CosmoFighter

Design Document

Darshan Patnekar
Rutuja Kulkarni
Tejas Kumthekar
Sagar Shah

CONTENTS

1. Introduction

2. General Description

3. Architecture Details

1. Introduction

1.1 Purpose

‘CosmoFighter’, an Android game, is a space shuttle which will fight against the debris in space to prevent collision and protect the universe. The intention of the game is to have fun by attracting the users and by keeping them engaged in such a way that it will help in increasing their concentration and analytical skills.

1.2 Scope

- CosmoFighter is an Android game application which is designed for the user entertainment. It has a spaceship which will escape the debris and reach the destination by travelling in a projectile trajectory.
- The game will be available free for downloading from any Android play store. The game requires internet connection for downloading and sharing game activity but can be played offline.
- This is a single player game which can be played from any Android device. The game will contain multiple difficulty levels and packages such that every level of difficulty will be unlocked if you clear a level prior to it.
- The game will have social media integration that will enable users to share their game activity with the community.

1.3 Definitions and Abbreviations

- **CosmoFighter** - This is the name of the game.
- **Debris** - This are the obstacles that will be generated while playing each level. They are not user controlled.
- **Spaceship** - This will be a user controlled object that is used to navigate through the game.
- **Flag** - This is the destination where the spaceship must reach.

1.4 Overview

The project will follow MVC architecture. The parts of the game are front end, game logic and database. Database used will be SQLite which comes under ‘Model’ section of the architecture. The Controllers are ‘Game Logic’ and ‘User Controller’ and ‘Game Renderer’. View includes UI for the actual game play and other Screens like Share, Menu and User Profile.

2. General Description

2.1 Tools and Technology

Software Requirements: Android SDK, IDE like Eclipse or Android Studio, Android Debugger

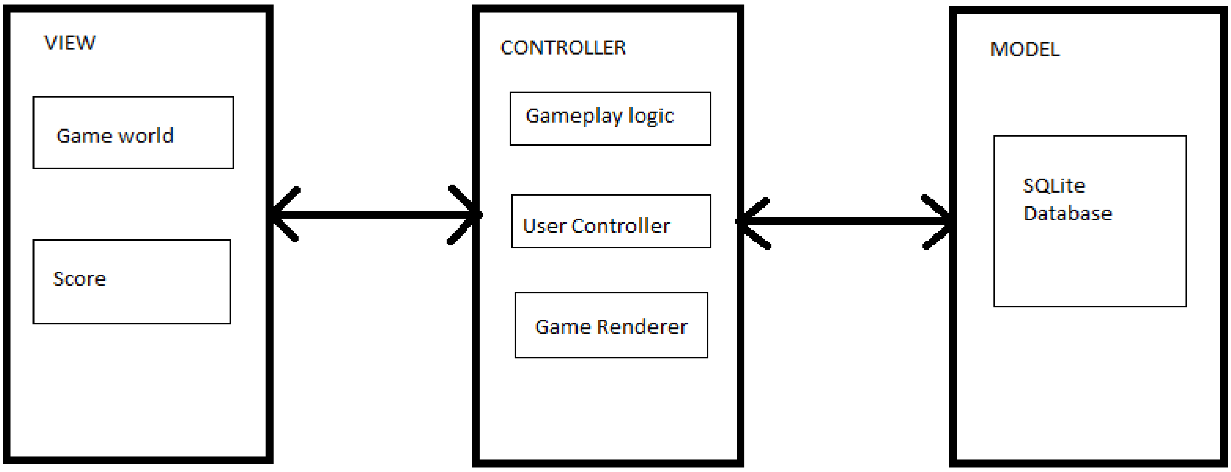
Programming Languages: Java

Other technologies: XML

2.2 Working Environment for the Product

The game could be played on all Android devices having underlying operating system Android v2.3 or above.

