System and Software Architecture Description (SSAD)

Spy – The Android Mobile Game

John Sun (Product Manager),

Kaya Ota (UX Designer),

Frank Li (Tech Lead),

Eric Vuu (Tester)

3/1/2017

Version History

Date	Author	Version	Changes made	Rationale
03/1/2017	JS	1	• Original document	• Initial draft for semester project

Table of Contents

Table of Tables

Table of Figures

1. Introduction

Spy – a mobile game designed for Android – is a game of intrigue, and deception, where players must figure out who is trust worthy, and who isn't. At the start of every game, players are randomly assigned location and role. Everyone who isn't a spy, has a day job at this random location. The spy does not know what the location is, and it is her job to figure out the location, based on the questions the other players are asking each other. Players ask each other questions to determine who is the spy. If someone arouses enough suspicion – to be voted as a spy by most players – then that player must guess the location. If she is wrong, then she loses the game and the other players win. Therefore, it is in the spy's best interest to stall the game. (As this is another way the spy can win, stalling the game until the timer runs out, without being caught.)

1.1 Purpose of the SSAD

The objectives of this document are to: Help our team formulate a concrete implementation plan for our game, and to describe how the game is designed; so that it can be reproduced and built upon by other developers.

1.2 Status of the SSAD

This is the first version of this document. There are no changes from a previous version.

2. System Analysis

2.1 System Analysis Overview

The purpose of this mobile game is to bring Spy Fall – a beloved card and web application based game – to Android. This way, players do not have to use their browsers to play the game, and can access it right on their cell phones. Furthermore, by adding massively multiplayer online role playing game (MMORPG) elements, players have a goal outside of the individual games to strive towards. Winning games awards Spy Points, which allows players to purchase rewards from an in-game shop. By increasing player incentives to play the game, and making a mobile optimized app, our mobile game will be able to better represent the experience of playing the physical card game (Spy Fall), compared to any web application.

2.1.1 System Context

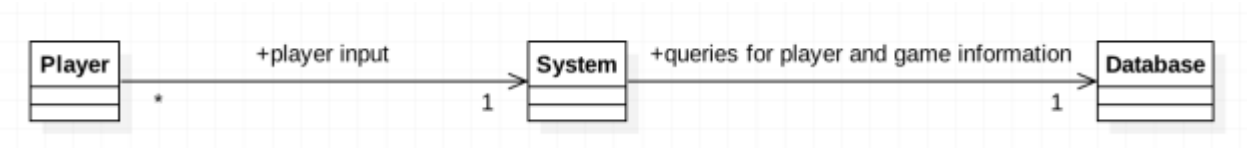

Figure 1: System Context Diagram

Player
The player is the actor that interacts directly with the system, and causes the system to query the database. Player responsibilities are very limited, as most of the work is done by the system/db. When the user logs into the game, that will cause a query against the db. Later, when the player creates, joins or plays a game of *Spy*, those will cause more queries against the db.

System
The system is comprised of the Android application, along with all the activities, resource files, libraries, and so on that make it up. Our system will help hide all the complexity of the game's backend and the database so that the user experience is accessible. It will achieve this by automating a lot of the work of creating, joining, and starting a game.

Database
Our application's backbone is the MySQL database, which hosts all our player and game data. When the system queries the database for a player's profile – earned Spy Points, player status, player name, player avatar and so on – the database will return all that information.

Table 1: Actors Summary

Actor	Description	Responsibilities
Player	The person playing the game	Play the game
System	The Android application	Hide complexity from user and query database for appropriate information
Database	MySQL database	System with asynchronous access to player and game data

