Bilkent University

Department of Computer Engineering

**CS319 Project**

*Project short-name: Animal Uprising*

Analysis Report

Bora Ecer, Ata Gün Öğün, Albjon Gjuzi, Tanay Akgül

Instructor: Bora Güngören

Progress

October 7, 2017

This report is submitted to the GitHub page in partial fulfillment of the requirements of the Object Oriented Software Engineering Project, for course CS319.

Contents

[1. Introduction 3](#_Toc495171223)

[2. Proposed System 3](#_Toc495171224)

[Overview 3](#_Toc495171225)

[Hero 3](#_Toc495171226)

[Soldiers 3](#_Toc495171227)

[Enemies 4](#_Toc495171228)

[Skills 5](#_Toc495171229)

[Upgrades 5](#_Toc495171230)

[Functional Requirements 6](#_Toc495171231)

[Play Game 6](#_Toc495171232)

[Settings 6](#_Toc495171233)

[How To Play 6](#_Toc495171234)

[Non-functional Requirements 6](#_Toc495171235)

[Pseudo Requirements 6](#_Toc495171236)

[System Models 6](#_Toc495171237)

[Scenarios 6](#_Toc495171238)

[Use-Case Model 7](#_Toc495171239)

[1) Object and Class Model 10](#_Toc495171240)

[2) Dynamic Models 10](#_Toc495171241)

[3) User Interface 10](#_Toc495171242)

[3. Glossary 10](#_Toc495171243)

[4. References 10](#_Toc495171244)

Analysis Report

*Project short-name: project title*

# Introduction

Animal Uprising is a different variation of the basic Castle Wars game, in which the main goal is destroying the enemy castle with the soldiers that the player summons with in-game currency, food. The game will be a desktop app and will be controlled with keyboard.

# Proposed System

## Overview

Animal Uprising is a strategy/adventure game in which the player will be able to control a hero rather than having a castle. During the game, in order to destroy the enemy castle, the player will have to carve a path to the enemy castle by defeating the enemy soldiers. The player can summon ally soldiers in order to have a bigger striking force towards enemies soldiers. Once the player and allied soldiers reach to the enemy castle, they can attack it to tear it down. The player hero will be able to cast both defensive and offensive skills which either help the allies or harm the enemies. The game has five different difficulty levels and in each level, the player has to face more and harder types of enemy soldiers. Upon finishing each level, and destroying all enemies soldiers the player will gain coin which will allow the player to buy items, and upgrade skills or soldiers. The player will be able to pause the game and continue whenever they want to. Also, the player will be able to replay completed levels to gain more coins. There will also be an option to reset the whole game from scratch in order to set a higher score. The game will be played with the keyboard and mouse, specific buttons will be assigned for moving the hero, summoning different soldiers and casting skills. The mouse will be used only on the shop and the main screen.

### **Hero**

The hero is the main character and the only character the player is able to control.. Since, the hero will be the one who is summoning soldiers and casting skills, it should have the in-game currency, food and mana. At first, when the game is initially started, both food and mana will be zero. Also, they will have upper limits so the player cannot reach high amounts of mana and food.

* HP: 1000
* Mana upper limit: 100
* Food upper limit: 150

### **Soldiers**

There will be four different soldiers which can be summoned by the player. All of the soldiers will have a certain role, and in total there will be three different roles: melee, ranged and tank.

* Melees:
* Dog: the basic unit, it has low health and low damage but the summoning it requires less food.
* HP: 75
* Damage: 15
* Food: 20
* Bear: compared to the dog, bear has more HP and more damage but summoning it requires more food.
* HP: 150,
* Damage: 50
* Food: 50
* Ranged:
* Monkey, is the only ranged unit of the game, it has less health but it offers ranged attack, so if the frontline is strong enough it can provide more damage.
* HP: 50
* Damage: 20
* Food: 20
* Tank:
* Tortoise: the only tank unit, the tortoise does not offer any damage output but it can absorb the damage done by the enemies.
* HP: 800
* Damage: 0
* Food: 100

### **Enemies**

In the game, there will be three different types of enemy soldiers

* Infantry:
* HP: 100,
* Damage: 20
* Knight:
* HP: 200
* Damage: 25
* Crusader:
* HP: 400
* Damage: 50

### **Skills**

The hero will have four different spells. Each spell can be casted if the player has enough mana required to cast. These spells will have two different types: Offensive spells and defensive spells.

