TERM PROJECT SOFTWARE DESIGN REPORT

CSE 344 - SOFTWARE ENGINEERING

Spring 2022

Project Name: Golden Railways

May 13, 2022

Şeyda Gürsel
Defne Su Özbudak
Merve Çalışkan
Efe Ömer Erdem
Fatih Tutcu
Mert Celikan

Yeditepe University
Faculty of Engineering
Department of Computer Engineering

TABLE OF CONTENTS

1. INTRODUCTION

1.1. Purpose of the Document

The main purpose of this document is to describe the structure of our software system meeting the requirements which we specified in the Analysis Report. In this document, we described the components of our software system and the component relationships clearly. We added all the necessary information for programmers so that they can use this design to implement the software system.

1.2. Purpose of the System

In the system. Users can easily open and play anywhere, since it is playable on the phone. It also provides a daily and fun game with an easy-to-understand interface.

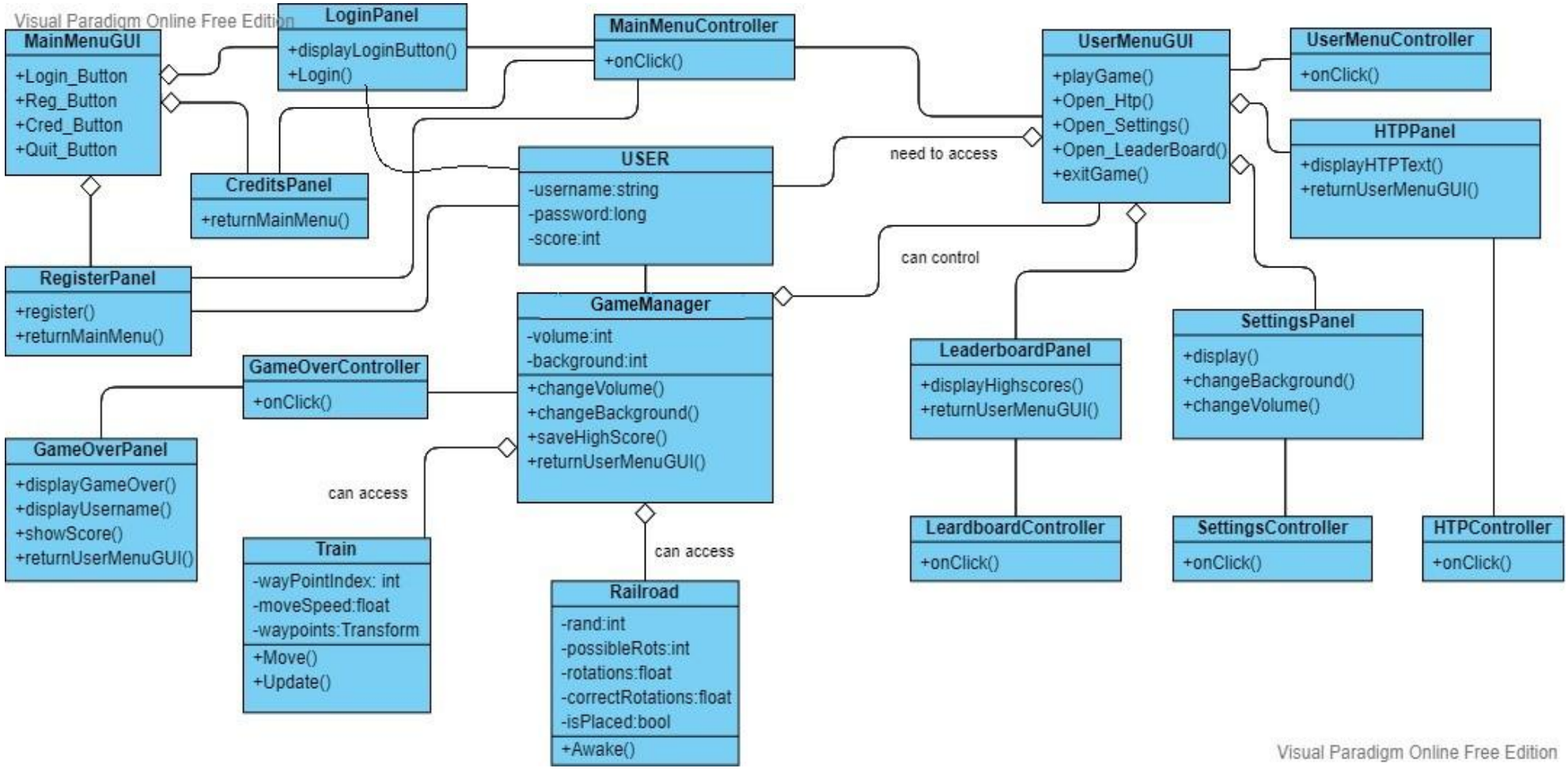
1.2.1. The New System

In addition to the system we mentioned in the analysis report, in the new system, with the contributions of the VCD team, the font and size were changed, some sound effects were added and the visual images in the game were developed in order to increase the game's impressiveness for the user. The visual narratives of the game have been strengthened in general.