2.1.2 Artifacts & Information

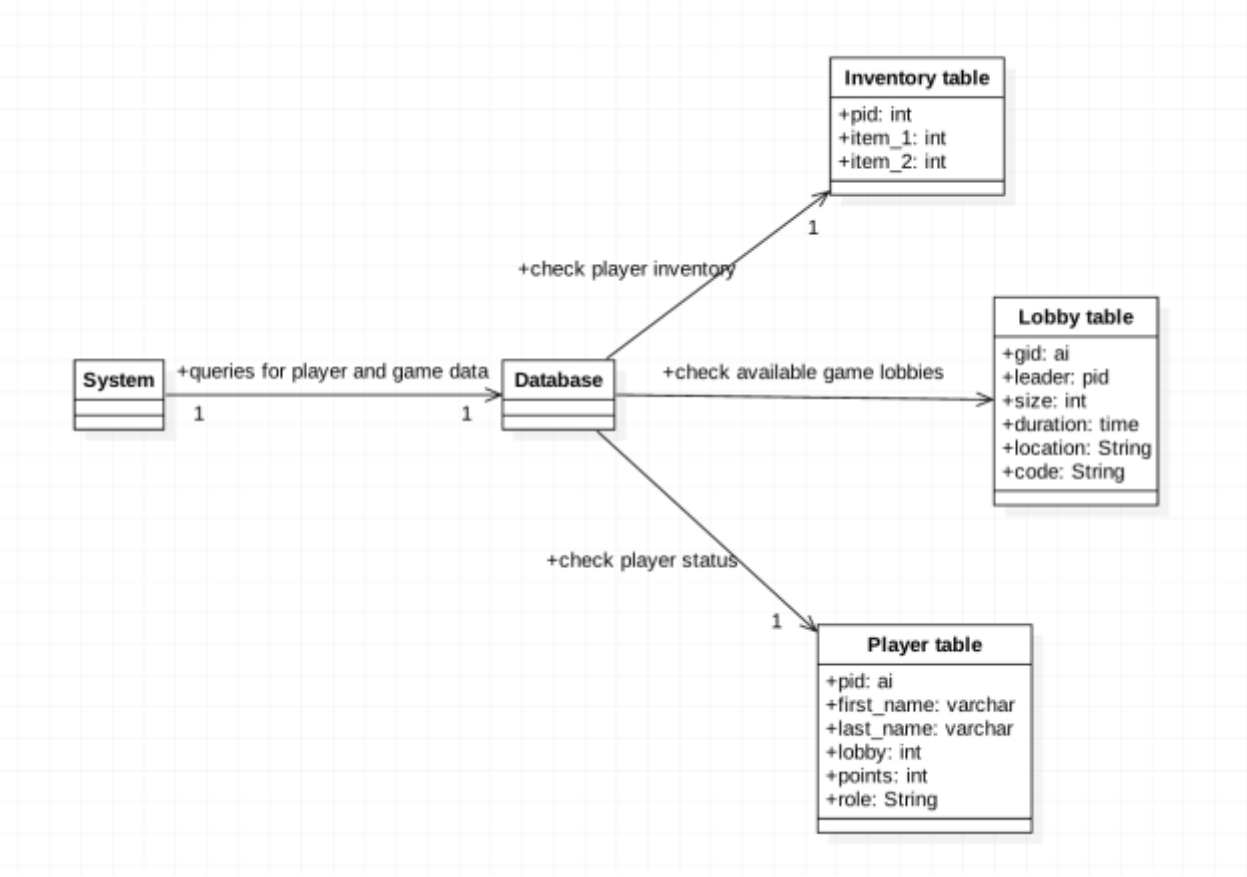

Figure 2: Artifacts and Information Diagram

Table 2: Artifacts and Information Summary

Artifact	Purpose
Player table	The player table keeps track of basic user and game information. When a player joins a game, that will be reflected in this game (under the status attribute). Pid will link the player and inventory tables, so that we can do natural join and know which user has which items.
Inventory table	This table keeps track of which items players have obtained. The downside to this design, is that the number of slots in player inventory is limited by the number of columns.

