System and Software Architecture Description (SSAD)

<Populic>

<Team 4>

<Team members and roles>

Chengyu Shen Project manager, Tester
Shiji Zhou Prototype
Yufei Hong Feasibility Analyst
Guanghe Cao Software Architect, UML modeler
Yang Wei Operational Concept Developer
Lin Xia Life Cycle Planner
William Goishi IIV&V, Quality Focal Point, Tester

<10/07/2017>

Version History

Date	Author	Version	Changes made	Rationale
10/07/2017	Guanghe Cao	1.0	• The Introduction and system analysis part of the report	• Initial draft for the whole project

Table of Contents

Table of Tables

Table of Figures

1. Introduction

1.1 Purpose of the SSAD

This document is going to elaborate the system analysis, system design and it model, and architectural style, parttern and framework. The main part in this document is using system context diagram and use case to show the key properties of the system.

1.2 Status of the SSAD

This document is the first version. It contains the first two chapter, which is the introduction and the system analysis.

2. System Analysis

2.1 System Analysis Overview

The purpose of the challenge system is to appeal more user join in our app. In the original Map interface, we add a button to guide user open the ‘Challenge’ page, and take part in to the daily challenge, which is posted by us. The users have to challenge their friend to make a pair to compete with each other who can finish the challenge game first. Any user who can finish the challenge will have some point based on the time they spend. Also, user could invite their friend using contact list, which could make more people download the app.

2.1.1 System Context

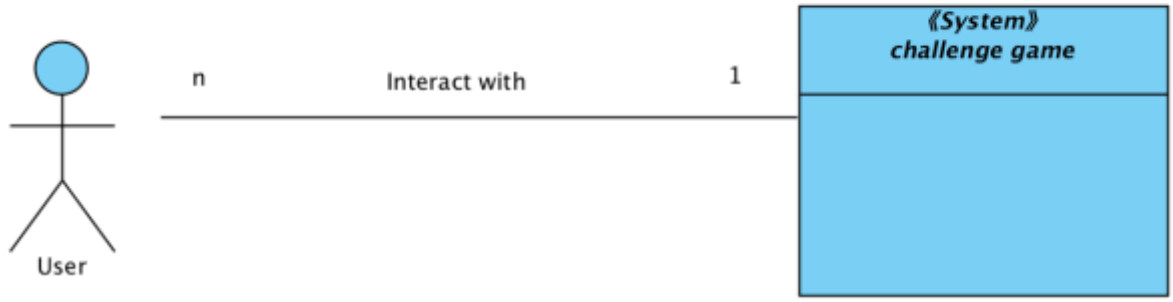

Figure 1: System Context Diagram

Table 1: Actors Summary

Actor	Description	Responsibilities
User	General user taking part in the challenge game.(We only have one kinds of user)	• Challenge other user. • Accept other user's challenge. • Confirm or decline other challenge result. • Post their challenge result. • Submit their idea of challenge.

