 Use cases priority

*High*

* Start new game
* Move
* Melee Attack

*Mid*

* Range attack
* Spell attack
* Interact
* Interact with object
* Use
* Use item
* Use ability
* Equip item
* Open inventory
* Save game
* Load Game

*Low*

* Buy
* Sell
* Interact with NPC

### 2.3.3 Domain model

A graphical view of the domain model of the application can be seen in the appendix, [Domain model](#h.p9o6asnbw8ji).

#### Adlez

Represents the whole game and contains three objects, **Character**, **World** and **Object**.

#### Character

Represents a movable character in the game that are split into three different kinds of characters **Player** which is the character that the user will control, **Enemy** which is a character that will attack the **Player** and **Friendly character** which is a character that is friendly to the **Player**. The **Character** also has an inventory of **Items** and a **Position** in the **World**.

#### World

Represents the game world in which the **Characters** can move around in and the **Objects** are located in. The world is split into **Areas** such as **Building** and **Cave** that represents a certain kind of environment. The **Areas** are in turn split into **Positions** that represents a specific point in an **Area**.

#### Object

Represents a non-movable object in the game such as **Chest** and **Wall**. **Chest** also contains **Items** which the player can receive when interacting with the **Chest**.

#### Item

Represents an item that the **Player** can use and benefit from in some way. **Items** can be a **Weapon** which the **Player** can equip to do more damage, or an **Armor** such as **Boots** that the **Player** can equip to take less damage, or an **Usable item** that the **Player** can activate to gain some bonus such as recover health.

### 2.3.4 User interface

Graphical representations of the user interface can be seen in the appendix, [GUI](#h.6zjxdmjmvha1).

The game will at least have three main screens:

1. The starting screen: Will have buttons for starting the game along with buttons for options and instructions that may lead to subscreens.
2. The playing screen: Where the user actually plays the game. User interface elements such as a health bar, what weapons are equipped, a map etc. will be present.
3. The menu screen: Where the user will be able to do things such as save and/or quit the game as well as browse an inventory of weapons and items. (The inventory could possibly be integrated in the playing screen)

## 2.4 References

# APPENDIX Namnlös teckning (1).png

## GUINamnlös teckning (3).png

## Use case diagram

Use case diagram.png

## Domain model

Domain model.png

## 

## 

## Use case scenarios

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Start new game **Priority:** High  **Extends:** -  **Includes:** - Load game  **Participators:** Player  **Normal flow**  Starts the game   |  |  |  | | --- | --- | --- | |  | **Actor** | **System** | | 1) |  | Presents different options to start the game | | 2) | Clicks on new game |  | | 3) |  | Launches a new game |   **Alternate flows**  **Flow 2.1:** Player clicks on Load Game option   |  |  |  | | --- | --- | --- | |  | **Actor** | **System** | | 2.1.1) | Clicks on load game |  | | 2.1.2) |  | Launches Load Game screen containing saved games (if any) and a return option | | 2.1.3) | Clicks on a saved game |  | | 2.1.4) |  | Launches the chosen saved game |   **Flow 2.2.1:** Player clicks on return   |  |  |  | | --- | --- | --- | |  | **Actor** | **System** | | 2.2.1.1) | Clicks on return |  | | 2.2.1.2) |  | Sends the user to 1) | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Move **Priority:** High  **Extends:** -  **Includes:** -  **Participators:** Player  **Normal flow**  Move in direction of key pressed   |  |  |  | | --- | --- | --- | |  | **Actor** | **System** | | 1) | Player presses an arrow key |  | | 2) |  | Moves the player’s character in direction pressed |   **Alternate flows**  **Flow 2.1:** Player is next to an obstacle and moves towards the obstacle   |  |  |  | | --- | --- | --- | |  | **Actor** | **System** | | 2.1) |  | The character does not move but turns the player’s character in the direction pressed |   **Flow 2.2:** Player is next to a transition funneling point (cave entrance, door, etc.) and moves in that direction   |  |  |  | | --- | --- | --- | |  | **Actor** | **System** | | 2.2) |  | Loads the new area moved into |   //Eventuellt fler alternative flows, kan lägga till fler sen |

## Sequence diagrams

### Move

