D&J

Version <1.4>

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 21/11/2017 | 1.1 | Introduction + Architecture Goals | Phy Liêng |
| 21/11/2017 | 1.2 | Use Case Model + Overview | Phy Liêng + Thoại Thông |
| 3/12/2017 | 1.3 | Implementation View | Phy Liêng |
| 4/12/2017 | 1.4 | Deployment | Nguyên Hồ |
| 5/12/2017 | 1.5 | Class diagram with package division | Bach Le |

Table of Contents

1. Introduction 4

2. Architectural Goals and Constraints 5

3. Use-Case Model 5

4. Logical View 6

4.1 Component: abc 7

5. Deployment 13

6. Implementation View 13

Software Architecture Document

# Introduction

**Purpose**

This document gives an extensive architectural overview of the system which uses diverse engineering perspectives to show different parts of the system. It is used to convey important architectural decisions on the system.

**Scope**

This Software Architecture Document applies to the D&J which will be developed by Group04.

**Definitions, acronyms, abbreviations**

* **Social Network Interactions**
  + User interaction with the game via social network includes user authentication, sharing, inviting friends.
* **Step Counting**
  + A Pedometer API provides step count and distance covered which will be used in a certain mission or an achievement.
* **Mission**
  + Information about a mission related to walking is given to a user which is then implemented. This may include visiting a position, walking a certain distance, or inviting friends to join in game.
* **Achievements**
  + Information about several special milestones that a user may reach. For example, categories of missions, how many miles user has already achieved, …
* **Pet**
  + A 3D dog can perform some basic animations of a normal dog. It is raised by bone and milk bought from shop.
* **Affection Points**
  + Points are not only used to raise user’s level and but also is an efficient way to prevent users from not quitting the game by decreasing affection points.
* **Shop**
  + A place where users can buy food or skins for their dogs.

**References**

1. Project Plan

2. Vision Document

3. Use case model and use case specification.

**Overview**

This document presents the architectural as a series of views; use case view, process view, deployment view, and implementation view. These views are used from the Unified Modeling Language (UML).

# Architectural Goals and Constraints

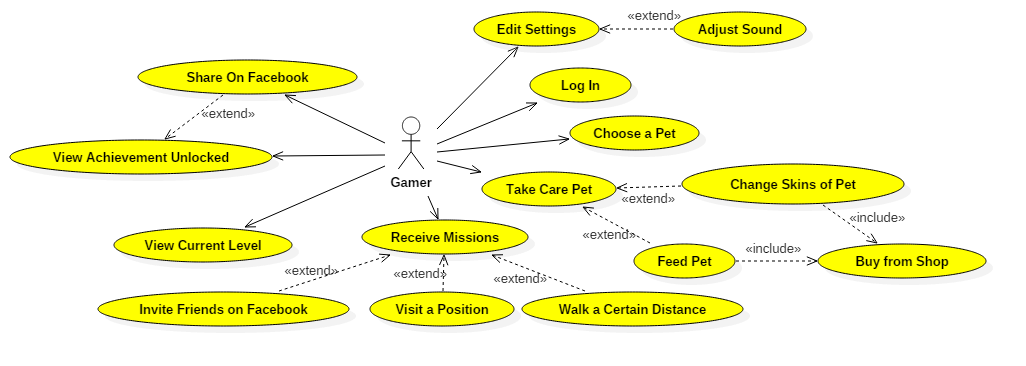
There are some software requirements and objectives that have significant impact on the architecture:

* All the user database must be kept in a secured manner to provide a fair and fun game environment.
* The existing D&J is developed on Unity, so it must be capable of importing to multiple platform later on.
* This game project deploys in step counting a so it has an ability to use again in other health projects.
* All performance and loading requirements must be taken in to consideration as the architecture is being developed since the game uses 3D model design and animation.