1.3. Structure of the Document

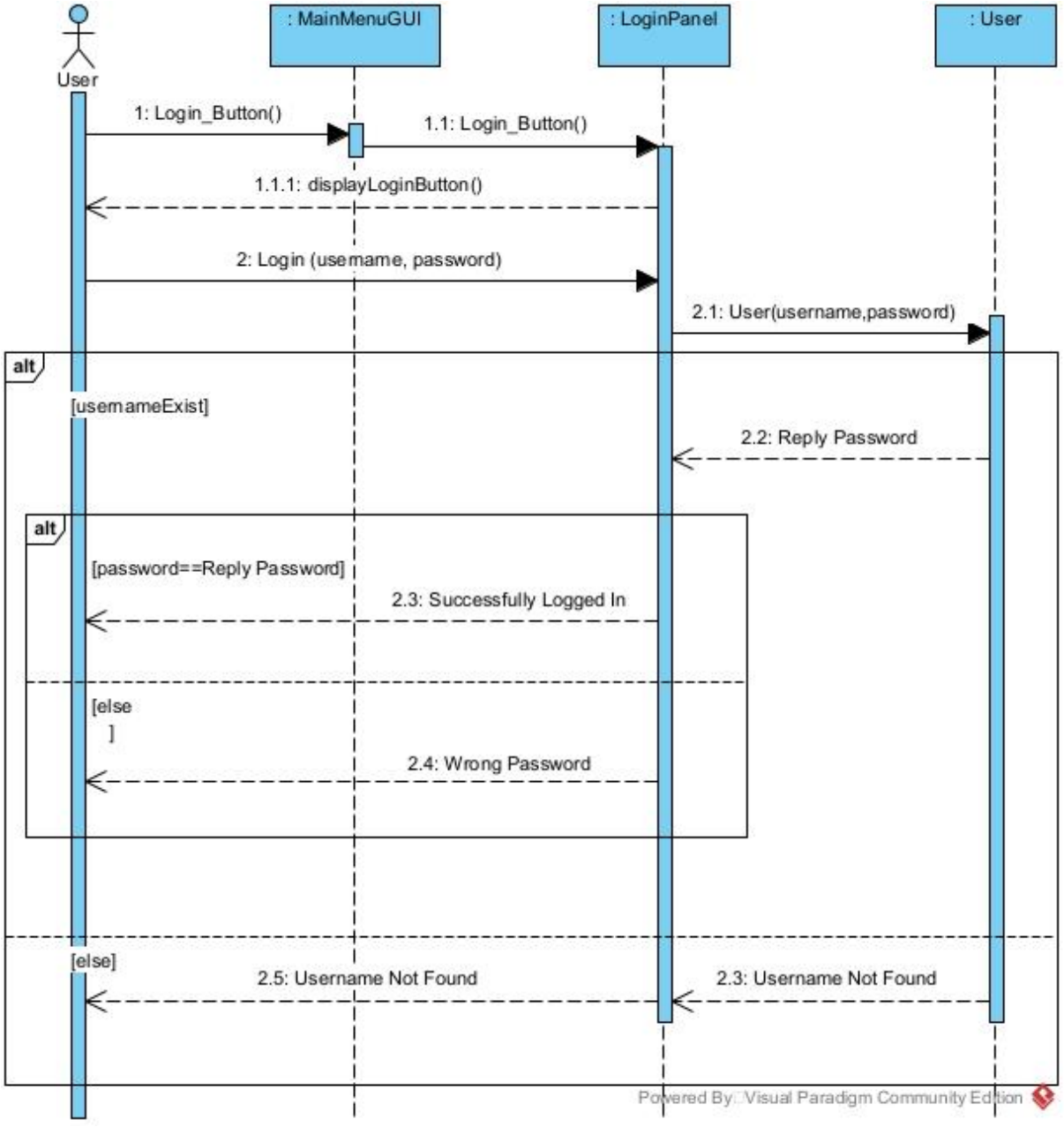
This software design report consists of six sections, some of them with subsections. In the *Introduction* section, we explained the purpose of the document and the system and briefly described our system in terms of its users and the features provided to them. In the *Detailed Design Class Diagram* section, we provided a refined and corrected form of our previous class diagram. We added all the attributes, methods, associations, aggregations and relationship names etc to the class diagram. In the *Dynamic Models* section in order to show the object interactions, the behavior of the system and operations between components, we designed the sequence, state and activity diagrams based on our use case specifications in the analysis report. In the *Software Architecture* section, we designed the UML package and component diagrams to represent the hierarchy of our software design system. In the *Entity Relationship Diagram* section, we showed the relationships between the system components. In the last section of the document, we included the glossary and references.

