Dragons Keep

Draft 1.0

9/19/2014

Revisions

| Version | Primary Author(s) | Description of Version | Date Completed |
| --- | --- | --- | --- |
| Draft 1.0 | Matthew Berger  Quan Tran  Brock Bearchell | This is the first version written of this requirements document. All requirements will be explained in detail for this application. | 9/19/2014 |

Contents

[1 Introduction 1](#_Toc520006015)

[1.1 Purpose 1](#_Toc520006016)

[1.2 Scope 1](#_Toc520006017)

[1.3 Definitions, Acronyms, and Abbreviations 1](#_Toc520006018)

[1.4 References 1](#_Toc520006019)

[1.5 Overview 1](#_Toc520006020)

[2 Overall Description 2](#_Toc520006021)

[2.1 Product Perspective 2](#_Toc520006022)

[2.1.1 System Interfaces 2](#_Toc520006023)

[2.1.2 User Interfaces 2](#_Toc520006024)

[2.1.3 Hardware Interfaces 2](#_Toc520006025)

[2.1.4 Software Interfaces 2](#_Toc520006026)

[2.1.5 Communications Interfaces 2](#_Toc520006027)

[2.1.6 Memory Constraints 3](#_Toc520006028)

[2.1.7 Operations 3](#_Toc520006029)

[2.1.8 Site Adaptation Requirements 3](#_Toc520006030)

[2.2 Product Functions 3](#_Toc520006031)

[2.3 User Characteristics 3](#_Toc520006032)

[2.4 Constraints 3](#_Toc520006033)

[2.5 Assumptions and Dependencies 4](#_Toc520006034)

[2.6 Apportioning of Requirements 4](#_Toc520006035)

[3 Specific Requirements 5](#_Toc520006036)

[3.1 External Interface Requirements 5](#_Toc520006037)

[3.2 Software Product Features 6](#_Toc520006042)

[3.2.1 Feature 1 6](#_Toc520006043)

[3.2.1 Feature 2 6](#_Toc520006043)

[3.2.1 Feature 3 6](#_Toc520006043)

[3.2.1 Feature 4 6](#_Toc520006043)

[3.2.1 Feature 5 6](#_Toc520006043)

[3.2.1 Feature 6 6](#_Toc520006043)

[3.2.1 Feature 7 6](#_Toc520006043)

[3.2.1 Feature 8 6](#_Toc520006043)

[3.2.1 Feature 9 6](#_Toc520006043)

[3.3 Performance Requirements 6](#_Toc520006047)

[3.4 Design Constraints 7](#_Toc520006048)

[3.5 Software System Attributes 7](#_Toc520006049)

[3.5.1 Reliability 7](#_Toc520006050)

[3.5.2 Availability 7](#_Toc520006051)

[3.5.3 Security 7](#_Toc520006052)

[3.5.4 Maintainability 7](#_Toc520006053)

[3.6 Logical Database Requirements 7](#_Toc520006054)

# Introduction

## Purpose

The purpose of this specification is to pinpoint specific requirements for a text based game software application. Furthermore, all background information for the software, functional and non-functional requirements, system and user interfaces, and much more information regarding the software will be explicitly stated. The intended audience for this document is a development team of software engineers/architects who will implement all specifications from this document into the software application.

## Scope

Product Name: Dragon’s Keep

The player will have to experience challenging puzzles, overcome difficult monsters, and reach the last level of the game and escape the dungeon of dragon’s keep. This is to promote a fun experience for the player that is rewarding and challenging to a certain extent. This is an RPG TBG where the player can have simulated battle scenarios with monsters where they can use items or special attacks at their disposal to vanquish their enemies. Furthermore, there will be a large map with many rooms that could contain these monsters or challenging puzzles to solve. The player will have to battle their way through the dungeon until they can reach the exit of the keep.

There could be an optional 2D-board showing the situation including positions of these doors, player, and monsters.

## Definitions, Acronyms, and Abbreviations

Java API – Application Programming Interface that is the documentation of class libraries used to program the game.

