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

<12/04/2017>

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
10/11/2017	Guanghe Cao	1.1	• Add the technology-specific model and the frameworks	• Second version for the project
11/29/2017	Guanghe Cao	2.0	• Change the Process Realization Diagram and the class diagram	• Initial draft for the As-Built Package
12/04/2017	Guanghe Cao	2.1	• Fixed the mistake in the 1.1 version.	• Final draft for the As-Built Package

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, pattern 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 final 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

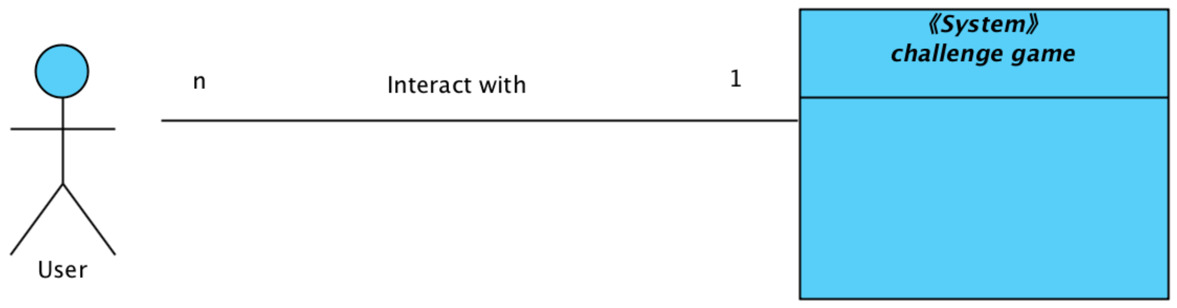


Figure 1: System Context Diagram

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 users. • 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

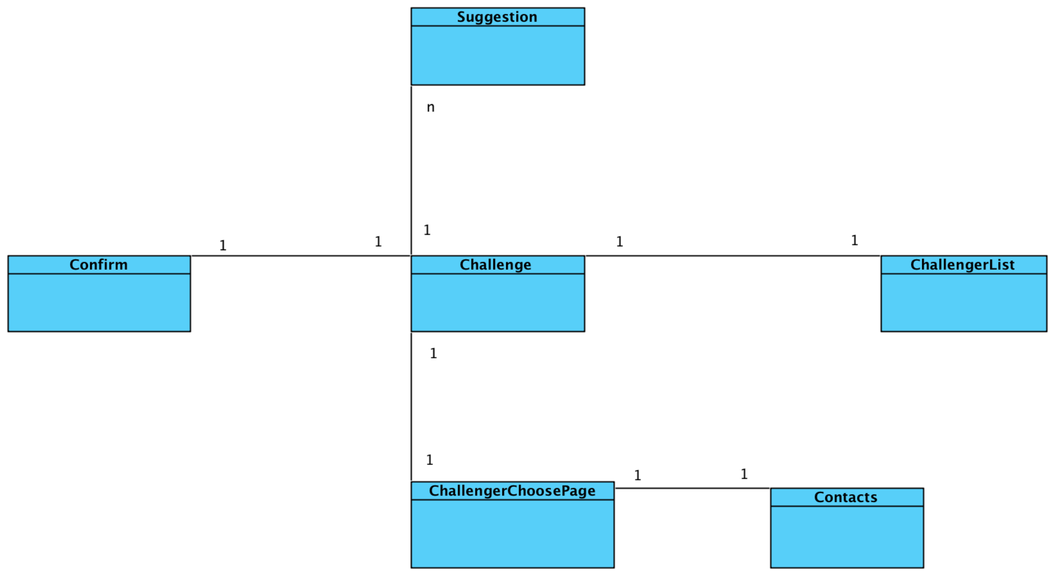


Figure 2: Artifacts and Information Diagram

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 pages.
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

