Object Oriented Software Engineering Project \_ April 2017

Project: Development of a computer system for a Minecraft Game System

Students:

|  |  |  |
| --- | --- | --- |
| Name | Student ID | Email |
| Vinit Date | x16135270 | x16135270@student.ncirl.ie |
| Carl Mohn | x16136799 | x16136799@ student.ncirl.ie |
| Rajeeva Revankar | x16137124 | x16137124@student.ncirl.ie |

# **Project Scope**

## **Project work to be accomplished**

The main objective of this project is to develop a “**Sample Software Engineering Model**” of a computer system for a “**Minecraft Game System”**. The game is intended and should be suitable for gamers over age of 5 years. The model would be developed using Object Oriented Concepts. The concept would be presented using Unified Modelling language(UML).

## **The purpose this project?**

The project work being undertaken as part of academic presentation for “**Object Oriented Software Engineering Module-2017”** being studied for the course, “Higher Diploma In Science In Computing HDCBIBM”.

Since this is an academic project with time restrictions only a sub-section of Minecraft is in the project scope.

## **Project Stakeholders**

The project is presented to National College of Ireland by students,

Vinit Date

Rajeeva Revankar

Carl Mohn

## **Exemptions / Out of Project Scope**

There will be no Software Implementation of the project. All material of this project is for academic purpose only.

# Objectives

## Technical objectives

|  |
| --- |
| * Demonstrate the conceptual, practical and technical skills of planning and monitoring a project plan using an appropriate CASE tool |
| * Demonstrate an understanding of [Agile Development](https://moodle.ncirl.ie/mod/resource/view.php?id=10277) |
| * Describe in detail the theory, concepts and methods pertaining to the Unified Modelling Language ([UML](https://moodle.ncirl.ie/mod/url/view.php?id=10226)). |
| * Create requirements using use case modelling concepts. |
| * Demonstrate conceptual and technical skills in the analysis, design and implementation of a software system using Object Oriented Concepts. |
| * Employ tools and techniques for Object Oriented Software Engineering, |
| * Demonstrate an ability to adapt and solve problems in software development activities from specification to [testing](https://moodle.ncirl.ie/mod/resource/view.php?id=10273) individually and as part of a team. |

## Schedule objectives

* Project team formed during wk6 of current semester of this academic year.
* Plan is to develop a “**Sample Software Engineering Model**” of a computer system for a “Minecraft Game System”.
* Project is to be submitted on or before end of week 13 of the module.

# Deliverables

The Project deliverables are listed below. It can be noted that group deliverables are highlighted in GREEN and individual deliverables in BLUE.

1. [Group] Identify the actors.
2. [Group] Construct a Use Case Model.
3. [Individual] Describe in detail any use case from the use case model. The use case must contain an alternate flow.
4. [Group] Create a Project Plan to deliver your application. The plan must include a minimum of 3 nesting levels and include all the major tasks and deliverables.
5. [Individual] Create a conceptual class diagram of the chosen use case. The conceptual class diagram should demonstrate the use of many of the following: attributes, relationships, navigability, association class, multiplicity and composition.
6. [Individual] Create a glossary that lists and defines all the terms that require clarification.
7. [Individual] Draw a System Sequence diagram from the conceptual class diagram.
8. [Individual] Develop Contracts for a minimum of two of the system operations in the system sequence diagram.
9. Individual] Draw Communication diagrams based on the contracts. The communication diagrams should demonstrate the use of design patterns.
10. [Group] Presentation (how well does the package of models look?).
11. [Group] Use of a UML tool.
12. [Group] Put together a Testing Plan outlining how you propose to validate the application and verify that it is free of defects.

## Success Factors

* Demonstrate the conceptual, practical and technical skills of planning and monitoring a project plan using an appropriate CASE tool
* Describe in detail the theory, concepts and methods pertaining to the Unified Modelling Language (UML).
* Create requirements using use case modelling concepts.
* Demonstrate conceptual and technical skills in the analysis, design and implementation of a software system using Object Oriented Concepts.