JRE – Java Runtime Environment

This is required to run a java application on a device. Contains all the packaged libraries and necessary programs to build and run applications for users. Can be downloaded at oracles home website for free use to the user.

DBMS – Database Management System

This is the storage system used for monsters, puzzles, rooms (game map), and user saved data in the game.

RPG – Role Playing Game

Fantasy game that involves a hero trying to accomplish a task, objective, or quest. Typically they have monsters, magic, expansive world or dungeon, and mythical elements that define the culture of the game. Players usually are involved in a type of combat system that is real time or turn based.

TBG – Text Based Game

All functionality the player has in the game is issuing text based commands to the console. They are then notified on the outcomes of their commands.

## References

Here are some examples of text based games found on the internet to **NOT PROVIDE REQUIREMENTS INFORMATION** but to help simulate the thought process and implementation of how this game will be developed and played by the users. These can be used as a visual aid.

<http://www.bbc.co.uk/radio4/hitchhikers/game_nolan.shtml>

<http://www.gamefools.com/onlinegames/free/zorktrilogy.html>

## Overview

The rest of this document will go into detail of background information for the software product as well as specific requirements. This will list all integrated systems, hardware specifications, user interfaces, performance requirements, specific game requirements, and much more information.

# Overall Description

## Product Perspective

Dragon’s Keep is a RPG TBG that will simulate a player controlled by the user traveling through dungeon rooms, battling monsters, solving puzzles, and exploring and finding items until they can escape the dungeon keep. The player will be able to enter text based commands as functional play that will have cause and effect on the game. This is a standalone java software application that interacts with a DBMS system to read in data for the game. There are no other related systems such as servers or web front end utilities that the game interacts with. This game will be independent of operating systems or network configurations due it being a standalone application.

### System Interfaces

DBMS will be used to house data for the game including the rooms of the dungeon, map holding all of the rooms, all monsters in the game, the hero’s saved data, and users who login to the game. The DBMS can be in the form of MySQL, SQLite, Access, and any other DBMS system that can be integrated with the game so long as it is not hosted on a Server for required network connection. Remember, it has to be a standalone game application.

### User Interfaces

The only user interface that the game will implement is a command line where the user can input text commands to execute the game. There will be a list of commands available to the user to enter in order to play the game, and a help command will list all commands available to the user. When commands are entered, the user is notified by the game of the results of their commands through text display messages in the command line interface. If a command did not exist, they must be notified as well. There is no other UI needed to be implemented, unless the team wants to add this addition to the game display character stats, inventory of character items, a map, or any other aspect of the game. Again, the UI is not required, but optional. The command line to enter commands IS REQUIRED and necessary for the player to play the game.

### Hardware Interfaces

There is no specific hardware required for this standalone java application. The point of it being standalone is that as long as the machine has the JRE installed (java runtime environment), the game can run on any OS regardless of hardware specs. As it being a text based game with a command line interface, no optimal graphics card is required nor is an optimal network card required to play this game. This game should run on any device including Windows, Mac, Linux, UNIX, etc. because of the JRE. The user only needs a keyboard to type in text commands and a monitor to view the game itself, which are two necessary hardware pieces to operate a machine in general.

### Software Interfaces

The game requires a DMBS system to house all the data of the game. This will allow the game to be changed solely on the change of data in the DBMS. The DBMS used is up to the development team, just as long as it adheres to the game being a standalone java application and not dependent on network systems. Furthermore, users playing the game must have java installed (JRE) to run java applications, which is the library of java source code documented in the java API.

### Communications Interfaces

There are no communications interfaces (networks, servers, etc.) required for this game. This is a standalone application independent of wireless networks.

### Memory Constraints

This is a small scale gaming application. There is not a set standard on this requirements document on how much hard disk space is needed to run the application nor is there a RAM requirements for memory processing due to the size of this application.

### Operations

The user will have to type in text commands in order to play the game. The user will also have a text command that will allow them to store their current data into a save slot under their user. That way they can exit the game and upon return load their data in and resume playing where they had left off. The game will continue running if left unattended and if exited on accident will ask for a saving option before closing out the game.