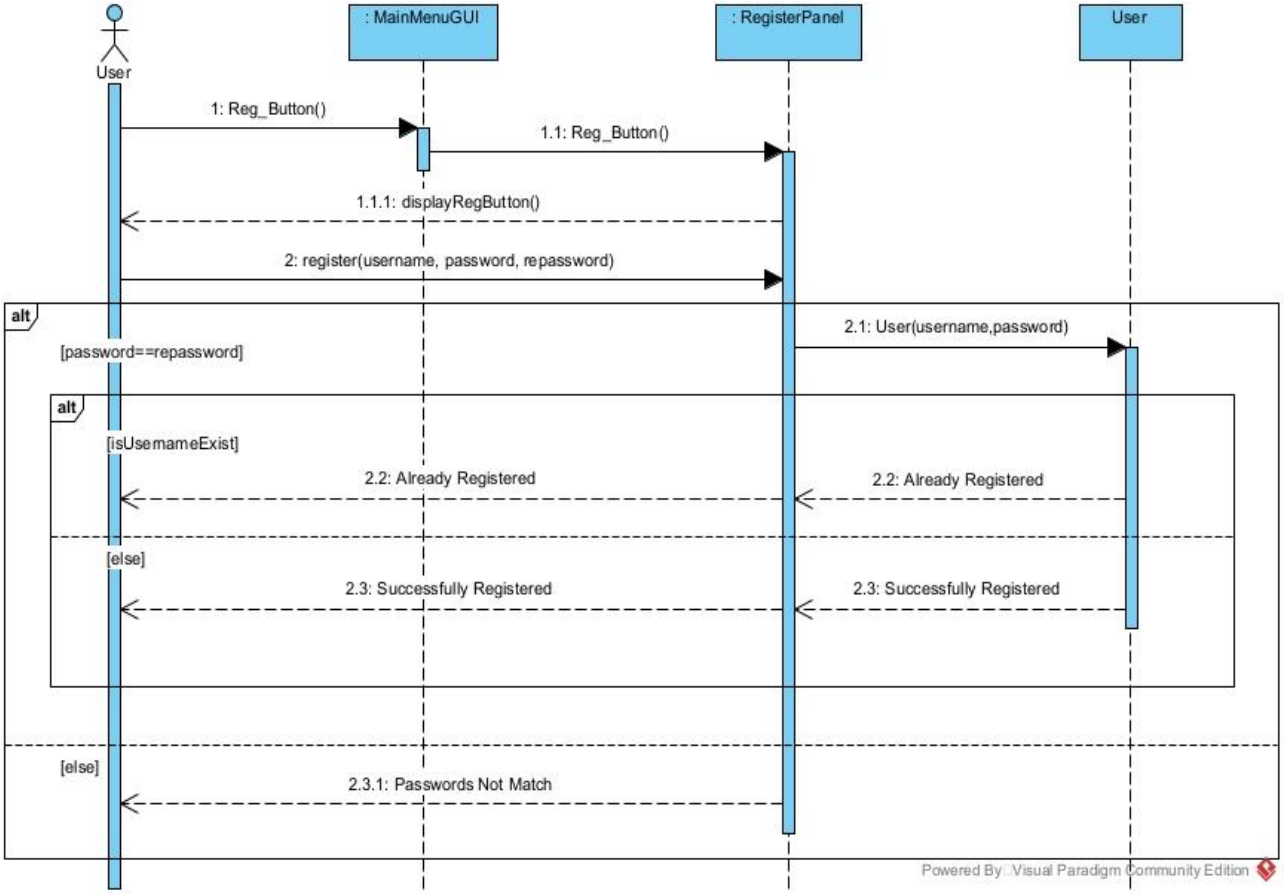
2. DETAILED DESIGN CLASS DIAGRAM

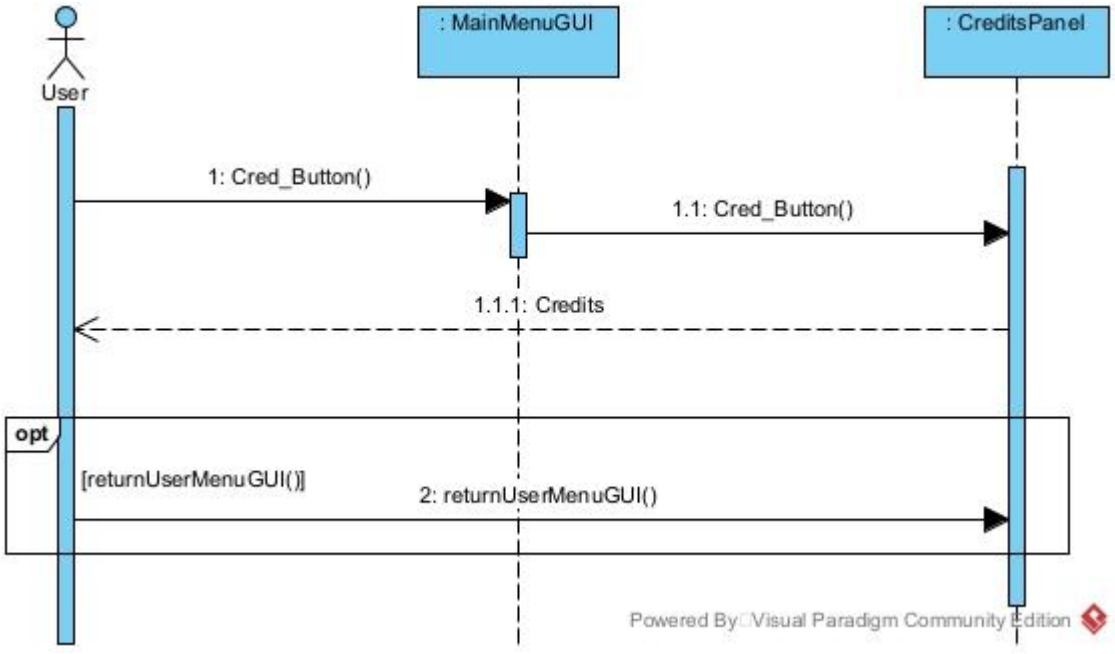


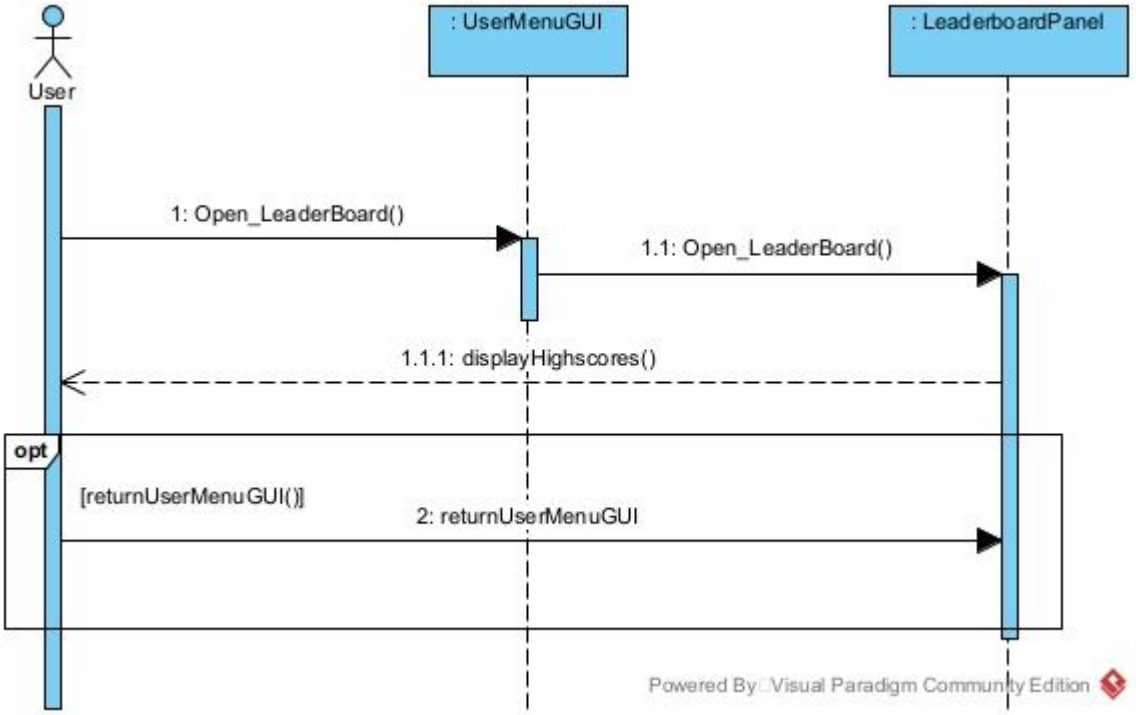
3. DYNAMIC MODELS

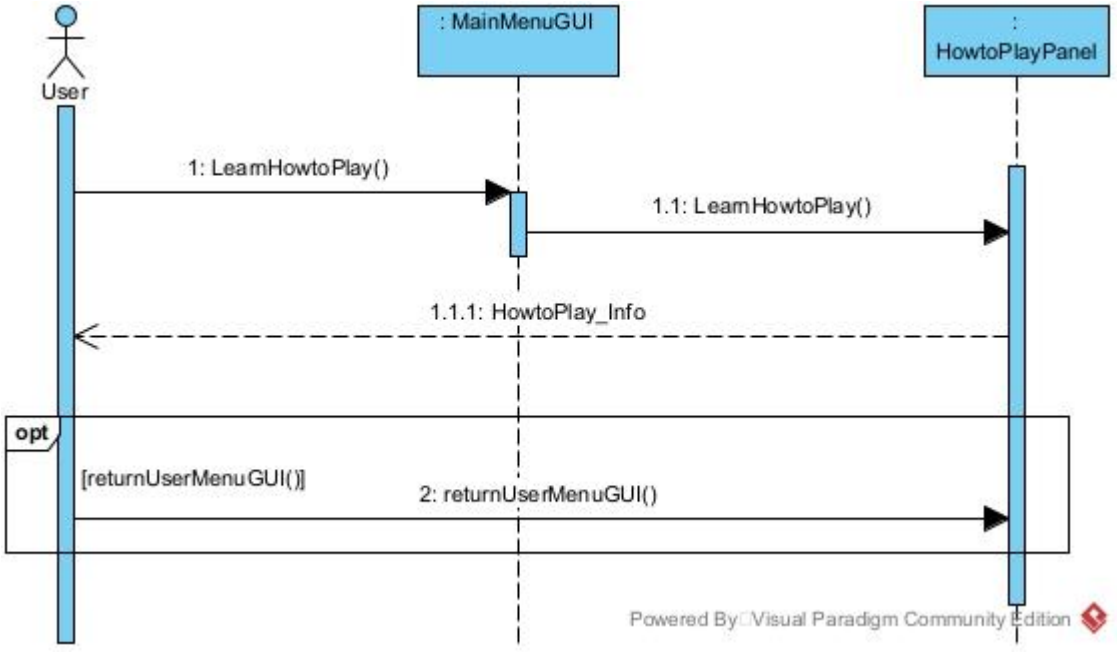
3.1. Sequence Diagrams

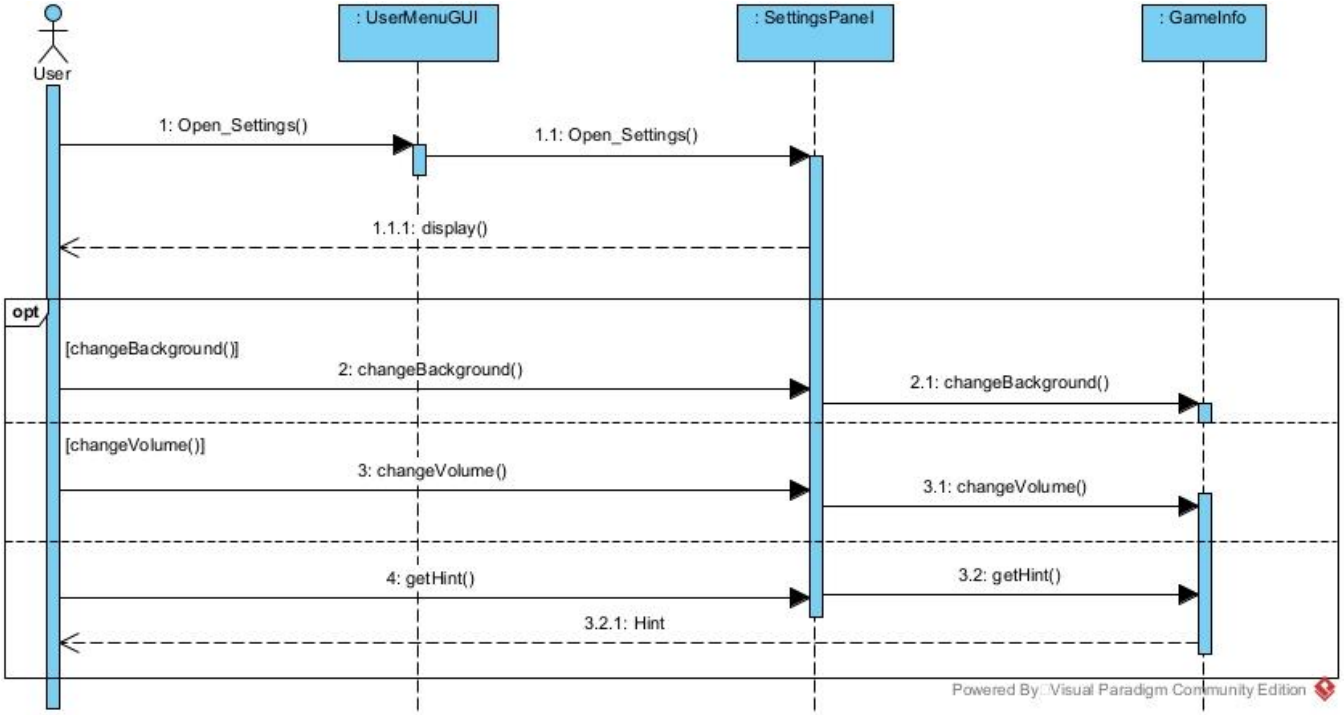


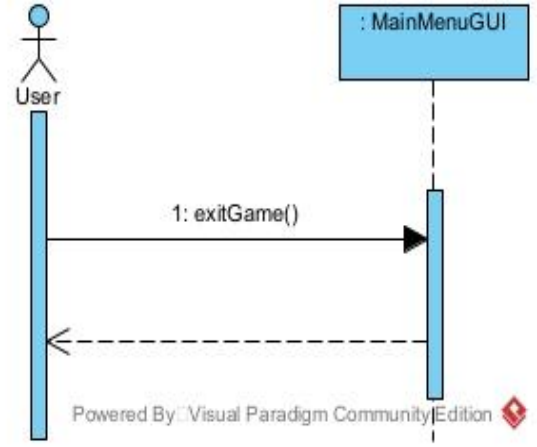

User
: MainMenuGUI
: RegisterPanel
User
1: Reg_Button()
1.1: Reg_Button()
1.1.1: displayRegButton()
2: register(username, password, repassword)
alt