2.3 High Level Block Diagram



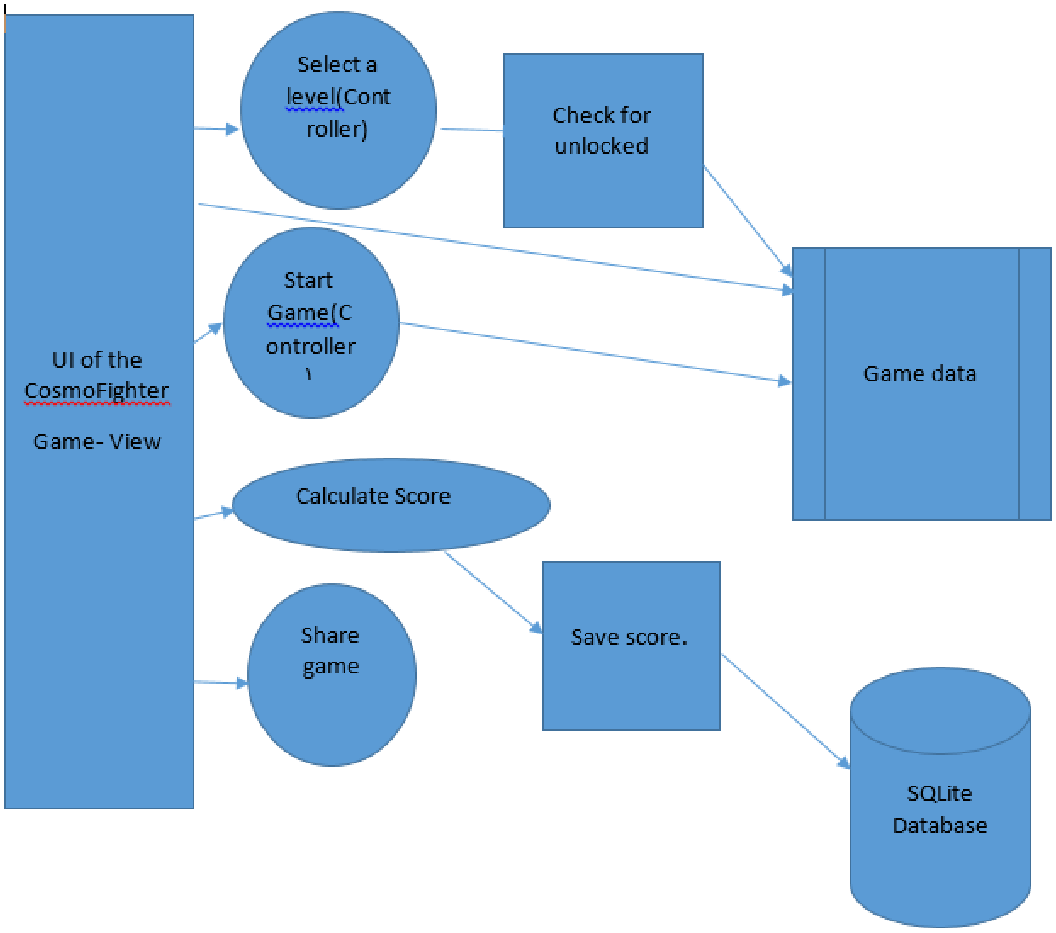
2.4 Assumptions

Pre requisites for project implementation are:

- The developer is assumed to have knowledge for setting up the working environment for the game.
- It is assumed that the developer has an Android device with version 2.3 or above.