2.1.2 Artifacts & Information

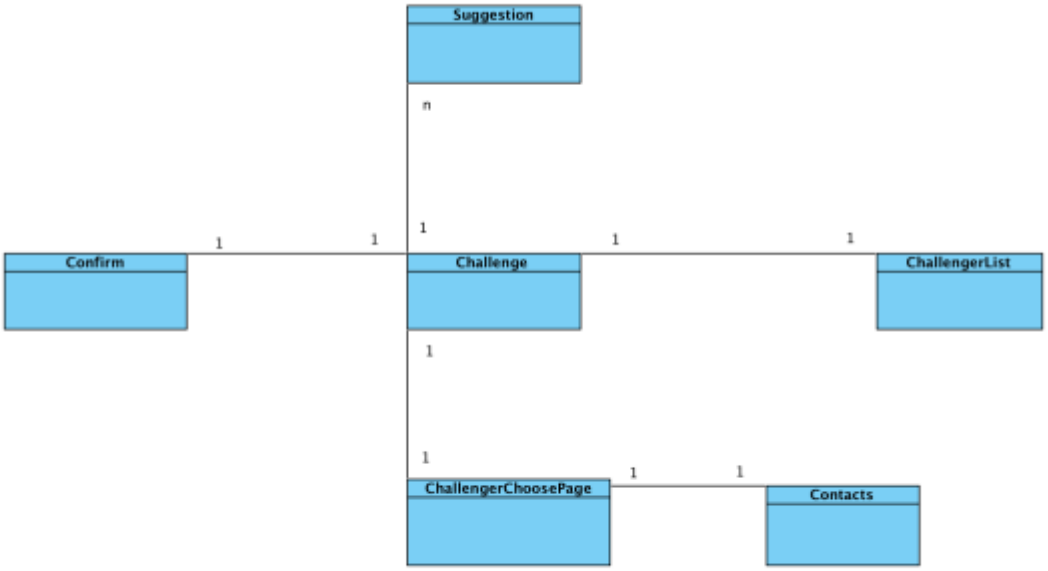

Figure 2: Artifacts and Information Diagram

Table 2: Artifacts and Information Summary

Artifact	Purpose
Challenge	To help users know what's the daily challenge and upcoming challenge, and guide user to other page.
ChallengerChooserPage	To help users choose a friend from friend list.
Contact	To help user contact the friend who never download this app before.

ChallengerList	To help user choose one of other users who have challenge this user.
Confirm	To help user inspect whether their opponent have finish the challenge or not.
Suggestion	To help us get more suggestion of the challenge idea from the users.

2.1.3 Behavior

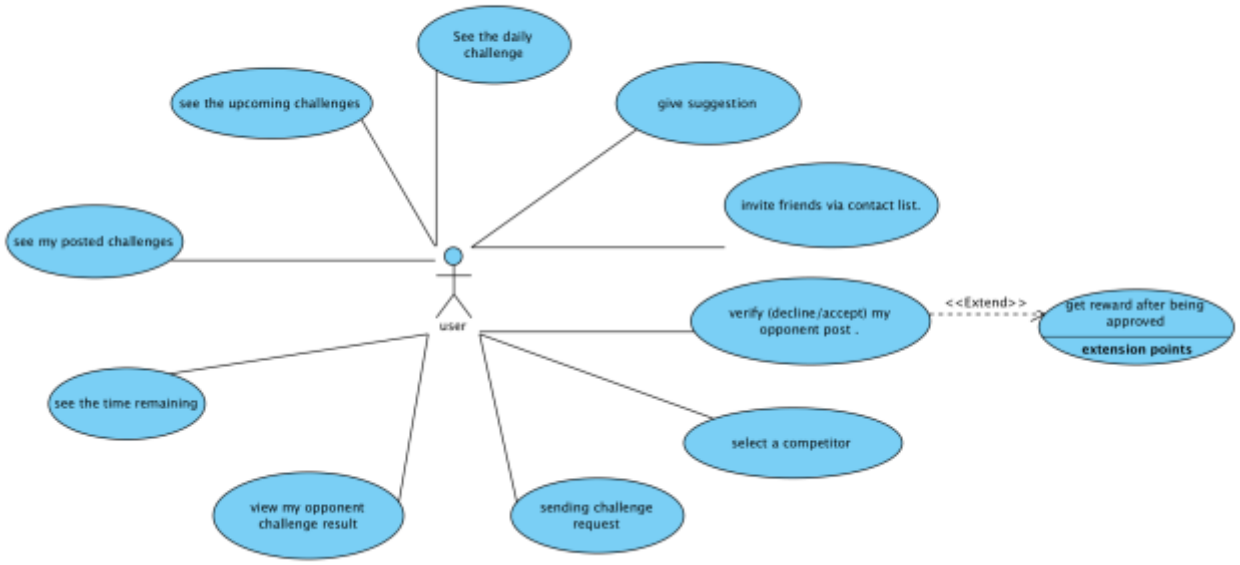

Figure 3: Process Diagram

2.1.3.1 Capability view

2.1.3.1.1 Process daily challenge

Table 3: Process Description

Identifier	UC-1: seeing the daily challenge
Purpose	Make user interested in taking part in challenge.
Requirements	
Development Risks	None
Pre-conditions	User click the challenge button.

Post-conditions	User is able to see the content of daily challenge.

Table 4: Typical Course of Action

Seq#	Actor's Action	System's Response
1	Open challenge page	
2		Show the daily challenge

2.1.3.1.2 Process upcoming challenge

Table 5: Process Description

Identifier	UC-2: seeing the upcoming challenge
Purpose	Make user more interested in this challenge game.
Requirements	
Development Risks	None
Pre-conditions	User click the challenge button.
Post-conditions	User is able to see the content of upcoming challenge.

Table 6: Typical Course of Action

Seq#	Actor's Action	System's Response
1	Open challenge page	
2		Show the upcoming challenge.