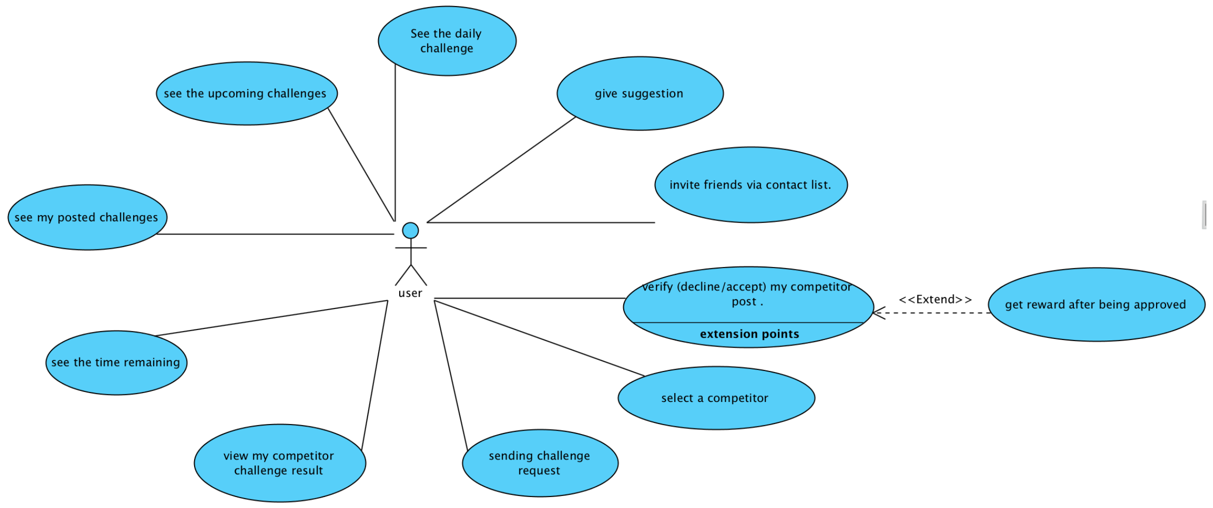

See the daily challenge
see the upcoming challenges
give suggestion
invite friends via contact list.
see my posted challenges
user
verify (decline/accept) my competitor post .
extension points
<<Extend>>
get reward after being approved
see the time remaining
select a competitor
view my competitor challenge result
sending challenge request

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: See 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: See 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: See 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: See 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

This section is blank because our project is technology-specific model. So you can refer to the next chapter for the who content as required in the template.

4. Technology-Specific System Design

4.1. Design Overview

4.1.1. System Structure

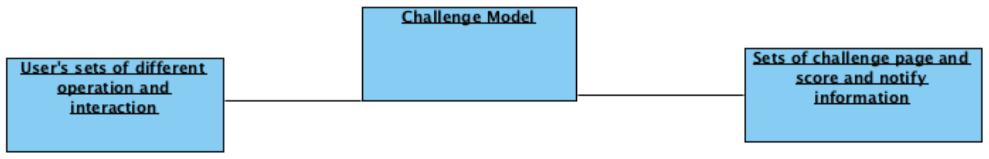

User's sets of different operation and interaction
Challenge Model
Sets of challenge page and score and notify information

Figure 4: Conceptual Domain Diagram

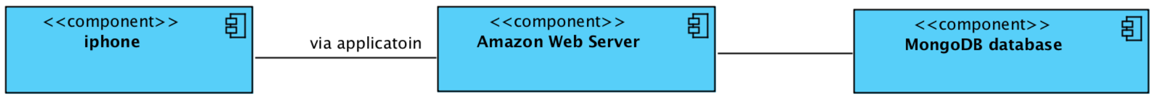


Figure 5: Hardware Component Class Diagram

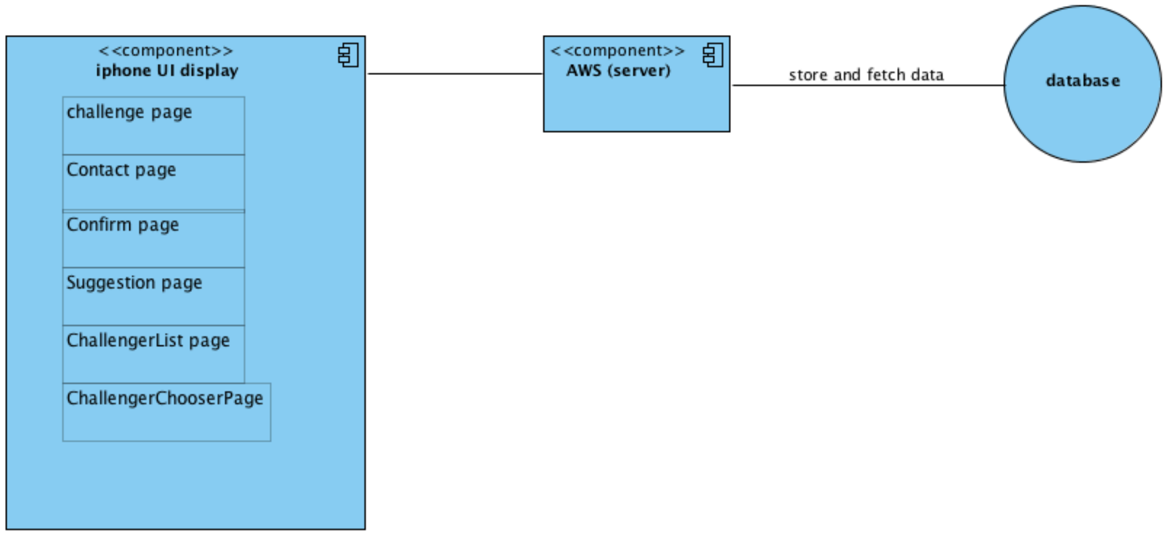


Figure 6: Software Component Class Diagram

Table 25: Hardware Component Description

Hardware Component	Description
Iphone	Any iphone that download populic application
Amazon Web Service	A web server that could provide the service for the whole app
MongoDB database	A NoSQL database that store users' profile, post and game info

Table 26: Software Component Description

Software Component	Description
Challenge	The challenge that every user could get when joining in the challenge game part.
ChallengerList	A list that the user has for choosing a challenger to challenge with.
Contact	A contact list that user has for inviting other users who have not download this app.
Suggestion	A suggestion part that could help users give more suggestion for the idea of the challenge topic
Confirm	A component that could help user go confirm opponent's challenge result
ChallengerChooserPage	A page that could helper users choose one of the challenger
AWS (server)	For we use Amazon Web Server, we do not have to worry about the maintain of the server, so it could help us maintain the app very vell
DataBase	It could help us store the data of the user and challenge.