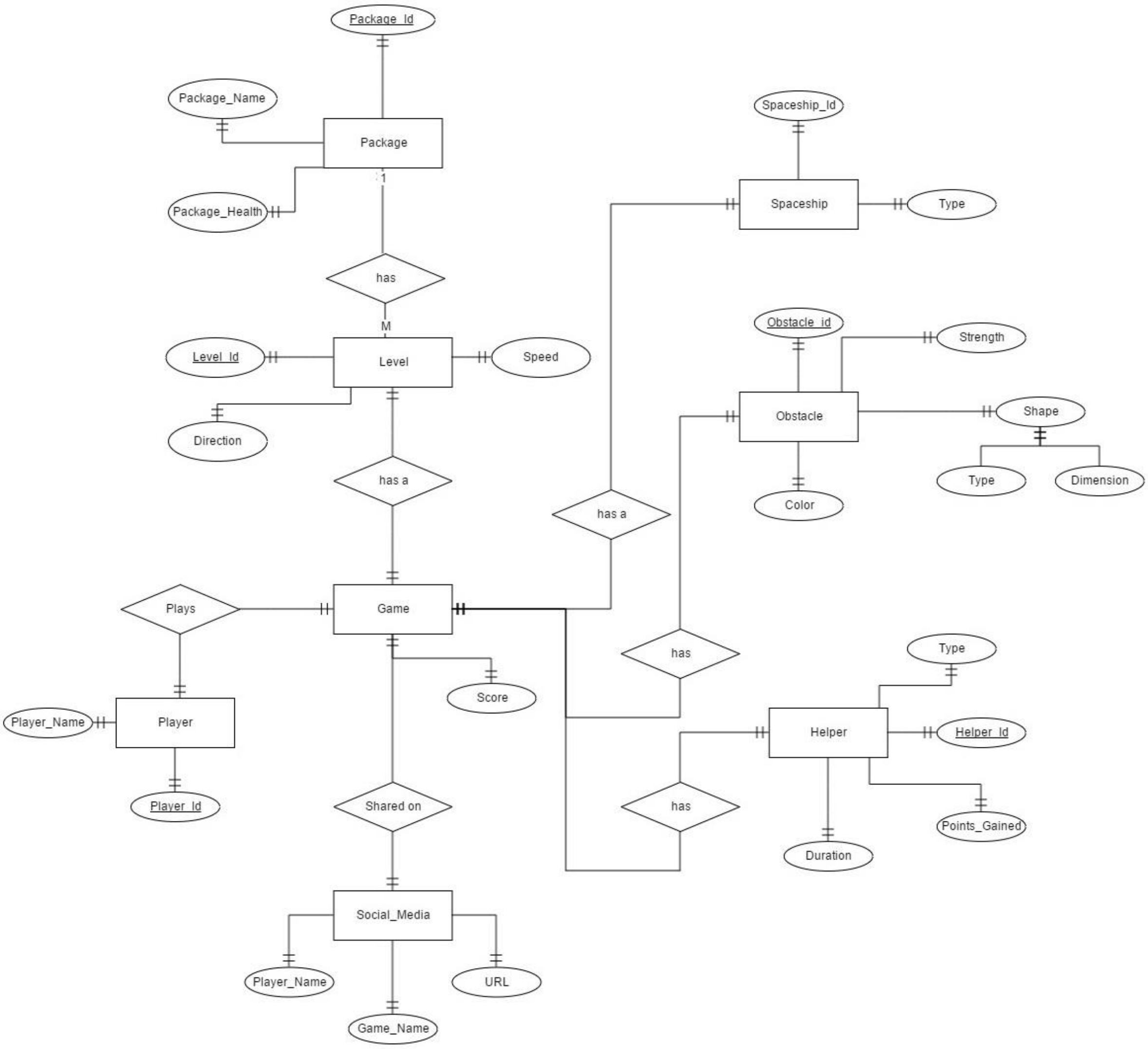
3. Architecture Details

3.1 Top Level Architecture



3.2 Database Architecture

ER Diagram



3.3 Use Cases

1. Load Game

Load Game	
Description	On clicking the game icon, the game will be loaded.
Pre-Condition	Game is installed on the device.
Workflow	Game is initialized.
Post Condition	Menu screen will be displayed.

2. Play Game

Play Game	
Description	On clicking the play option in the menu, the game will be started.
Pre-Condition	Menu has been loaded.
Workflow	The UI is loaded for the selected level.
Post Condition	Score is calculated.

3. Change settings

Change settings	
Description	On clicking settings icon, volume and music controls can be changed.
Pre-Condition	Menu screen is displayed.
Workflow	Volume and music are changed.
Post Condition	Volume and music settings are saved.

4. Share Game Activity

Share Game Activity	
Description	The game activity like score can be shared on Facebook.
Pre-Condition	User is connected to Facebook.
Workflow	User can share his high score and game profile on Facebook.
Post Condition	Activity is shared as a post on Facebook.