2.1.3 Behavior

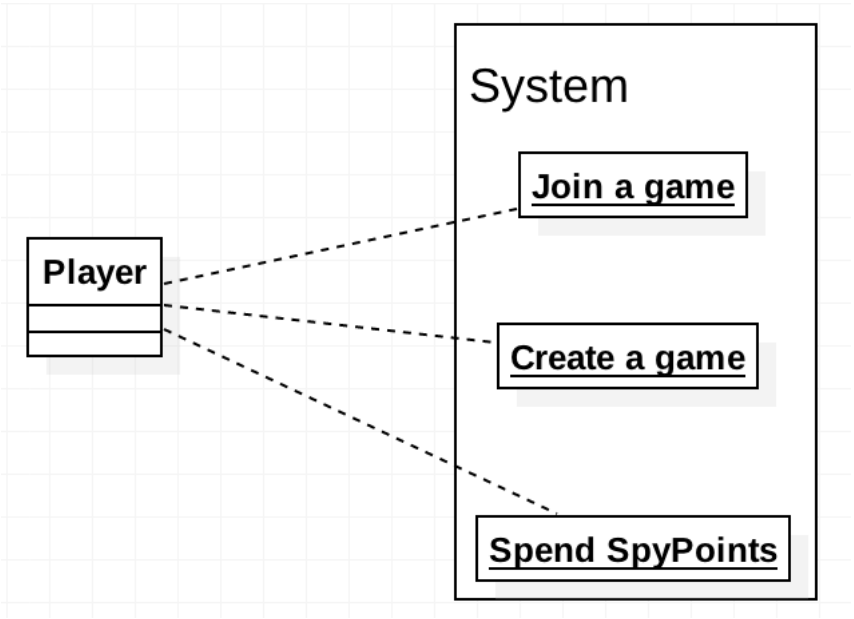

Figure 3: Player-System Interactions (After User Logs In)

2.1.3.1 Player-System Interactions (After User Logs In)

2.1.3.1.1 Create/Join a Game

Table 3: Process Description

Identifier	Create a Game/Join a Game
Purpose	Allows player to create or join a customizable game lobby.
Requirements	Activity front end (within the app), and a database back end.
Development Risks	Using one database for backend is not a scalable solution.
Pre-conditions	The user must already have logged into their account.
Post-conditions	The database must be up and running.

Table 4: Typical Course of Action – User Joins a Game

Seq#	Actor's Action	System's Response
1	Player selects Join a Game.	System displays activity with Text (Enter a game code), EditText and Button (Enter) views.
2	Player enters in the code associated with the game lobby they wish to join, then clicks Enter.	System searches the Lobby table for games that have a matching code. If found, the user is automatically ushered into the desired game lobby.

Table 5: Alternate Course of Action – User Enters Incorrect Code

Seq#	Actor's Action	System's Response
1	Player selects Join a Game	System displays activity with Text (Enter a game code), EditText and Button (Enter) views.
2	Player enters in the code associated with the game lobby they wish to join, then clicks Enter.	After searching the Lobby table for the appropriate game, the system determines that the user entered an incorrect game code. An error message is displayed to the user, informing them that they have entered an incorrect game code; and they should try again.

Table 6: Exceptional Course of Action – Database is Down

Seq#	Actor's Action	System's Response
1	Player selects Join a Game	System displays activity with Text (Enter a game code), EditText and Button (Enter) views.

2	Player enters in the code associated with the game lobby they wish to join, then clicks Enter.	The database is down, and the application shows a user-friendly error message that the server is down.