# Implementation

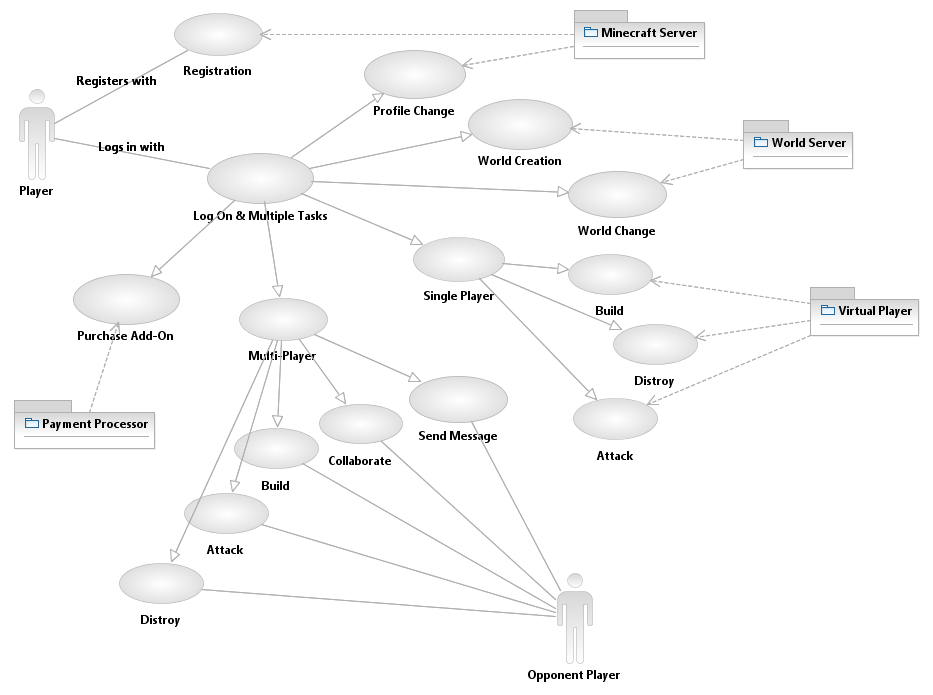
## Implementation (Deliveries)

### Actor Identification

* Player
* Minecraft Server
* World Server
* Virtual Player
* Opponent Player
* Payment processor

### Construct a Use Case Diagram ( High Level)

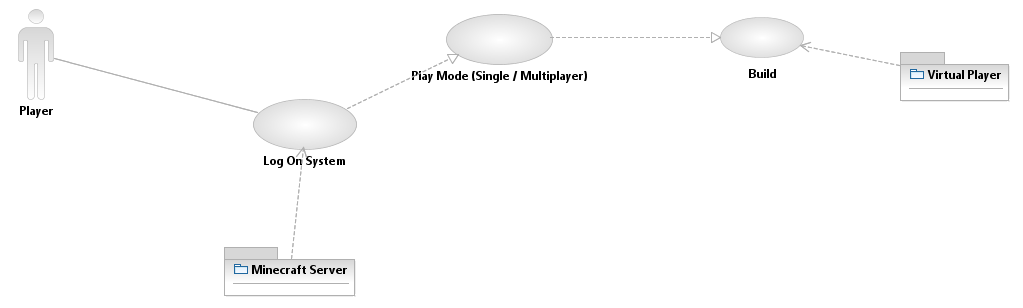
Based on earlier step of identifying actors and use cases, the Use Case Diagram has been plotted using IBM’s Rational Modeler Application tool.