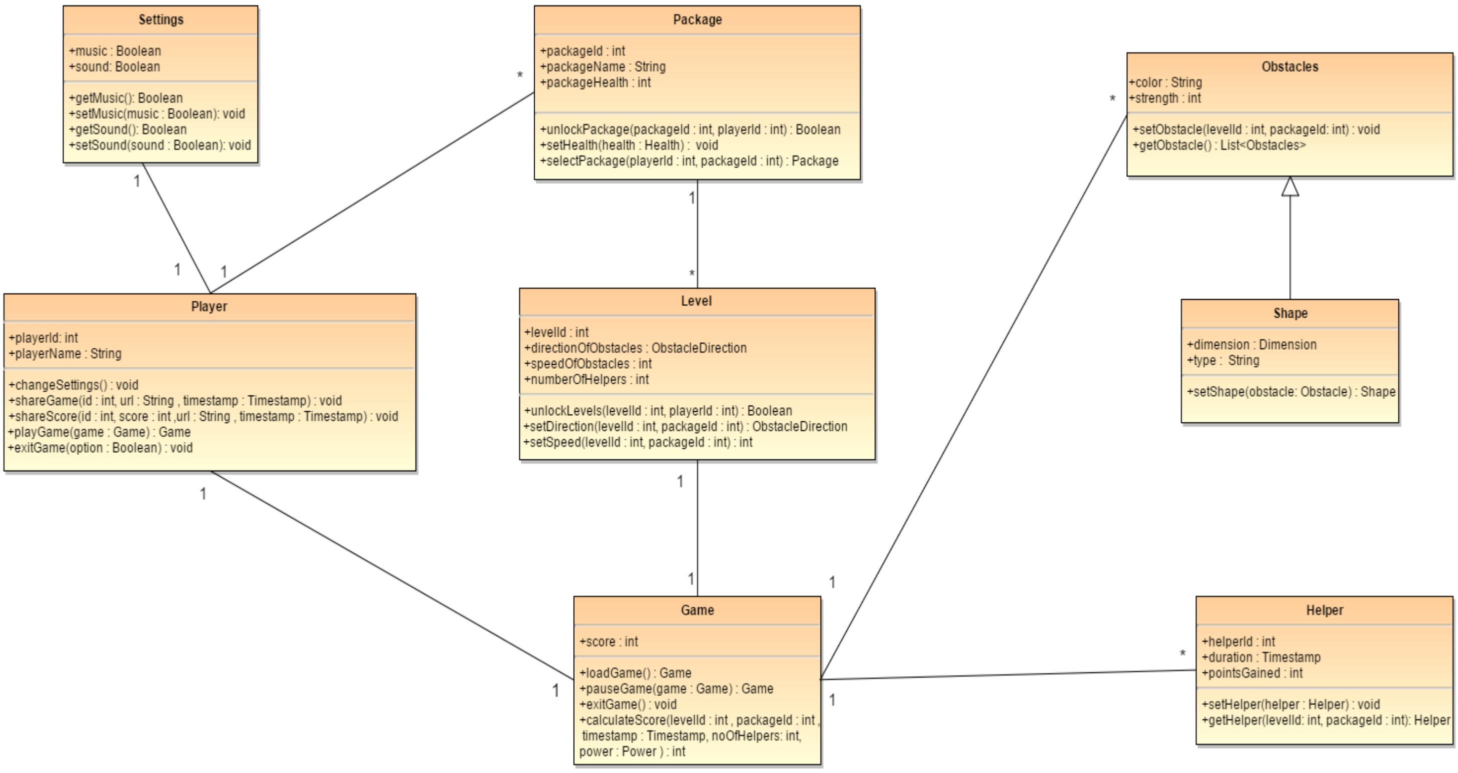
5. View High Score

View High Score	
Description	User can view his score for the particular level played in the game.
Pre-Condition	User has played that level at least once.
Workflow	User selects the level for which he wants the score to be displayed.
Post Condition	Score is displayed for that particular level.

6. Exit Game

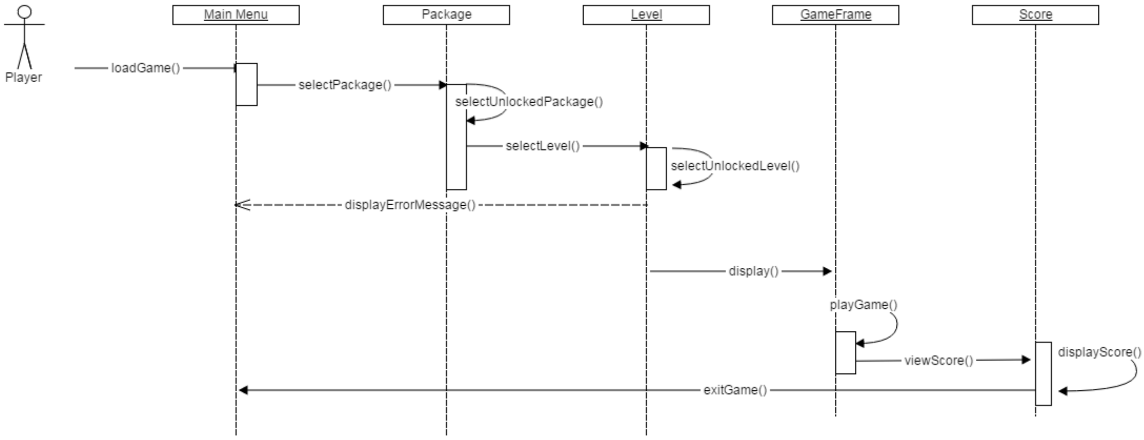
Exit Game	
Description	User can exit the game if he wishes to.
Pre-Condition	Game has been loaded and exit option has been displayed on the screen.
Workflow	On clicking exit button in the menu, game can be exited.
Post Condition	User exits the game.