[password==repassword]
2.1: User(username,password)
alt
[isUsernameExist]
2.2: Already Registered
2.2: Already Registered
[else]
2.3: Successfully Registered
2.3: Successfully Registered
[else]
2.3.1: Passwords Not Match
Powered By: Visual Paradigm Community Edition

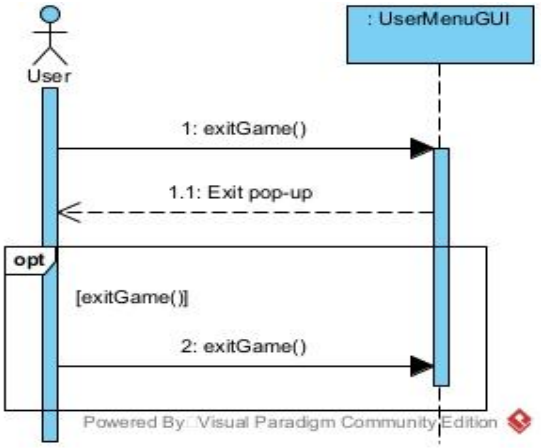

User
: MainMenuGUI
: CreditsPanel
1: Cred_Button()
1.1: Cred_Button()
1.1.1: Credits
opt
[returnUserMenuGUI()]
2: returnUserMenuGUI()
Powered By: Visual Paradigm Community Edition