2.1.3.1.3 Process time remaining

Table 7: Process Description

Identifier	UC-3: seeing the time remaining of the challenge
Purpose	Make user more urgent to finish the challenge.
Requirements	
Development Risks	None
Pre-conditions	User click the challenge button.
Post-conditions	User is able to see the time remaining of the challenge.

Table 8: Typical Course of Action

Seq#	Actor's Action	System's Response

1	Open challenge page	
2		Show the time remaining of the challenge.

2.1.3.1.4 Process challenge result

Table 9: Process Description

Identifier	UC-4: seeing own post challenge result.
Purpose	Make users make sure what they have done.
Requirements	
Development Risks	None
Pre-conditions	User click the own challenge list button.
Post-conditions	User is able to see own challenge result.

Table 10: Typical Course of Action

Seq#	Actor's Action	System's Response
1	click the own challenge list button	
2		Show own challenge result.

2.1.3.1.5 Process y

Table 11: Process Description

Identifier	UC-5: seeing opponent challenge result.
Purpose	Make users see what their opponent have done.
Requirements	
Development Risks	None
Pre-conditions	User click opponent's challenge list button.
Post-conditions	User is able to see opponent challenge result.

Table 12: Typical Course of Action

Seq#	Actor's Action	System's Response
1	click opponent's challenge list button.	
2		Show opponent challenge result.

2.1.3.2 **Capability** challenging

2.1.3.2.1 **Process** sending challenge

Table 13: Process Description

Identifier	UC-6: send challenge request
Purpose	Make more users take part in the challenge game.
Requirements	User has friends in the app.
Development Risks	None
Pre-conditions	User click the challenge button.
Post-conditions	User is able to see the content of daily challenge.

Table 14: Typical Course of Action

Seq#	Actor's Action	System's Response
1	User clicks the challenge button.	
2		Showing friends list
3	User clicks the friends that he/she want to challenge	
4		Sending the challenge require to other users.

Table 15: Alternate Course of Action

Seq#	Actor's Action	System's Response
1	User clicks the challenge button.	
2		Showing friends list
3	User clicks 'go back'	
4		Return to the former page

2.1.3.2.2 **Process** select a competitor

Table 16: Process Description

Identifier	UC-7: select a competitor from the challenger list
Purpose	Make a pair that compete with each other
Requirements	User has friends who challenge him/her.
Development Risks	None
Pre-conditions	User click the challenger list button.

Post-conditions	User is able to have a competitor to compete about the challenge.

Table 17: Typical Course of Action

Seq#	Actor's Action	System's Response
1	User clicks the challenge list button.	
2		Showing challengerlist
3	User clicks the friends that he/she want to challenge	
4		Make a pair successfully.

2.1.3.2.3 **Process** daily challenge

Table 18: Process Description

Identifier	UC-8: verify competitor challenger result.
Purpose	Make sure that the user finish the challenge.
Requirements	User has posted the challenge result.
Development Risks	None
Pre-conditions	User click opponent challenger list button.
Post-conditions	User is able to approve or decline their opponent's challenge result.

Table 19: Typical Course of Action

Seq#	Actor's Action	System's Response
1	User clicks the opponent challenge list button.	
2		Showing result of opponent's result
3	User clicks approve button	
4		Opponent get the reward.

Table 20: Alternate Course of Action

Seq#	Actor's Action	System's Response
1	User clicks the opponent challenge list button.	
2		Showing result of opponent's result
3	User clicks decline button	

n		Opponent's result cancel and have to redo the challenge.

2.1.3.2.4 **Process** invite

Table 21: Process Description

Identifier	UC-9: invite friend from contact list.
Purpose	Make more users download the app.
Requirements	User want to challenge the friend who do not have the app
Development Risks	None
Pre-conditions	User click contact list button.
Post-conditions	User is able to send a challenge message to friend.

Table 22: Typical Course of Action

Seq#	Actor's Action	System's Response
1	User clicks contact list button.	
2		Showing result of contact list
3	User clicks the person he/she wants to challenge.	
4		Challenge invitation is sent.

2.1.3.3 **Capability** suggestion

Table 23: Process Description

Identifier	UC-10:giving suggestion.
Purpose	To get more idea of the challenge topic.
Requirements	User have a new idea about the topic of the challenge
Development Risks	None
Pre-conditions	User click the suggestion button.
Post-conditions	We can get the idea of the challenge topic

Table 24: Typical Course of Action

Seq#	Actor's Action	System's Response
1	User clicks suggestion button.	
2		Showing suggestion page.
3	User write ideas and submit	
4		Getting suggestion.