3.4 Class Diagram for CosmoFighter

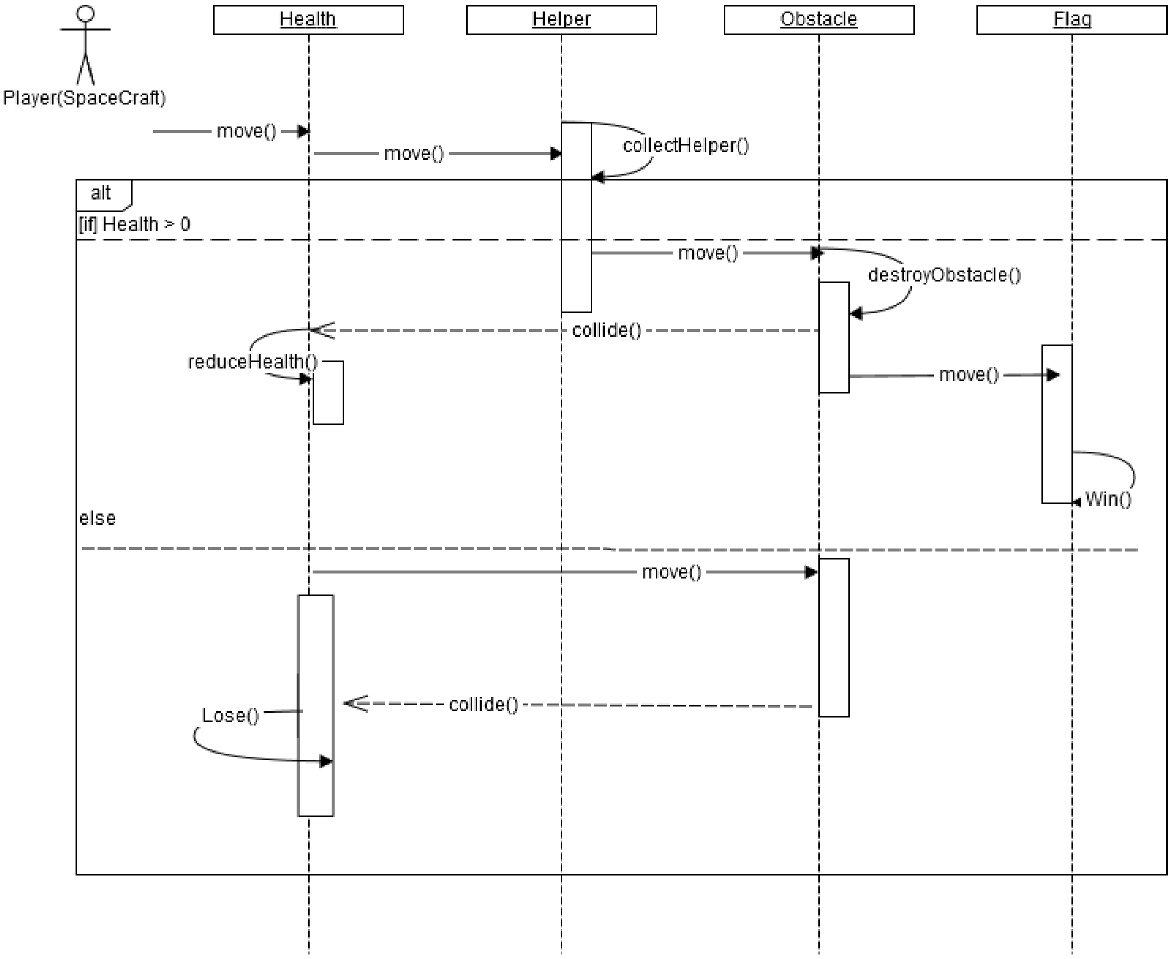

Settings
+music : Boolean
+sound: Boolean
+getMusic(): Boolean
+setMusic(music : Boolean): void
+getSound(): Boolean
+setSound(sound : Boolean): void
Package
+packageId : int
+packageName : String
+packageHealth : int
+unlockPackage(packageId : int, playerId : int) : Boolean
+setHealth(health : Health) : void
+selectPackage(playerId : int, packageId : int) : Package
Obstacles
+color : String
+strength : int
+setObstacle(levelId : int, packageId: int) : void
+getObstacle() : List<Obstacles>
Player
+playerId: int
+playerName : String
+changeSettings() : void
+shareGame(id : int, url : String , timestamp : Timestamp) : void
+shareScore(id : int, score : int ,url : String , timestamp : Timestamp) : void
+playGame(game : Game) : Game
+exitGame(option : Boolean) : void
Level
+levelId : int
+directionOfObstacles : ObstacleDirection
+speedOfObstacles : int
+numberOfHelpers : int
+unlockLevels(levelId : int, playerId : int) : Boolean
+setDirection(levelId : int, packageId : int) : ObstacleDirection
+setSpeed(levelId : int, packageId : int) : int
Shape
+dimension : Dimension
+type : String
+setShape(obstacle: Obstacle) : Shape
Game
+score : int
+loadGame() : Game
+pauseGame(game : Game) : Game
+exitGame() : void
+calculateScore(levelId : int , packageId : int , timestamp : Timestamp, noOfHelpers: int, power : Power) : int
Helper
+helperId : int
+duration : Timestamp
+pointsGained : int
+setHelper(helper : Helper) : void
+getHelper(levelId: int, packageId : int): Helper
1
1
1
*
1
*
*
1
1
1
1
1
1
*