### Site Adaptation Requirements

The JRE must be installed on the machine to run the Dragon’s Keep game. You cannot run a java application if you do not install java on the machine.

Web Link: http://www.oracle.com/technetwork/java/javase/downloads/index.html

## Product Functions

* Storing and Reading in saved data and game data from DBMS.
* Read in and parse commands from a command line interface that execute the game
* Simulate randomized dungeon map with monsters and puzzles
* Implement puzzle solving situations
* Implement battle scenarios with monsters
* Allow player choice in item inventory, optional level system, and navigation of dungeon map

## User Characteristics

Users playing this game are in the age range of 6 and above. They can have very minimal technical and typing skills. This is meant for players that are literate in the English language as this product will display all text in English writing.

## Constraints

The only major constraint for this application is the user knowing which commands they can use in certain situations. That is why the game will need to implement a help command that displays all commands available to them in the game to use. This is a constraint regarding the user’s ability to navigate and play the game, not designing the game itself. The game should make it very convenient and easy for the player to understand which commands are available to them through the help menu.

## Assumptions and Dependencies

This application is being made assuming all users playing the game speak the English language, are playing the game by themselves and do not have required network connection.

## Apportioning of Requirements

The OPTIONAL requirement of implementing a very extensive UI for the user playing the game. The screen could show a game map as well as the players stats, inventory, and all player data. Furthermore, the interface could simulate graphics for the battle scenarios as well as the puzzle solving scenarios. Future editions of the game could implement the player having access to more than one player allowing a party system to create a more diverse combat system.

# Specific Requirements

## External Interface Requirements

The DBMS must be implemented in this game and house all game data to run the application. This includes the rooms in the game map, the puzzles, the monsters, and the hero for the player. When the application is started, the application must read in the data from the DBMS to populate the data for the game. Then the player can resume from where they left off or start a new game. The DBMS will be what controls the values in the game, so if it is changed then the game changes as well without having to rewrite the source code for the application. In regards to the input and output of the application itself, the game will implement a command line console for text input so the player can issue commands in the game. The results of the actions taken by the player will be reflected in a display message on the command line screen.

## Software Product Features

### Feature 1: Game Map

When the player starts the game, they will be welcomed with a message describing their scenario on the game map. The game map will contain rooms with a variety of puzzles, monstes, and items to explore. For Example:

“You enter the dragons keep. As you look around, you notice that the room has a monster.

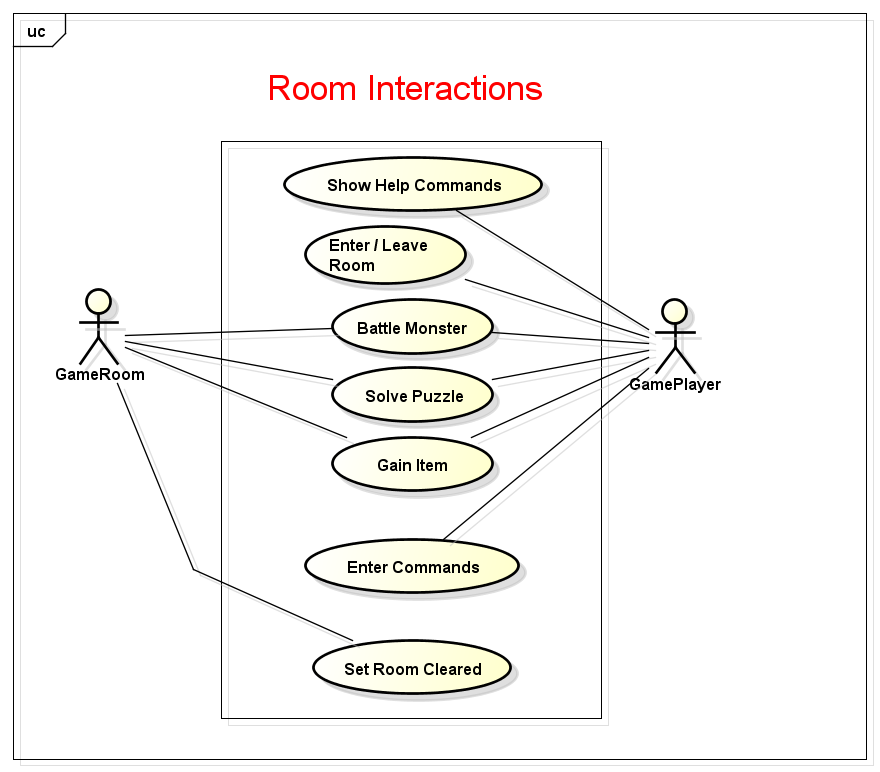
The monster approaches and the battle is initiated. The player has the first turn at combat, what will you do?”