2.1.4 Modes of Operation

We don't have more than one modes of operation in our system, so we will not contain any modes in this part.

2.2 System Analysis Rationale

In our team, there is no one who can design a good interface for the whole challenge process. And the interface is a little bit complicated and ugly. Given the user want to take part into the challenge game but without any impetus to figure out how to play the challenge game, it will be very dangerous for the app because the user would go away from the challenge game. Thus, our purpose is not accomplished because non-technology staff. The above mention is the aspects of analysis that are deemed by the team.

3. Technology-Independent Model

3.1 Design Overview

3.1.1 System Structure

<< This section should contain

- a conceptual domain model
- a UML hardware component class diagram
- a UML software component class diagram
- a UML deployment diagram
- If necessary, a class diagram for the system's supporting software infrastructure
- and descriptions of the hardware components, software components, and, if necessary, the supporting software infrastructure components of the technology/platform-independent system architecture

More information and example can be found in **ICM EPG> Task: Define Technology-Independent Architecture >>**

<<Conceptual Domain Model>>

Figure 4: Conceptual Domain Model

<<Hardware Component Class Diagram>>

Figure 5: Hardware Component Class Diagram

<<Software Component Class Diagram>>

Figure 6: Software Component Class Diagram

<<Deployment Diagram>>

Figure 7: Deployment Diagram

<<Optional: Supporting Software Infrastructure Diagram>>

Figure 8: Supporting Software Component Class Diagram

Table 25: Hardware Component Description

Hardware Component	Description

Table 26: Software Component Description

Software Component	Description

Table 27: Supporting Software Component Description

Support Software Component	Description

3.1.2 Design Classes

This section should contain:

- UML class diagrams showing all the boundary, entity, and control classes in the design of the system being developed
- and a description of each class in the diagram

More information and example can be found in **ICM EPG> Task: Define Technology-Independent Architecture** >>

3.1.2.1<Classes n>

<<Design Classes Class Diagram>>

Figure 9: Design Class Diagram

Table 28: Design Class Description

Class	Type	Description

3.1.3 Process Realization

<< This section shows how the proposed architecture can be realized by conducting robustness analysis and constructing sequence diagrams. Make sure the elements used in robustness diagrams or sequence diagrams are traceable to the design classes, for example, by using the same names, or annotating the elements with the name(s) or id(s) of the referenced design class(s). More information and example can be found in **ICM EPG> Task: Define Technology-Independent Architecture** >>

<<Robustness Diagram>>

Figure 10: Robustness Diagram

<<Sequence Diagram>>

Figure 11: Sequence Diagram

3.2 Design Rationale

<< This section should contain an explanation of how/why the architecture/design described in previous sections was chosen. More information and example can be found in **ICM EPG> Task: Define Technology-Independent Architecture** >>

4. Technology-Specific System Design

<< Once you know specific technology that you team is going to use, design the system and software architecture and document them in this section. >>

4.1 Design Overview

4.1.1 System Structure

<<Hardware Component Class Diagram>>

Figure 12: Hardware Component Class Diagram

<<Software Component Class Diagram>>

Figure 13: Software Component Class Diagram

<<Deployment Diagram>>

Figure 14: Deployment Diagram

<<Optional: Supporting Software Infrastructure Diagram>>

Figure 15: Supporting Software Component Class Diagram

Table 29: Hardware Component Description

Hardware Component	Description

Table 30: Software Component Description

Software Component	Description

Table 31: Supporting Software Component Description

Support Software Component	Description

4.1.2 Design Classes

4.1.2.1 <Classes n>

<<Design Classes Class Diagram>>

Figure 16: Design Class Diagram

Table 32: Design Class Description

Class	Type	Description

4.1.3 Process Realization

<<Make sure the elements used in robustness diagrams or sequence diagrams are traceable to the design classes, for example, by using the same names, or annotating the elements with the name(s) or id(s) of the referenced design class(s). >>

<<Robustness Diagram>>

Figure 17: Robustness Diagram

<<Process Realization Diagram>>

Figure 18: Process Realization Diagram

4.2 Design Rationale

5. Architectural Styles, Patterns and Frameworks

<< Describe any implementation architecture styles (e.g. the Prism style and 3-tier architecture), patterns (e.g. pipe-and-filter and client-server), or frameworks (e.g. Java and CORBA) used to describe the system architecture. >>

Table 33: Architectural Styles, Patterns, and Frameworks

Name	Description	Benefits, Costs, and Limitations