Constraints:

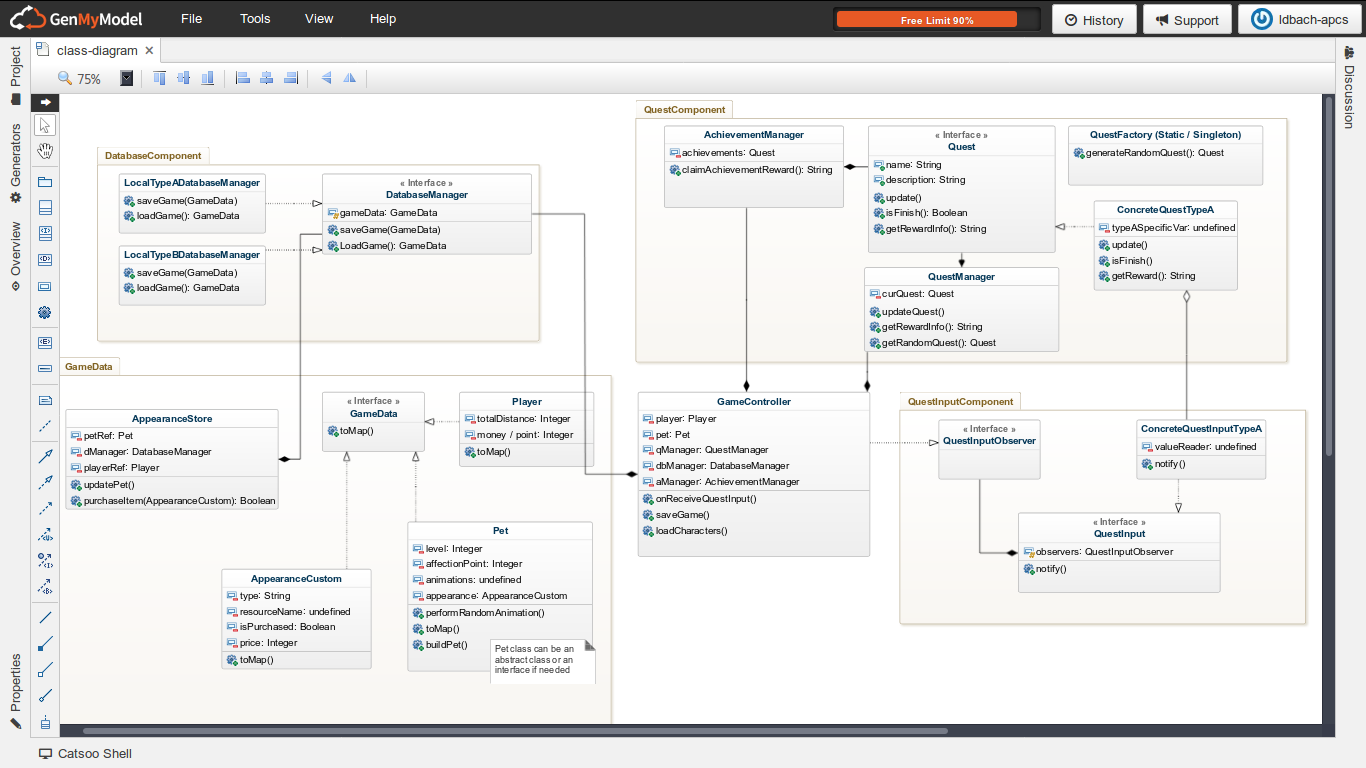
* Developers are new to Unity.

# Use-Case Model

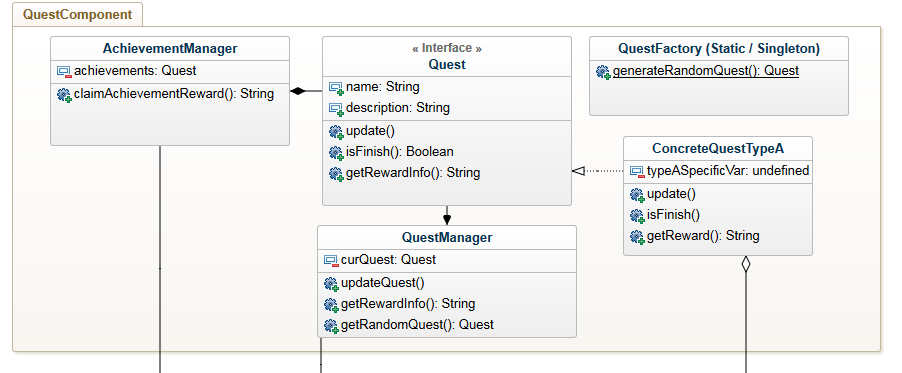


# Logical View

Overview: Class Diagram show both components and components’ contents.