1. Offensive Spells:

* Raven Strike: The hero calls upon a pack of ravens to attack the enemies.
* Mana: 20
* Hail: the hero summons a hail of rocks to hit the enemies
* Mana: 40

1. Defensive Spells:

* Heal: Heals the nearby allies.
* Mana: 40
* Heal amount: 50HP
* Speed Buff:
* Mana: 50
* Unit Speed Increase: %20

### **Upgrades**

After completing a round successfully or defeating an enemy soldier, the player will gain certain amount of coins. With coins the player can choose to buy new items or upgrade skills and soldiers.

* Items: \*TO DO\*
* Skills: There will be some types of skills with which our hero can attack the enemies soldiers.
* Offensive Spells: Increase amount
* Defensive Spells: Increase amount
* Upgrade Soldiers: The soldier upgrades will increase the damage done and the health of the soldier which is chosen to be upgraded.

## Functional Requirements

### **Play Game**

### This is our main functional requirement since it consists of the main screen of the application in which the player will be controlling the main hero and trying to conquer as many enemy castles as possible.

### **Settings**

Initially, the player will be provided a default game setting. From the settings menu, the player will be able to change them according to his own wishes. These settings will be:

* Sound Level
* Sound On/Off

### **How To Play**

In case the player has never played this type of game, he/she can simply open how to play in order to understand the main idea of how the game is actually played, and what are its components. We are thinking of a video tutorial, but in case that will be very difficult to implement we will simply insert a mockup of the game at the moment it is being played.

### **Pause Game**

While the player is playing the game he can simply stop it anytime he wants in order to go to the shop, restart the level or restart the whole game.

### **Shop**

In the shop the player can buy new items with the coins he will earn during levels, update his allies or update his heroes skills.

### **Restart Level**

The player can restart the level whenever he wants after pausing it for different reasons,

### **Toggle Sound**

The player can toggle the sound effects on or off depending on his preferences. The sound effects are activated at different stages of the game such as killing an enemy or winning/losing the round(level).

### **Toggle Music**

The player can toggle the music sounds on or off depending on his preferences too. The music is simply to make the game less boring but it can be turned off at anytime from settings menu.

## Non-functional Requirements

* We do not promise the best design ever, but at least we will try to implement our app in a way that it reacts fast enough to user commands,
* Game inputs will be common for the user like in most of the games, but maybe we will offer them the option to change them in case they think they can play better by using different keys from keyboard.
* The game implementation will provide us the opportunity to add many more levels, skills, allies or items because if we implement it for 5 levels, there will be no problem to make it 10 or 15 or even 20.

