1. **Introduction**
   1. **Purpose of the system**

Dread of the Evil Wizard aims to provide a unique gaming experience with presenting a timeless concept with a new infrastructure, a brand new implementation for a text based RPG. The main purpose of this game is to save the sister of our main character from the evil wizard. The way to defeat the evil wizard is completing the missions given by the characters in the game and fighting the monsters and become victorious along the way.

* 1. **Design goals**
* **Usability:** Text based nature of Dread of the Evil Wizard provides a very user friendly and easy to understand game for our players. The explanations regarding storyline, items, game dynamics, creatures, powers and quest goals provided in the help page using plain English. No special interface experience or knowledge about special key combinations are needed since the game is text based but player needs to know the valid commands provided in the help section.
* **Robustness:**  To have a reliable infrastructure Dread of the Evil Wizard especially command interpreter must differentiate valid and invalid commands entered by the user and only execute the valid commands. Command validation depends on the current situation of the character for example attack commands outside of battle mode ignored by default. Also while loading game if the save file is missing or corrupt game starts with default settings since dying results with deletion of the save file game is already compatible with missing files. These properties of File Manager increase the fault tolerance of the game.
* **Extensibility:** Additional characters or quests can be added or modified in the game later on. In order to achieve this, creating a good documentation regarding the inner dynamics and object models of the game was necessary. This documentation would minimize the compatibility problem may occur extending the features of the game.
* **Efficiency:** The throughput is limited to only one move at a time (since the game is turn based) so calculations and responses are not pose a big problem and to make the game more efficient the map itself (objects that are non-interactive or stays still) is not rendered but displayed as an image in the form of background then the objects that are dynamic is rendered by the game on top of this background. This method must be applied for both battle mode and regular game mode.
* **Portability:** The Java implementation provides a platform independent game where every system compatible with running Java software would be able to run this game successfully.

**Trade-offs:** Some trade-offs needed to be decided to implement these design goals that listed below.

* **Efficiency vs. Portability:** By implementing the system with Java we obtain a portable game which would be functional in any system capable of running Java but less efficient compared to beforehand compiled languages like C/C++.
* **Space vs Speed:** Handling the actions with two separate components with Battle Manager and Map Manager instead of just one. This eats up more space but runs more quickly because of the purpose built nature of the separate components.
* **Functionality vs. Usability:** The game provides the player with options about classes, character attributes, status affects etc. This options affect the player’s choices regarding the selection of a character type, a skill or an attack to be used in an arbitrary moment of the game. To simplify the learning curve relating the matter number of these factors must not overwhelm the user but simultaneously present a diversity of options to maximise the gaming experience.

1. **Software Architecture**
   1. **Subsystem Decomposition**
   2. **Hardware/Software Mapping**
   3. **Persistent Data Management**
   4. **Boundary Conditions**
2. **Subsystem Services**