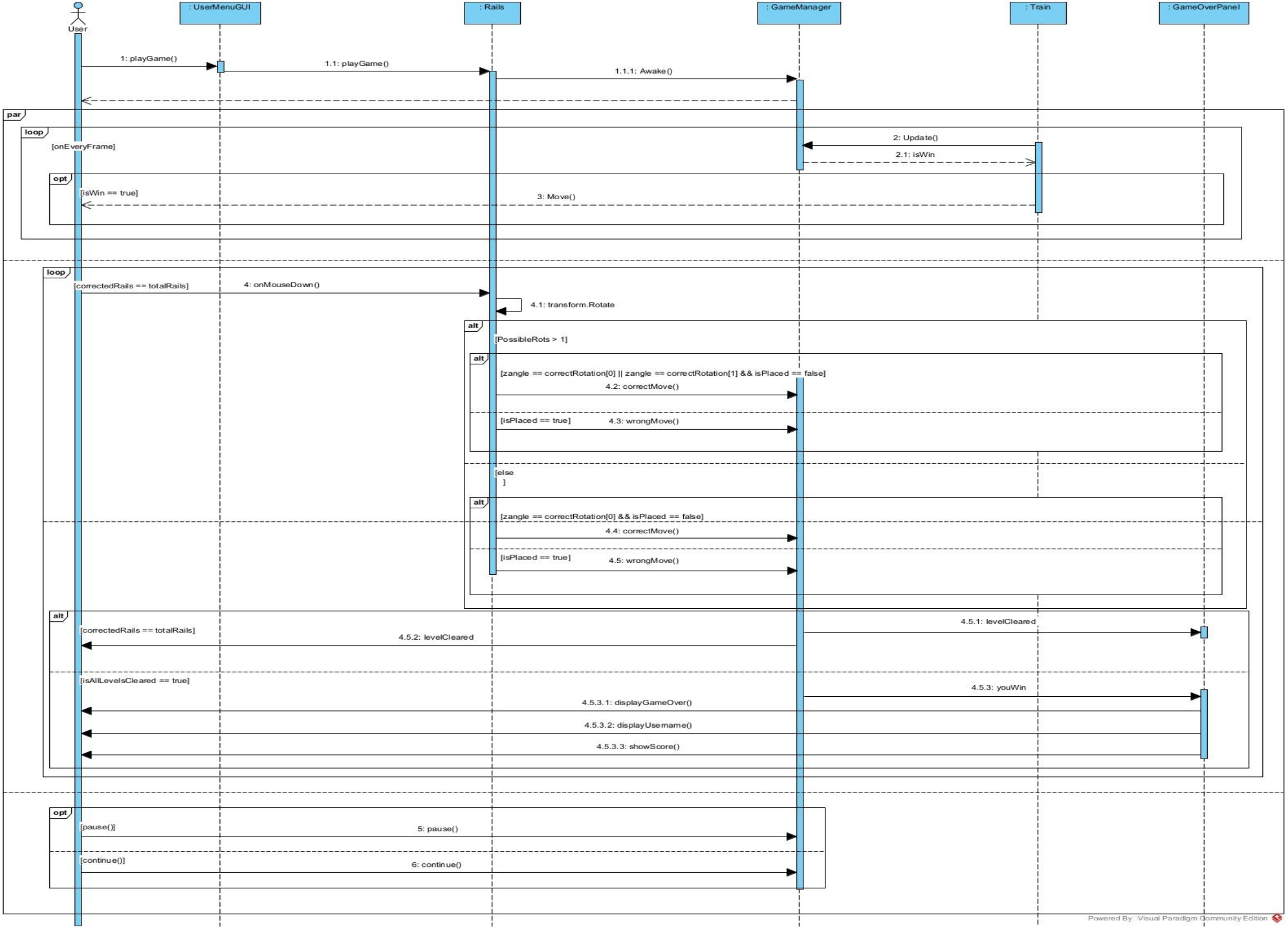

User
: UserMenuGUI
: LeaderboardPanel
1: Open_LeaderBoard()
1.1: Open_LeaderBoard()
1.1.1: displayHighscores()
opt
[returnUserMenuGUI()]
2: returnUserMenuGUI
Powered By Visual Paradigm Community Edition


User
: MainMenuGUI
: HowtoPlayPanel
1: LearnHowtoPlay()
1.1: LearnHowtoPlay()
1.1.1: HowtoPlay_Info
opt
[returnUserMenuGUI()]
2: returnUserMenuGUI()
Powered By Visual Paradigm Community Edition