## Pseudo Requirements

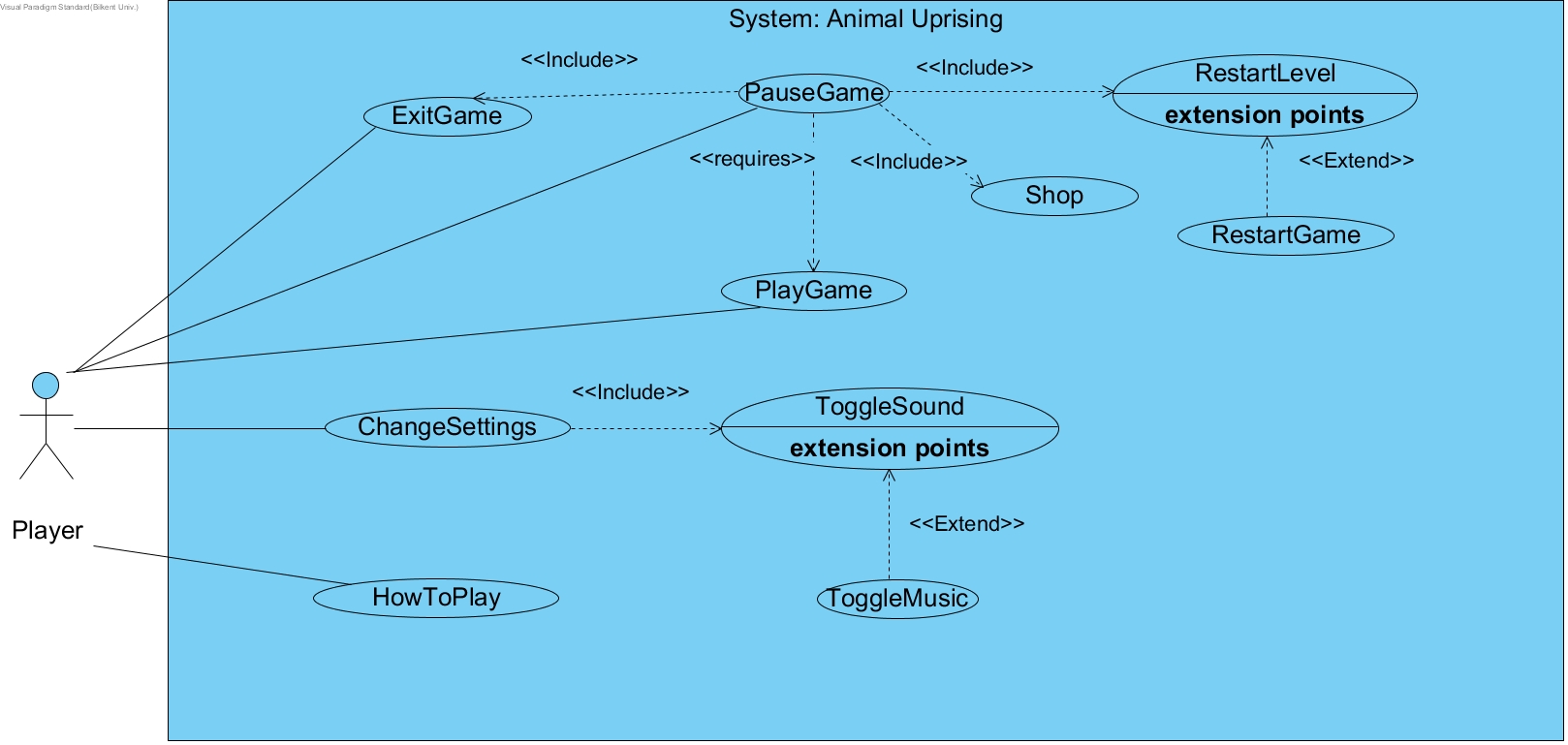
1. The game will be implemented using Java
2. We will use Adobe Photoshop to design the characters and the environment of the game.

## System Models

### **1) Scenarios**

### **Use-Case Model**

This section provides the main use case model of Animal Uprising game and detailed use case information.

****

**Figure 2.5.2 – The use-case model of Animal Uprising**

Use Case #1

**Use case name:** PlayGame

**Primary Actor:** Player

**Entry Condition:** Player has opened the game.

**Exit Condition:**

· If player completed all the levels of the game

· If player choses Pause Game or Exit Game

· If player lost all her or his lives.

**Main Flow Of Events:**

1. Player choses the level of the game

2. The system sets up the difficulty of the game

3. Player finishes all the levels

4. Player returns the main menu.

**Alternative Flow Of Events:**

· Player can lose all his or her lives at any level (go to step 4)

· At any time player can exit the game (go to step 4).