### Describe Use Cases (Bird’s eye view)

* Player Registers with Minecraft server
* Player Logs in, with Minecraft server.
* Player Logs in, the player has option to Create or Update world with World Server
* Player Logs in, Start Single Player Game Build/Destroy/Attack with
* opposite Virtual player.
* Player Logs in, starts multiplayer game Cooperate/Build/Destroy/Attack/Send Message with opponent player
* Player logs in, buys add-on

### Detailed Use Case Diagram for “Build” use case (Contributor: Vinit Date)



#### Actors

* Player
* Virtual Player
* Minecraft server (passive actor, needed for authentication)

#### Pre-Conditions

* The player is logged into the system.

#### Flow of Events

* The player selects to build object
* The object starts getting built.
* The resources needed for object build are deducted from player inventory as the object is being built.
* Object building is finished

#### Post-Conditions

* Render surrounding world with finished object

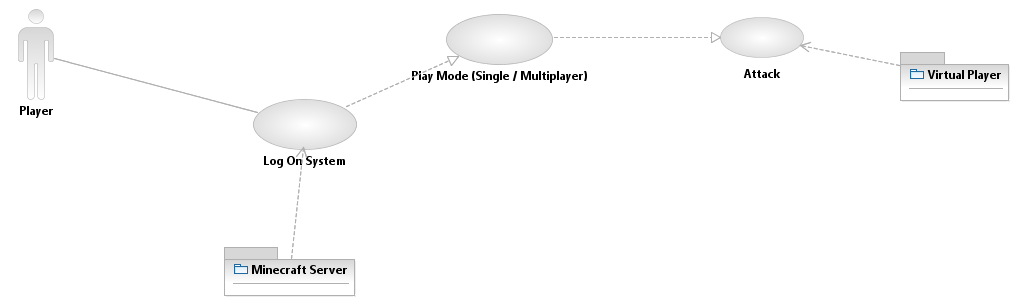
#### Alternative Flow of Event

* The player selects to build object
* Object starts to get built
* Not enough resources are found, Object building is stopped as is

#### Post-Conditions

* Render surrounding world with unfinished object

### Detailed Use Case Diagram for “Attack” use case (Contributor: Rajeeva Revankar)

****

#### Actors

* Player
* Virtual Player
* Minecraft server (passive actor, needed for authentication)

#### Pre-Conditions

* The player is logged into the system.

#### Flow of Events

* The player selects attack
* Cost of attack is deducted from player inventory.
* Object/s in surround are damaged

#### Post-Conditions

* Render surrounding world with damaged object/s

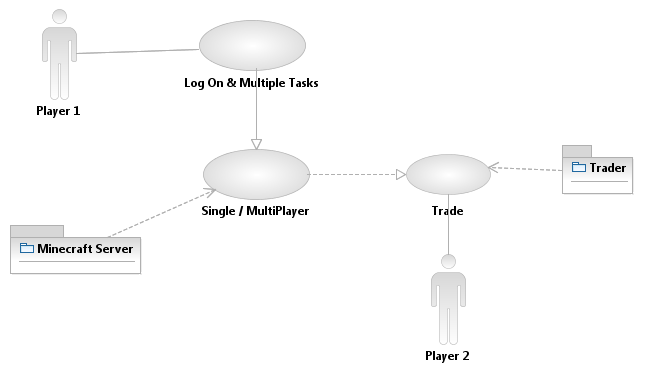
#### Alternative Flow of Event

* The player selects attack
* Cost of attack is deducted from player inventory
* Object/s in surround are indestructible e.g. sky / water

#### Post-Conditions

* Render surrounding world same as before event.

### Detailed Use Case Diagram for “Collaborate” use case (Contributor: Carl Mohn)



#### Actors

* Player x
* Player y
* Shop
* Minecraft server (passive actor, needed for authentication)

#### Pre-Conditions

* The player is logged into the system.
* The player has none or more objects in the inventory

#### Flow of Events

* The player targets another player and selects Trade, two inventory windows will pop up
* The player clicks and drags objects with the controller to or from the inventory window of one player to the inventory window of the other player
* Both players have to select “Accept” to complete the transaction, or click “Cancel” to cancel the transaction