The game map must contain a minimum of 50 rooms for the player to venture through. The rooms must be randomized when the player starts the game so the rooms are not the same through each play through. Each room can contain the possibility of a puzzle or a monster to fight with a minimum of 10 monsters and 10 puzzles in the entire game map. Puzzles and monsters do not have to be in the same room. When a puzzle is presented, a description of the puzzle and an acceptable solution to that puzzle must be provided for the player to progress further in the game. Puzzles should be decently challenging but have help and solution support for the player. For example:

“You step into a room and it is pitch black. You cannot see anything and have nowhere to go.” The user then types in “light torch”. “The light is now on and you see a treasure chest in the middle of the room. Do you wish to open it?”

When a monster is presented, a combat system is implemented simulating a turned based role playing game. When the monster is defeated, the user could be prompted to travel to a new room. For example:

“You have defeated the Pumpkin King. The room is now clear, what is your next destination?”

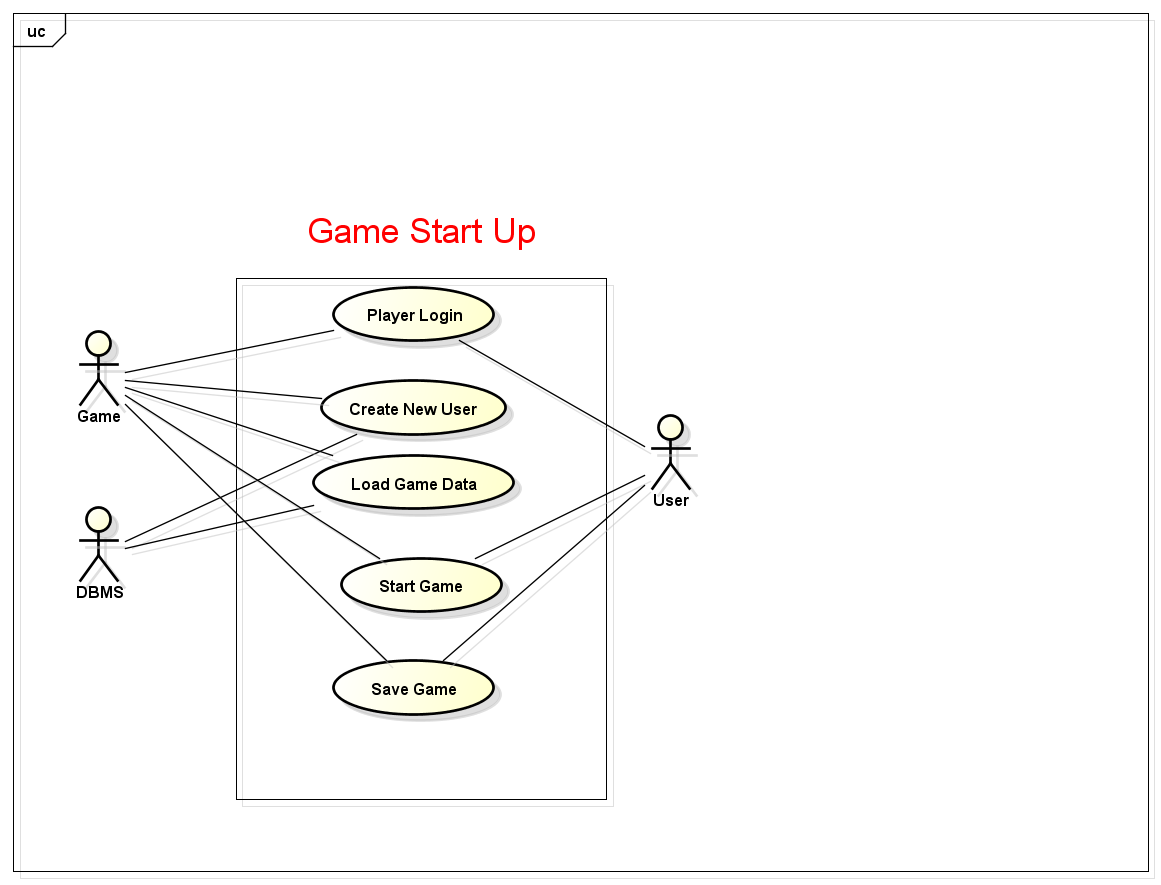


### Feature 2: Database Integration

All room interactions must be controlled through the database. Each room will have a description, defined exits, and a possibility of fighting a monster, solving a puzzle, or obtaining an item for use. This data will be read in from tables in the DBMS. By changing the values in the database, the amount of monsters, puzzles, rooms, and types of rooms available in the game map will change. This is required so that the game can be changed by the changing of the database. The database must also store dialog and test messages to notify the user of the games progress and happenings. If you execute a command, the player needs immediate feedback.

For Example: The player happens upon a monster and tries to use an irregular command. The response will be “This action cannot be implemented at this time. Try using the help menu to enter a correct command.”