Use Case #2

**Use case name:** ExitGame

**Primary Actor:** Player

**Entry Condition:** Player is already in the game.

**Exit Condition:**

· If player presses X from the game window OR

· If player pauses the game and selects exit OR

· If player chooses exit without even starting to play.

**Main Flow Of Events:**

1. Player presses exit button during the game

2. The system displays exit the game.

**Alternative Flow Of Events:**

· Player can press pause button then exit the game .

Use Case #3

**Use case name:** PauseGame

**Primary Actor:** Player

**Entry Condition:** Player is already in the game.

**Exit Condition:**

· If player continues to play game OR

· If player chooses to restart the level of game OR

· If player chooses restart game

· If player exits from the game OR

· If player shops items to upgrade his/her skills.

**Main Flow Of Events:**

1. Player press the pause button during the game

2. The system displays pause the game.

3. Player continues the game.

**Alternative Flow Of Events:**

· Player can pause the game then exit the game

· Player can pause and shop then continue the game

· Player can pause and restart level then continue the game

· Player can pause then restart the game.

Use Case #4

**Use case name:** Shop

**Primary Actor:** Player

**Entry Condition:** Player is already in the game and paused the game.

**Exit Condition:**

· If player continues to play game OR

· If player restarts the game OR

· If player restarts the level OR

· If player exits the game.

**Main Flow Of Events:**

1. Player pauses the game during the game

2. Player presses the shop button and buys some items to upgrade his/her skills.

3. The system displays shopping.

4. Player continues to play.

**Alternative Flow Of Events:**

· Player can shop then exit the game

· Player can shop then restart level or restart game.

Use Case #5

**Use case name:** RestartLevel

**Primary Actor:** Player

**Entry Condition:** Player is already in the game and paused the game.

**Exit Condition:**

· If player exits the game OR

· If player returns to play game OR

· If player restarts the game.

**Main Flow Of Events:**

1. Player pauses the game then presses restart level button

2. The system displays restart level

3. Player starts over again from that level.

**Alternative Flow Of Events:**

· Player can restart level then exits the game

· Player can restart level then shop and continue to play the game

· Player can restart level then restarts the game.

Use Case #6

**Use case name:** RestartGame

**Primary Actor:** Player

**Entry Condition:** If player is already in the game and paused the game.

**Exit Condition:**

· If player exits the game OR

· If player returns to play game.

**Main Flow Of Events:**

1. Player pauses the game then presses restart game button

2. The system displays restart game

3. Player starts a new game.

**Alternative Flow Of Events:**

· Player can restart the game then exit the game

· Player can restart the game then shop and continue to play the game.

Use Case #7

**Use Case Name:** Change Settings

**Primary Actor:** Player

**Entry Condition:** Player selects “Settings” from main menu.

**Exit Condition:** Player returns to the menu.

**Main Flow of Events:**

* 1. Player selects “settings” to change or view game settings.
  2. The system displays game settings to the player.
  3. Player changes the settings according to his/her own choice.
  4. System updates the game settings.

**Alternative Flow of Events:**

1. Player wants to return to the main menu.
2. Player pushes the “Back” button
3. Player returns to the main menu.
4. Player wants to restore the default settings.
5. Player selects “Default Settings” to restore default settings.
6. The system updates the game settings to default.

Use Case #8

**How to Play**

**Use Case Name:** How to Play

**Primary Actor:** Player

**Entry Condition:** Player selects “How to Play” from main menu.

**Exit Condition:** Player returns to the main menu.

Use Case #9

**ToggleSound**

**Use Case Name:** ToggleSound

**Primary Actor:** Player

**Entry Condition:** Player selects “Change Settings” from main menu.

**Exit Condition:** Player returns to the main menu.

**Main Flow of Events:**

1. Player selects “Change Settings” from the main menu OR
2. Player selects “Change Settings” from pause menu
3. The system displays the options to the player.
4. The player can then toggle the sound effects.