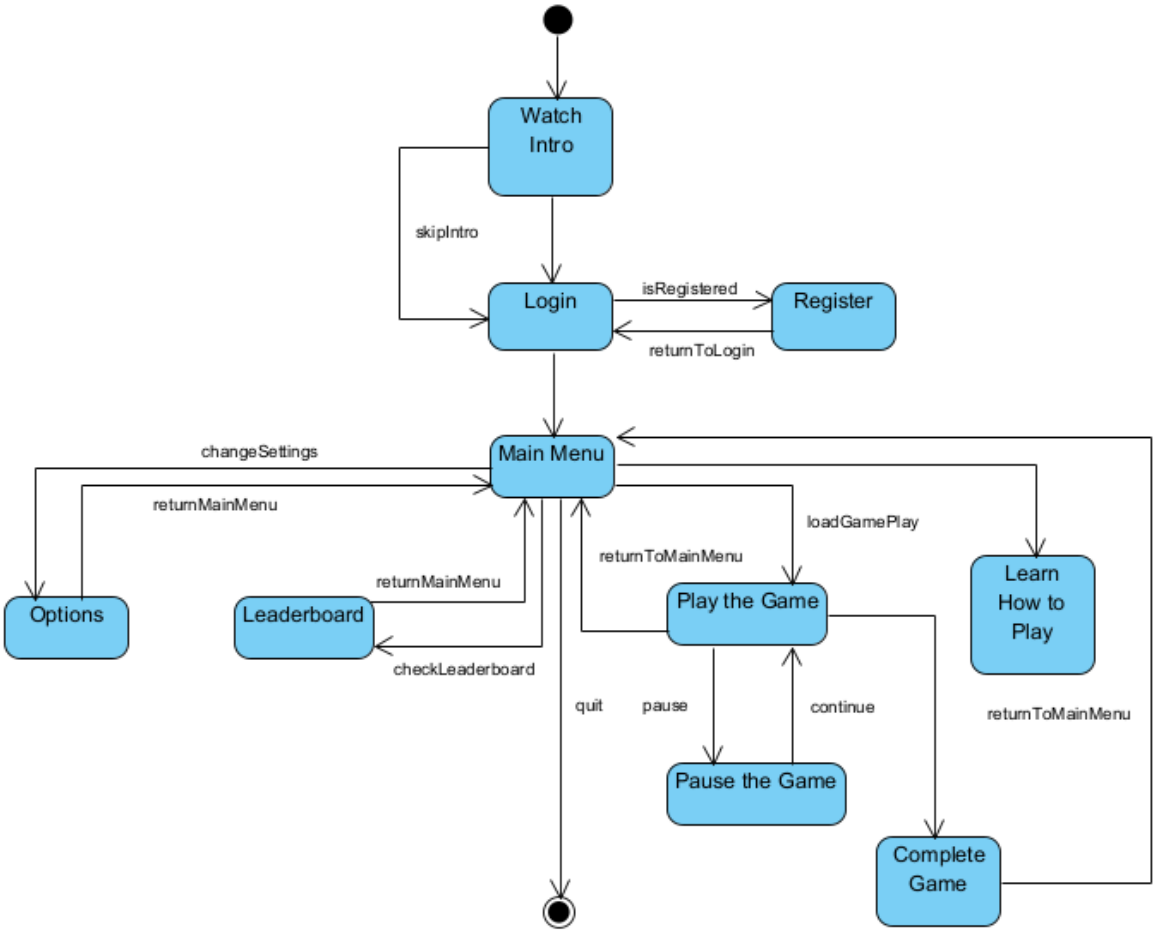

User
: UserMenuGUI
: SettingsPanel
: GameInfo
1: Open_Settings()
1.1: Open_Settings()
1.1.1: display()
opt
[changeBackground()]
2: changeBackground()
2.1: changeBackground()
[changeVolume()]
3: changeVolume()
3.1: changeVolume()
4: getHint()
3.2: getHint()
3.2.1: Hint
Powered By: Visual Paradigm Community Edition


User
: MainMenuGUI
1: exitGame()
Powered By: Visual Paradigm Community Edition

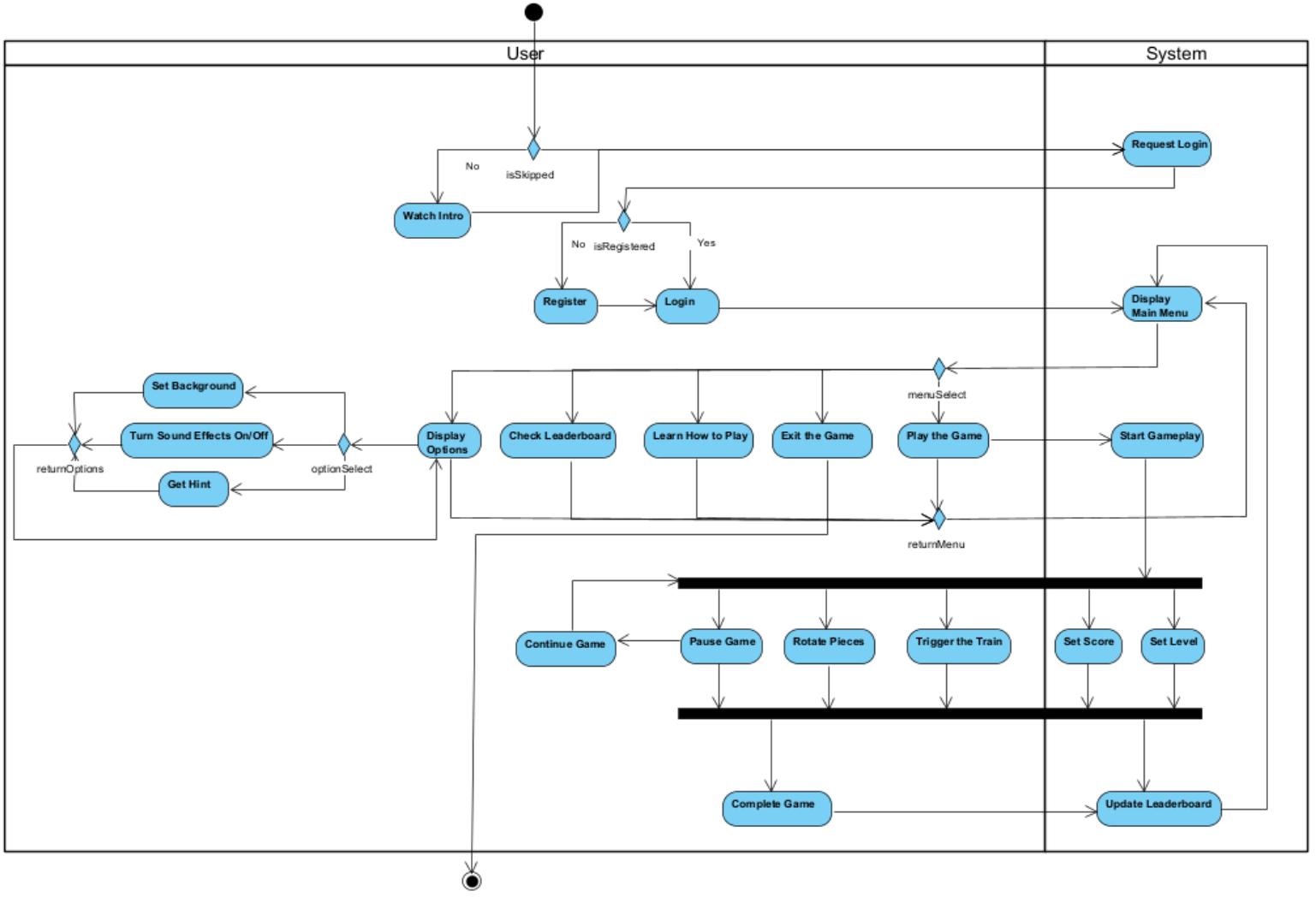

User
: UserMenuGUI
1: exitGame()
1.1: Exit pop-up
opt
[exitGame()]
2: exitGame()
Powered By: Visual Paradigm Community Edition