Furthermore, the DBMS must store users and their login information so their saved data can be read in to resume where they left off in the game. The users actual saved data can be written to and read in from a file or saved in the DBMS as well, that is up to the development team. It may be easier to just store users in the DBMS and then have an associated file to each user to store game data.



### Feature 3: Commands

The user must be able to request a list of commands to use as part of a help system. When the help command is given, all commands they can issue at that point in the game will be shown to them so they can have clues as to what they can do next. The player may have to solve certain puzzles or obtain items in order to use certain commands in the game. The development team can implement a UI to always display help commands but this is their choice and not required. Commands are used in order to play the game. The player must type in a command in order to do something. For Example: The player types “use potion” when battling a monster and the player’s health points is increased with a display message saying “You used a health potion to restore health points. Your current health points is *currentHP”.*

### Feature 4: Randomization

The player will have a map at their disposal to tell where they are. However, all the rooms they encounter will not show what they have until they enter the room. Rooms will have a monster, puzzle, item, or nothing and these components are randomly placed into the game map of rooms when a new game is made. The rooms are not randomized, just their contents, which will make saving the game and resuming much easier to implement. Furthermore, the boundaries of the map will be set so the player cannot skip over the bounds of the map.

### Feature 5: Creation System

An adventure creation system must be provided to allow the user to create and edit the current game information in the database. For example, the user can add a puzzle or a monster to the game with a String value that will be the terminator for progressing. For example: The player types in the command “edit game”. They can then type “enter monster” and then allow the user to create a monster with a help display of what should be typed in to add to the DBMS. For Example: “Dragon 12 30 Firebreathing dragon”