**Alternative Flow of Events:**

**a.** Player wants to return to the main menu.

1. Player pushes the “Back” button
2. Player returns to the main menu.

Use Case #10

**ToggleMusic**

**Use Case Name:** ToggleMusic

**Primary Actor:** Player

**Entry Condition:** Player selects “Change Settings” from main menu.

**Exit Condition:** Player returns to the main menu.

**Main Flow of Events:**

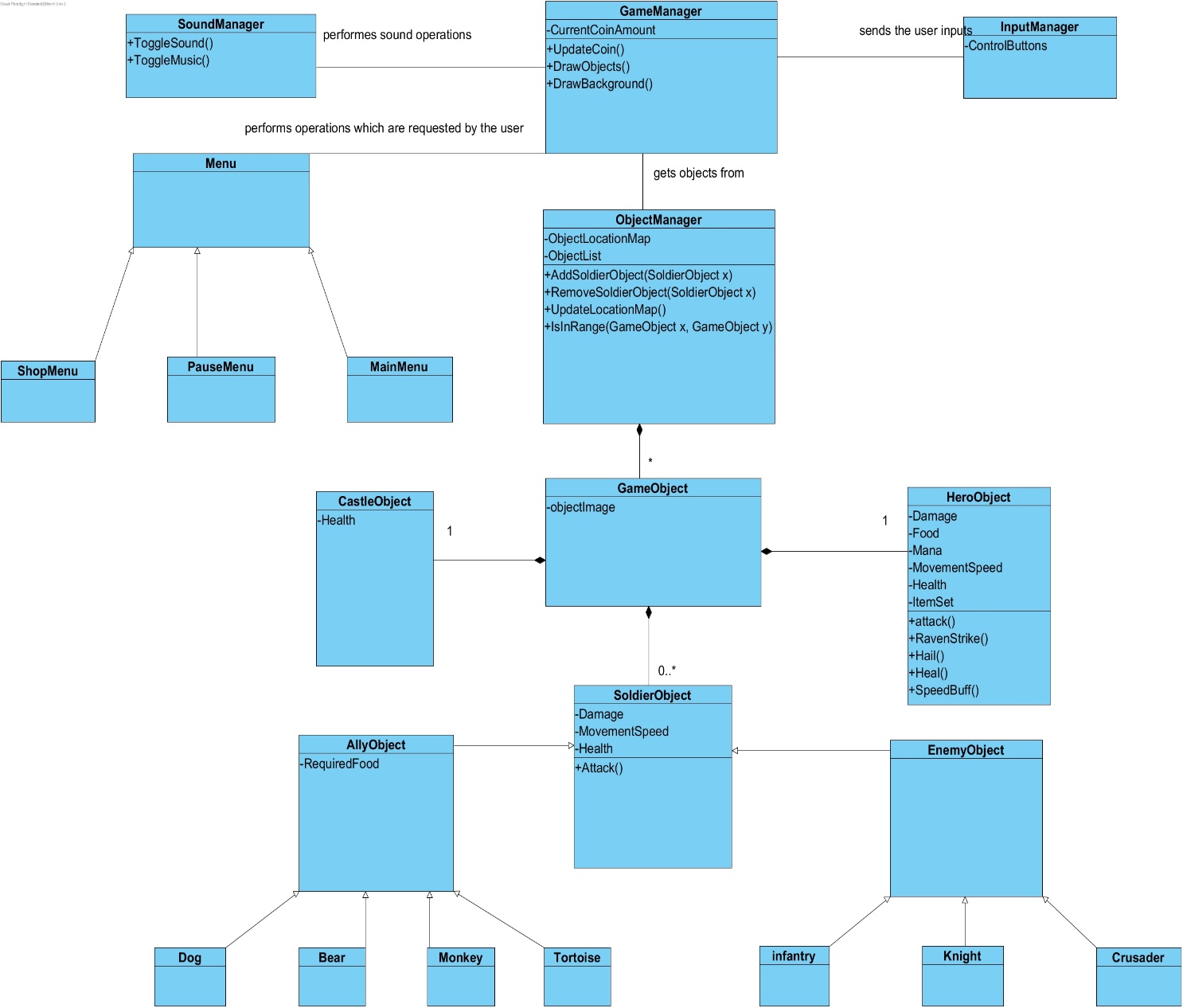
1. Player selects “Change Settings” from the main menu OR
2. Player selects “Change Settings” from pause menu
3. The system displays the options to the player.
4. The player can then toggle the music sounds.