3.5 Sequence Diagrams

Load Game -

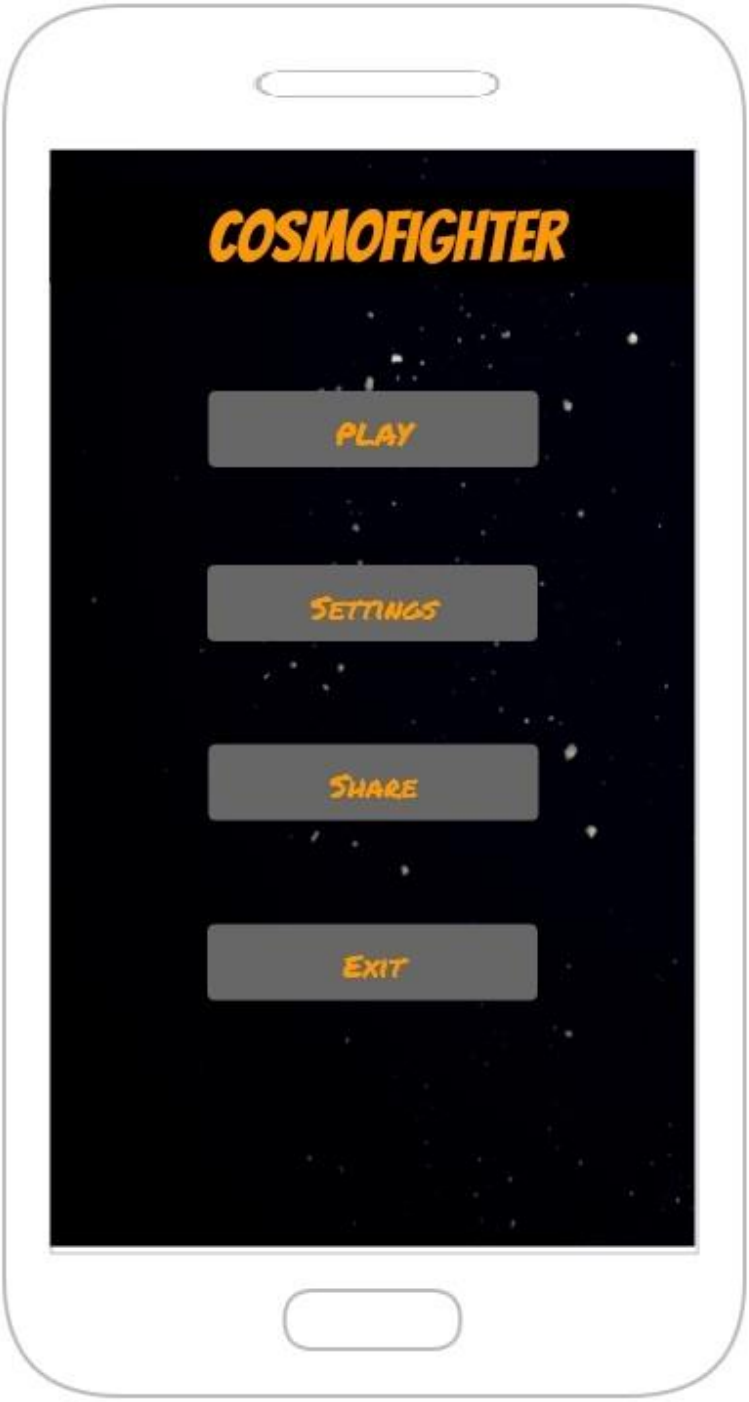


Game Play –

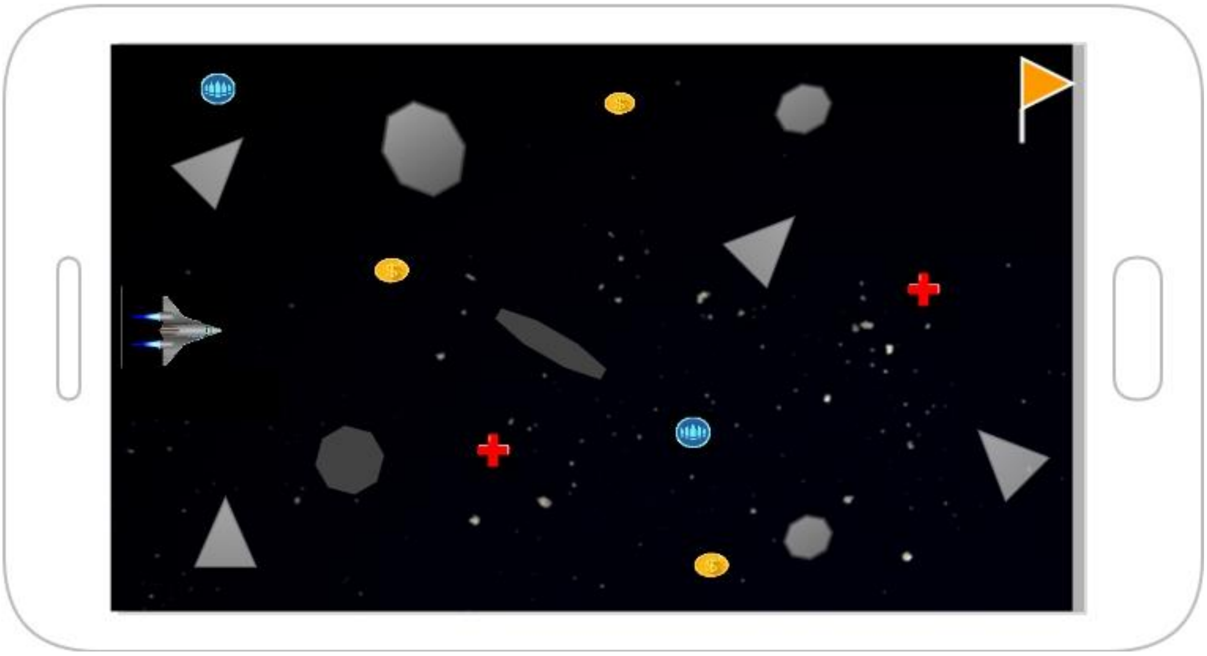


3.6 User Interface

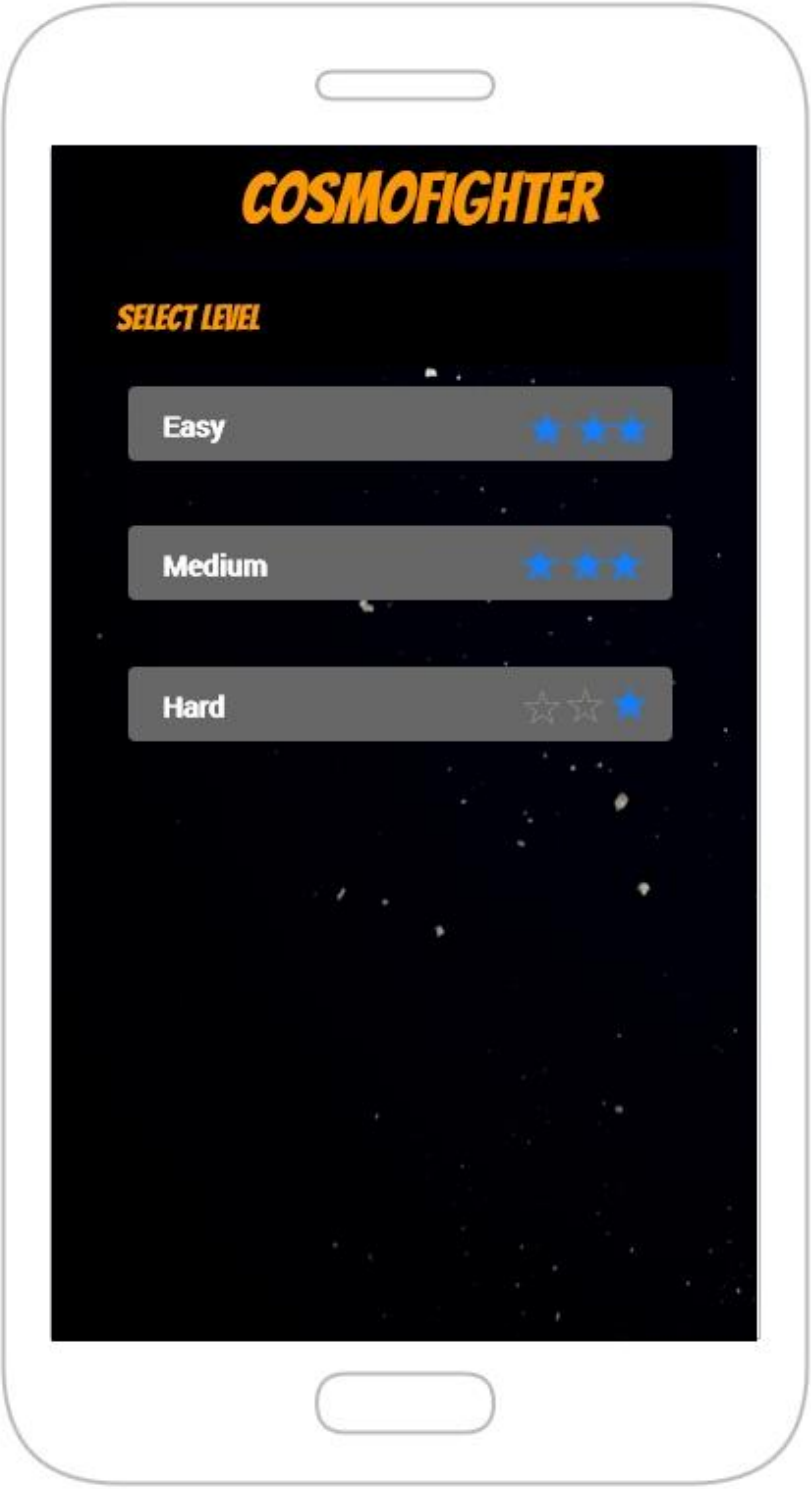