### Feature 6: Player Functionality

Players will be able to solve puzzles, fight monsters, take and inflict damage, equip and use items including armor, potions, and scrolls and weapons. This will be important for the combat mechanics of the game. Players can have abilities like attack, use item, equip armor, use special attack, etc. Basically, the character will have equipment that will add bonuses to base stats like strength and health points. Furthermore, they will obtain an inventory of items to use against special monsters in the game. These two concepts will be applied to a turn based simulated battle system between the player and an opposing monster. The player will be notified who is attacking, the results of the attack, and all battle information as it is simulated. Besides battle, the player will type text commands to navigate the map by choosing north, south, east, and west to go to the next room. For Example: “You have defeated the monster. Which direction do you want to travel?” They enter South. “There is no room in this direction”. They enter north. “The dungeon door opens and the player ventures north”. If they have already been in this room, they will be notified by “This room looks very familiar and there is nothing of interest here.” When the player comes across a puzzle, they can type a text command that will solve the puzzle. For Example: “The room is dark and you cannot see. You hear groaning in the distance and are unaware of the dangers ahead.” The player types “light torch”. “The room is decorated with rays of burning light from your torch. How do you wish to proceed?”

### Feature 7: Saving Data

Players must be able to save and return to the game and resume where they left off. The game must recognize different users and when they return to the game, allow them to resume where they left off. The player could also start a new game if they so choose. Players will be able to save the game when they have solved a room or have entered a room they have already solved. They can only save in a “safe place”, meaning not when they are solving a puzzle or in combat with a monster, and have to obtain an item from a treasure chest if the room contains an item before they can save. An example command would be “save game”. The player’s game data can be saved in the DBMS or written to a file depending on how the team wants to implement this requirement.

### Feature 8: Tracking Player

The game should be able to track the player’s score and rooms that have been visited. If a player has solved a puzzle or defeated a planned monster, they should not have to do this again. The If a player enters a room they have already solved, they should be notified that the room is safe and empty. The player’s score will be updated based on how many monsters they defeat, how many puzzles they solve, and how many items they obtain in their play through. Each time they achieve one of these three categories, they can be rewarded a score point. At the end of the game, the player needs to be notified of their score.

### Feature 9: Inventory System

The game must track the items in the player’s inventory. There should be an upper limit to the number of items a player can carry causing them to have to make decisions about items to carry. This includes the inventory of items they can use as commands at their disposal against monsters or puzzles as well as the equipment items their character has equipped to increase base stats. This includes armor and weapons or any other piece of equipment.

## Performance Requirements

The game is meant for only one user to play at a time and they can interact with only one command terminal to type in text commands to play the game. When the user types in a command, the game should run the command and simulate within 5 seconds maximum. On startup for the application, the user should login or create a new user and start the game within 15-30 seconds maximum. There is not a lot of data and records to be read from the database so processing time should be short.

## Design Constraints

None of the game data can be hard coded into the application. All game data must be read in from a database allowing the game to be maintained easily or changed just by changing records in the database.

## Software System Attributes

### Reliability

The game should not crash because a wrong command was entered or a battle scenario was not computed correctly. Proper error handling with display messages telling the user valid input is necessary and required for this application. The game should be reliable in a sense that the user is not afraid to experiment with commands and have fun exploring the game without crashing the application.

### Availability

The application is available to only one user at a time as it is independent of online play and only supports one user at a time.

### Security

The user can only make changes in the database with the commands they are given in the help list. The user will be limited to the changes they can make protecting the game data from getting wiped or changed dramatically for malicious intent by the user.

### Maintainability

The game will be maintained easily through the DBMS system implemented in the game. All changes to the game for data regarding the monsters, rooms, map, items, etc. can be made in the DBMS system. That way the game functionality is independent and doesn’t have data hard coded into source code.

## Logical Database Requirements

The database must have valid data for game monsters, rooms, and all other tables housing records for the game. Monsters must have integer values for statistical attributes representing health points, strength, defense, speed etc. The same is for the hero (user player). Rooms must have valid descriptions identifying aspects of the room and what the room contains being a monster, puzzle, item, or nothing. Puzzles must have valid record data to solve the problem. The text commands that are associated with certain items, puzzles, and monsters must be stored in the DBMS and linked correctly with Primary and Foreign keys. That way the game will be able to change the data in the DBMS and still run the same allowing the player to change the data to change the game. If a text command such as “use item scroll” is used against a Pumpkin King monster to defeat that special monster, it needs to be tied to the Pumpkin King monster in the database to properly implement the design of the game.