**Alternative Flow of Events:**

**a.** Player wants to return to the main menu.

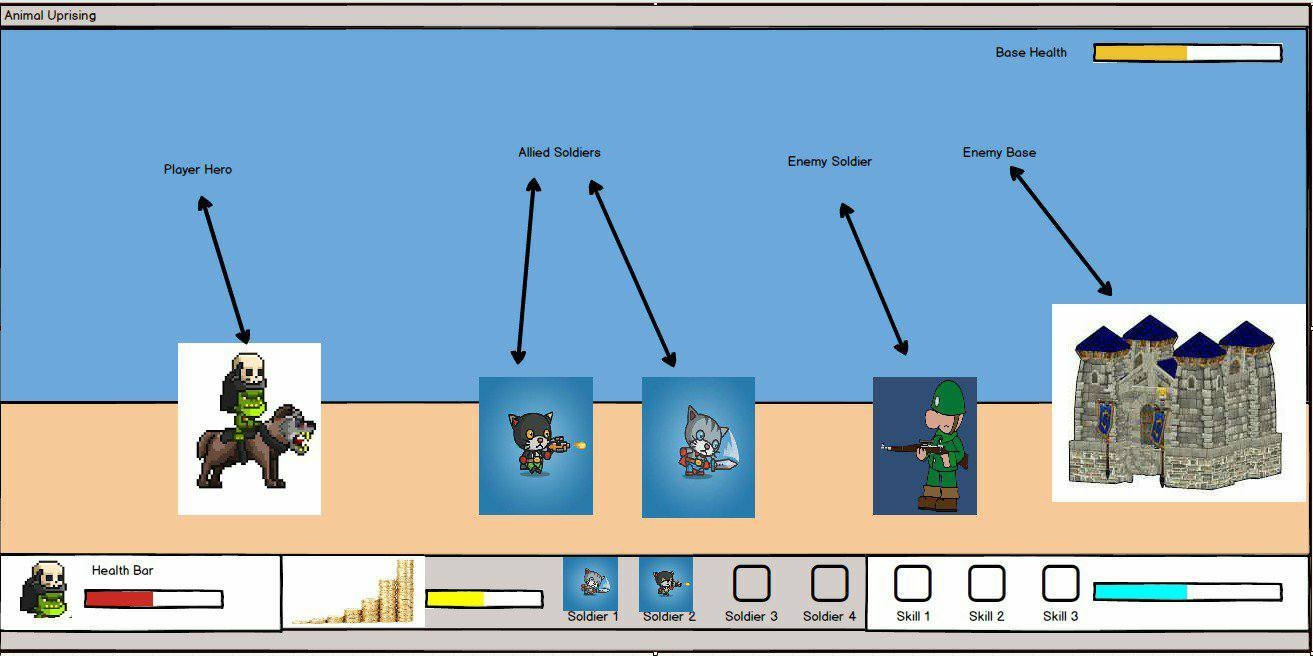
1. Player pushes the “Back” button
2. Player returns to the main menu.

### **Object and Class Model**



### **Dynamic Models**

### **User Interface**



# Glossary

Glossary for any domain-specific terms you use in your report.

# References

1. Object-Oriented Software Engineering, Using UML, Patterns, and Java, 2nd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2004, ISBN: 0-13-047110-0.