#### Post-Conditions

* Adjust player inventory, objects to transfer from source to destination
* Adjust player inventory weight by addition or subtraction of resource weight

#### Alternative Flow of Event

* The player targets a Shop and selects Trade, two inventory windows will pop up
* The player clicks and drags objects with the controller to or from the inventory window of one player to the inventory window of the other player
* Both players have to select “Accept” to complete the transaction, or click “Cancel” to cancel the transaction

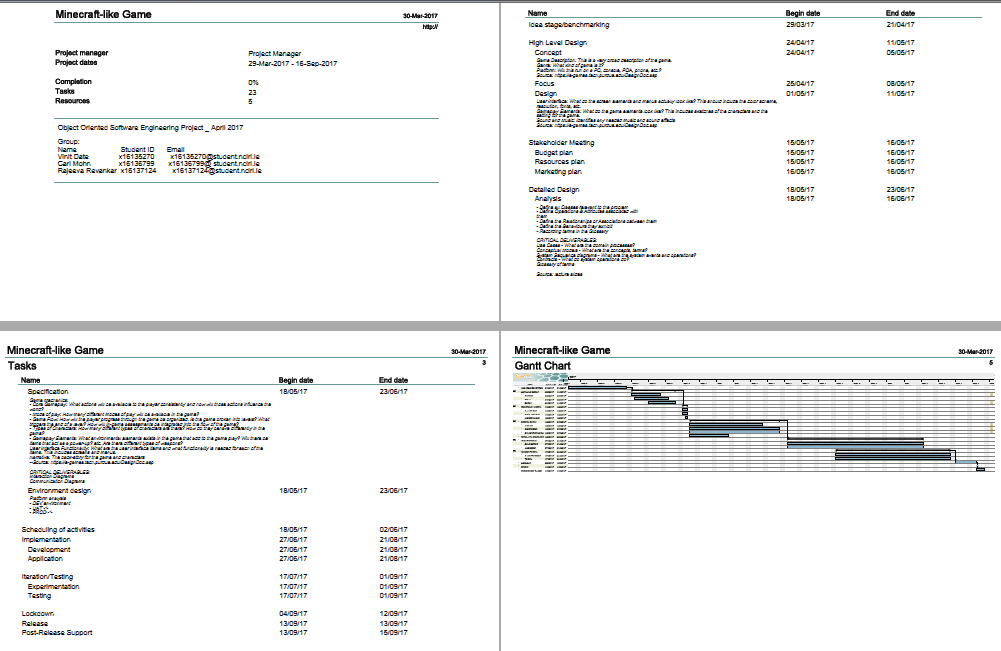
#### Post-Conditions

* Adjust player inventory, objects to transfer from source to destination
* Adjust player inventory weight by subtraction of resource weight
* Adjust player inventory, money is either added (items sold) or subtracted (items purchased)

## Deliverable Timeline

### Project Plan

For the full project plan, please refer to the attached PDF file Minecraft\_projplan\_VCR.pdf



# **Testing and Acceptance Criteria**

For each use case acceptance criteria is as

## Player Register with Minecraft server

#### Capture and Validate Name, Address, Date of Birth and email for new user.

#### Validate Address for country availability.

#### Validate Date of Birth

#### Validate email

## Player Logs in with Minecraft server.

#### Login is not locked or expired.

#### If login fails option to login alternatively example retrieve / change password or use OTP

#### If paid account option to top op

#### If free account invite to change to pay model.

## Player Logs in, The player has option to Create or Update world with World Server

#### Can create world

#### Can update world

## Player Logs in Start Single Player Game Build/Destroy/Attack with opposite Virtual player.

#### A virtual player can be created

#### Player can play with virtual player.

## Player Logs in Starts multiplayer game Cooperate/Build/Destroy/Attack/Send Message with opponent player

#### Can Cooperate/Build/Destroy/Attack/Send Message with opponent player

## Player logs in buys add on

#### Payments can be processed for valid user to buy add on.

# **Glossary**