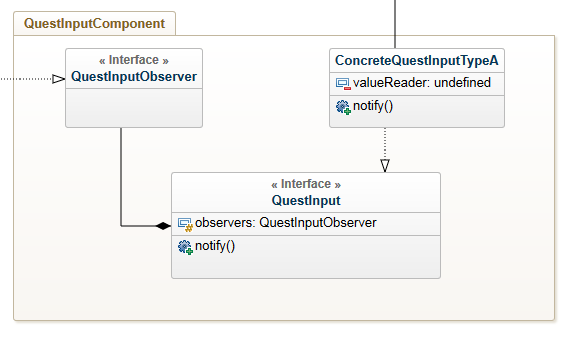
## Component: Quest



Quest component introduce a Quest interface which helps in later extension. With the help of QuestManager and QuestFactory, Main class does not need to know what kind of quest is in progress.

Achievement is a special kind of Quest, therefore it is classified into the same component.

## Component: QuestInput

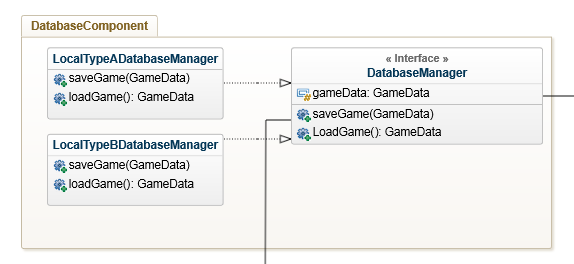


The QuestInput component consists of any kind of user interaction that helps in completion of quests and achievements.

The QuestInput component provides a QuestInput Interface, to which open to extension.

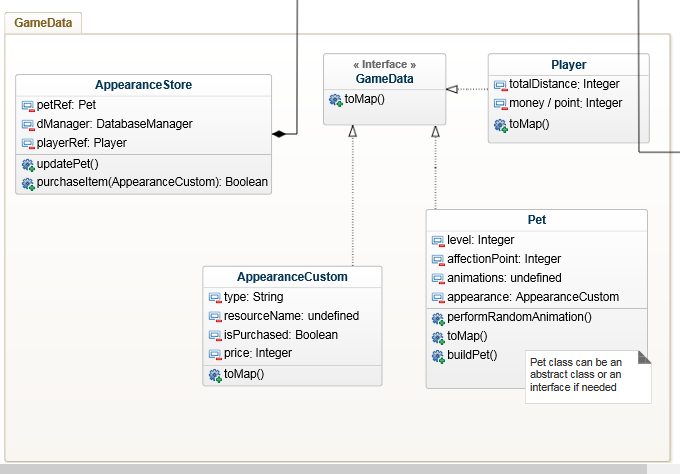
The QuestInput component also provides a QuestInputObserver interface that will notify the Main component to update game progress whenever an interaction is taken.

## Component: Database Component



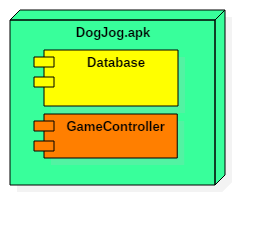
For now, the DatabaseComponenet is simple with one or two concrete classes, corresponding to saving different type of progress. Simple information will be saved and loaded with PlayerRef while more complex or set of data will be saved using relational database.

## Component: Game Data



GameData Component includes data classes and other components related directly to data.

# Deployment



* As shown in the figure, we deploy all components as well as database inside the product.
* The DogJog.apk depends on an internal database to store all necessary information. The GameController is responsible for every displays and calls by the users.

# C:\Users\Phy\AppData\Local\Microsoft\Windows\INetCache\Content.Word\ImplementationView.pngImplementation View