User
: UserMenuGUI
: Rails
: GameManager
: Train
: GameOverPanel
1: playGame()
1.1: playGame()
1.1.1: Awake()
par
loop
[onEveryFrame]
2: Update()
2.1: isWin
opt
[isWin == true]
3: Move()
loop
[correctedRails == totalRails]
4: onMouseDown()
4.1: transform.Rotate
alt
[PossibleRots > 1]
alt
[zangle == correctRotation[0] || zangle == correctRotation[1] && isPlaced == false]
4.2: correctMove()
[isPlaced == true]
4.3: wrongMove()
[else
]
alt
[zangle == correctRotation[0] && isPlaced == false]
4.4: correctMove()
[isPlaced == true]
4.5: wrongMove()
alt
[correctedRails == totalRails]
4.5.1: levelCleared
4.5.2: levelCleared
[isAllLevelsCleared == true]
4.5.3: youWin
4.5.3.1: displayGameOver()
4.5.3.2: displayUsername()
4.5.3.3: showScore()
opt
[pause()]
5: pause()
[continue()]
6: continue()
Powered By Visual Paradigm Community Edition

3.2. State Diagrams

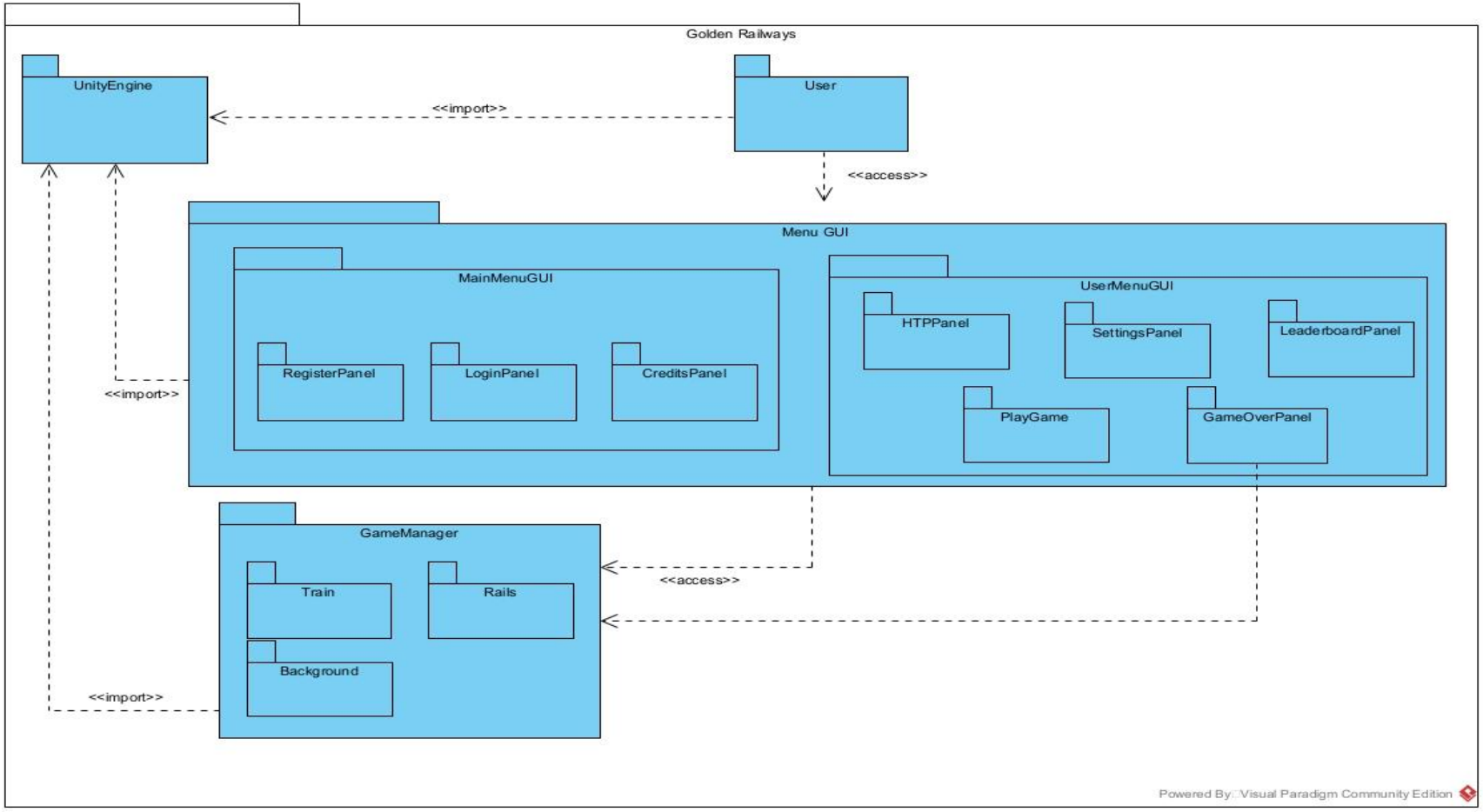


3.1. Activity Diagrams

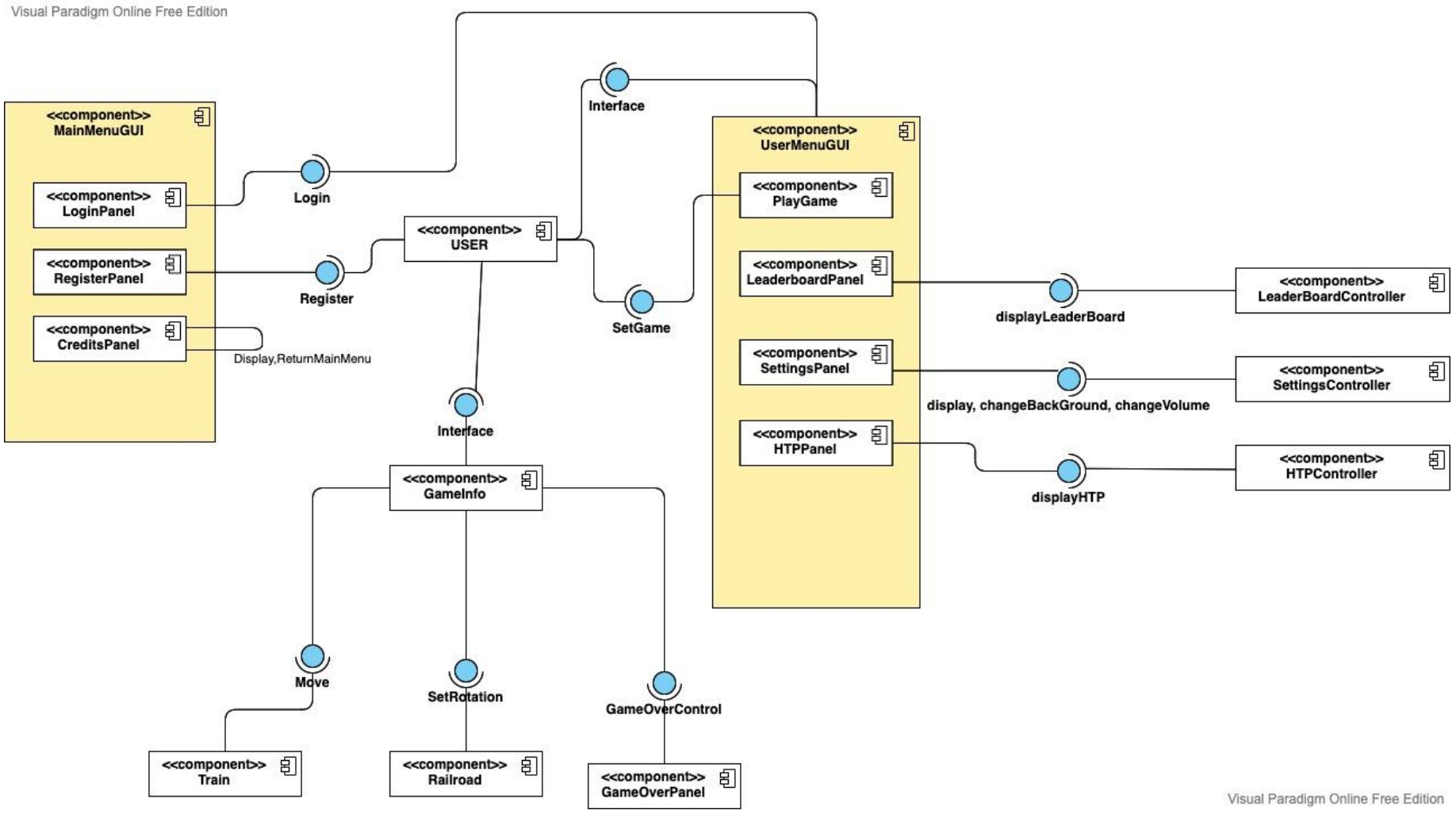


4. SOFTWARE ARCHITECTURE

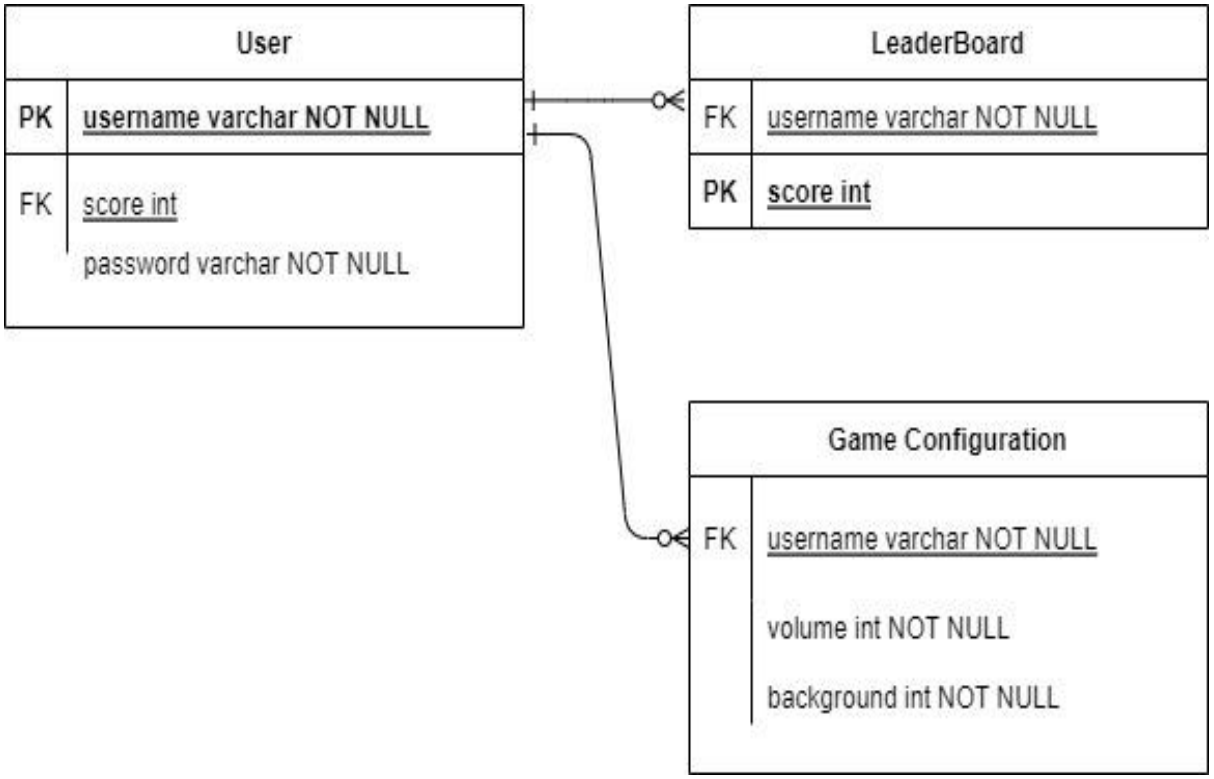
4.1. UML Package Diagram



4.2. UML Component Diagram



5. ENTITY RELATIONSHIP DIAGRAM



6. GLOSSARY & REFERENCES

6.1. Glossary

Activity Diagram: An activity diagram is basically an enhanced version of a flowchart that models the flow from one activity to another.

Entity Relationship Diagram: An entity relationship diagram is a type of structural diagram for use in database design. It contains different symbols and connectors.

Sequence Diagram: Interaction diagrams of classes (objects). It defines how our classes communicate with each other and in which order, in order to perform a specific operation.

State Diagram: State diagrams are UML diagrams used to model the dynamic nature of the system. They can also show how an entity responds to various events by changing from one state to another.

UML (Unified Modeling Language): UML is a standardized modeling language consisting of an integrated set of diagrams, developed to help system and software developers for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems.

UML Component Diagram: Component diagrams are essentially class diagrams that focus on a system's components that are often used to model the static implementation view of a system.

UML Package Diagram: Package diagram, a kind of structural diagram, shows the arrangement and organization of model elements in a middle to large scale project.

6.2. References

[1] Lethbridge, T. C., Laganière, R. (2005). Object-Oriented Software Engineering: Practical Software Development using UML and Java, McGraw-Hill Education.

[2] Golden Railways Requirement Analysis Report

[3] Visual Paradigm | Learning Guides