3. Technology-Specific System Design

3.1 Design Overview

3.1.1 System Structure

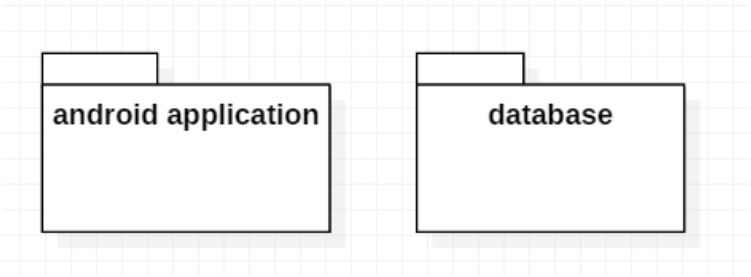

Figure 4: Conceptual Domain Model

Figure 5: Hardware Component Class Diagram

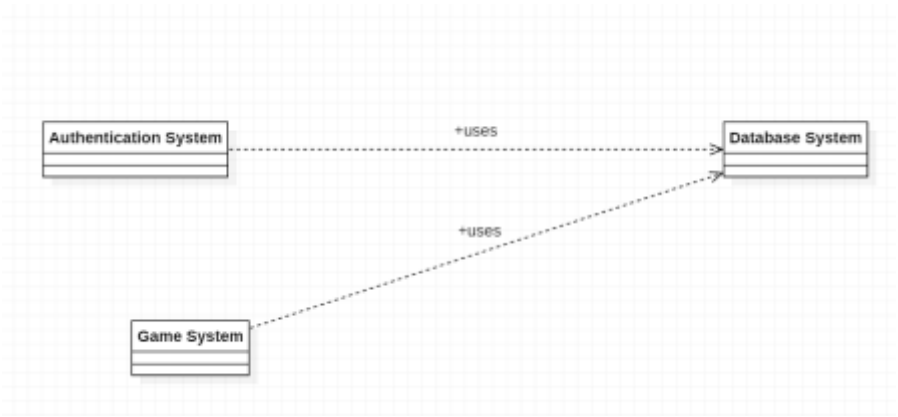

Figure 6: Software Component Class Diagram

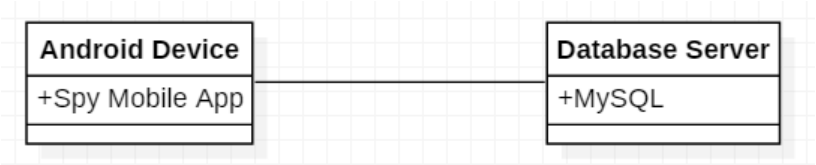

Figure 6: Deployment Diagram

Table 7: Hardware Component Description

Hardware Component	Description
Android Device	Any Android phone that runs Android Nougat
Database Server	Any machine with high uptime that runs MySQL

Table 8: Software Component Description

Software Component	Description
Spy Mobile Application	The game that we are developing
MySQL	Oracle's relational database software

3.1.2 Design Classes

3.1.2.1 Overview of Classes

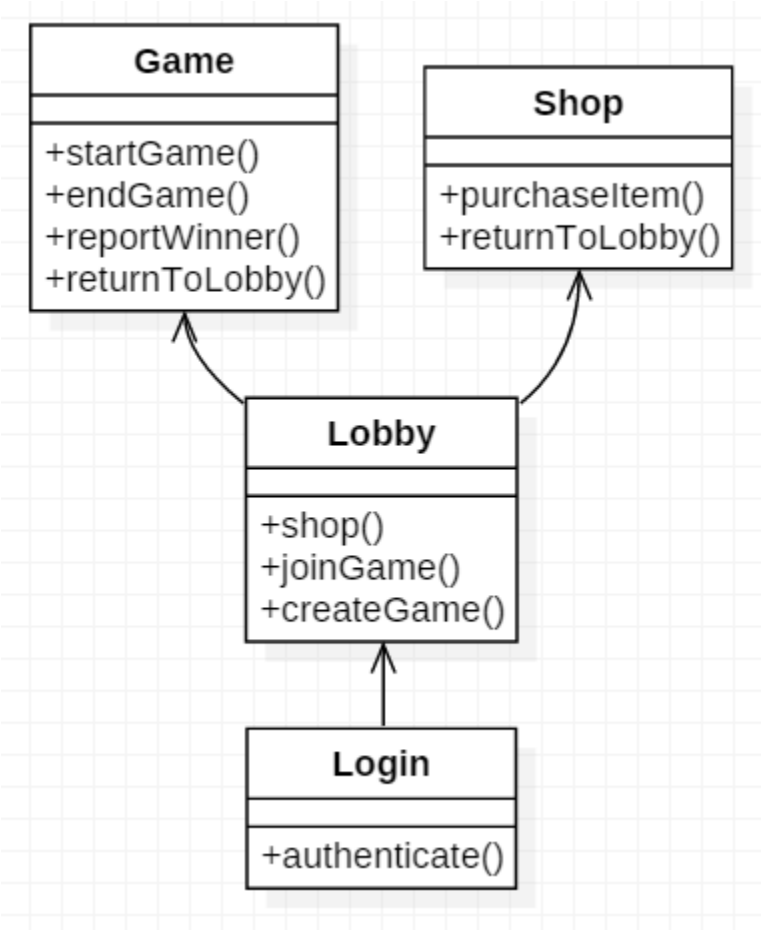

Figure 7: Design Class Diagram

Table 9: Design Class Description

Class	Type	Description
Login	Class	Authenticates user and allows her to associate game play data with her profile.
Lobby	Class	Displays a menu that asks the user what she would like to do: Shop for items, join or create a game of Spy.
Game	Class	Handles all the game functions such as: picking a random location, picking a random player to be a spy, adding points to player profiles, and so on.
Shop	Class	Displays a list of items that the user may purchase with her SpyPoints.