Game Menu



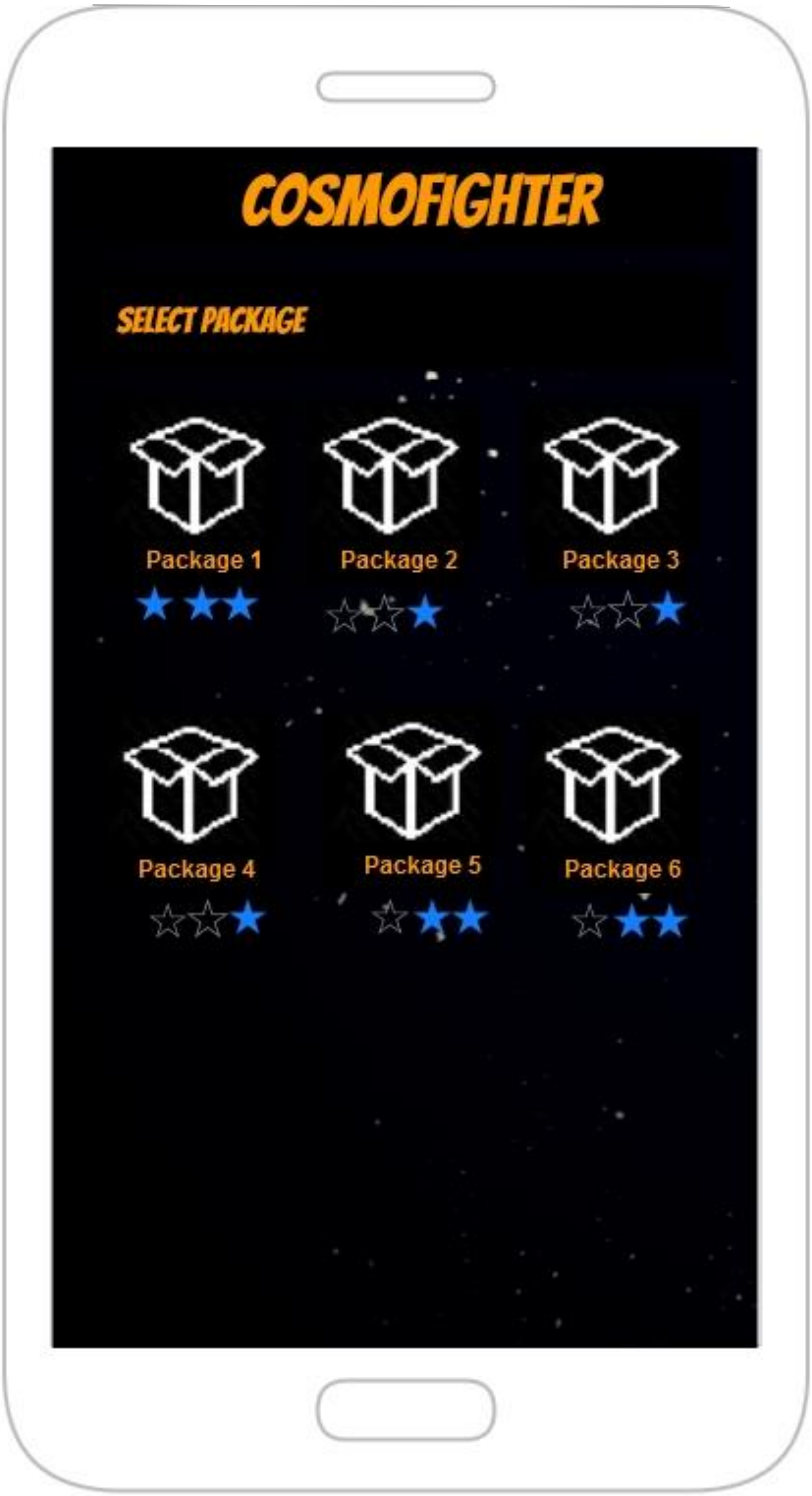
Play Game



Select Level



Select Package



3.7 Error Handling

Errors will be handled using Auditing and Logging techniques. Exceptions and failures will be handled by logging them in the Logging framework. Also, proper messages would be logged into Audit Logs so that it would be easy to debug the application.