3.1.3 Process Realization

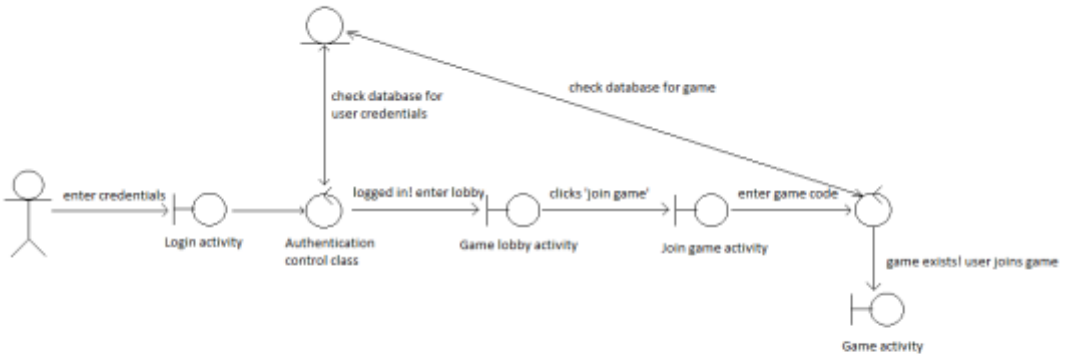

Figure 8: Robustness Diagram

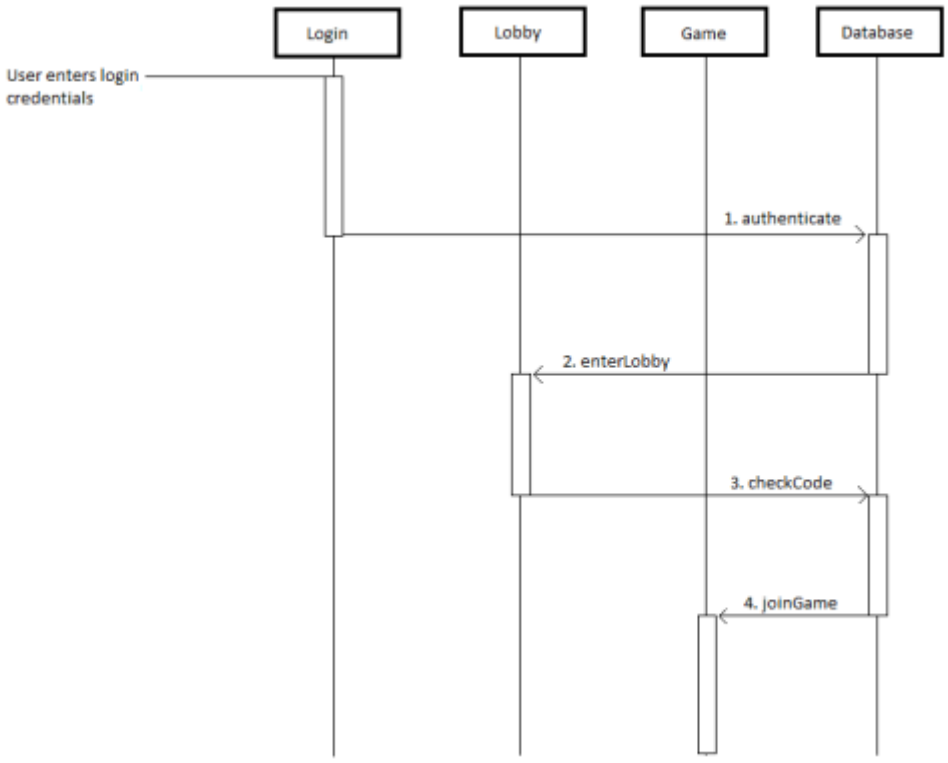

Figure 9: Sequence Diagram

3.2 Design Rationale

We chose the design of an Android device communicating with a MySQL database server because, our game requires a database outside of SQLite. This is because our game is multiplayer, and all players must access information from the same server.

4. Architectural Styles, Patterns and Frameworks

Table 10: Architectural Styles, Patterns, and Frameworks

Name	Description	Benefits, Costs, and Limitations
Java	Oracle's very own language	Everyone on the team knows Java, and it is the language for developing Android applications. Limitations include: We may not be able to use JDBC drivers to broker data between our app and the DB.