4.1.2. Design Classes

4.1.2.1. <User Classe>

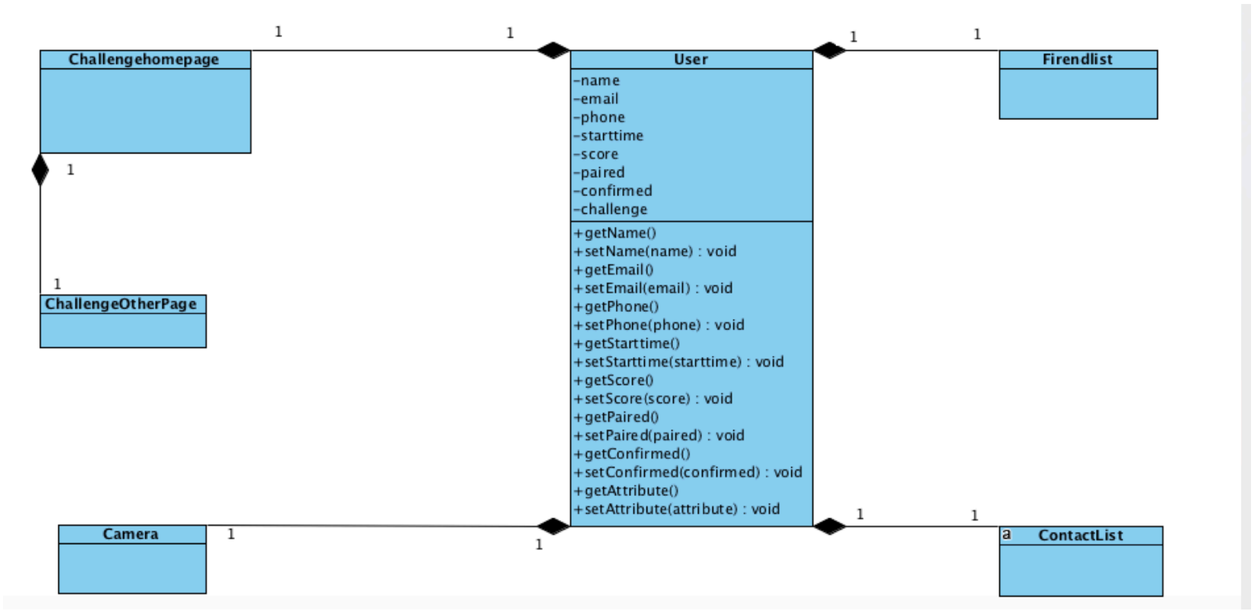
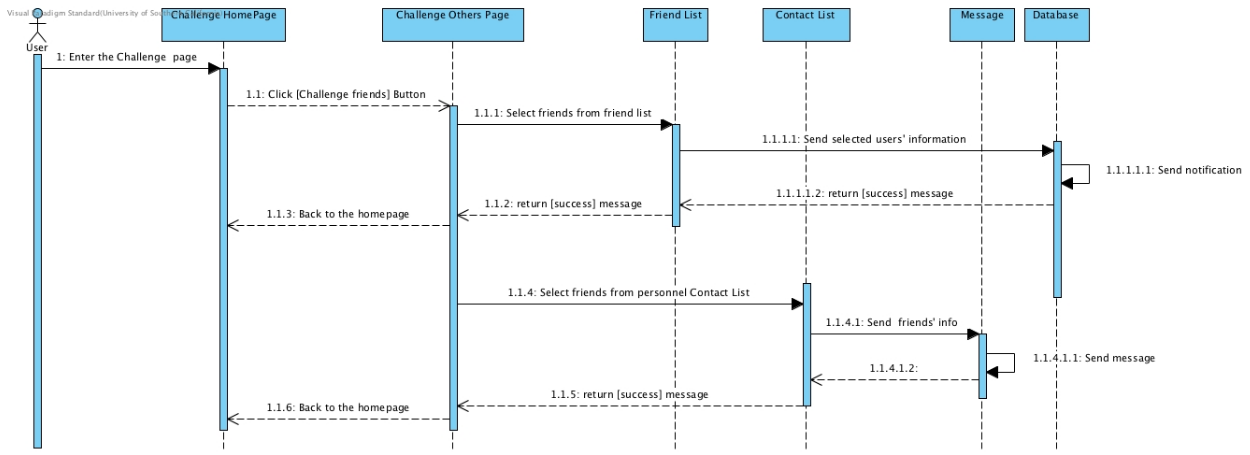


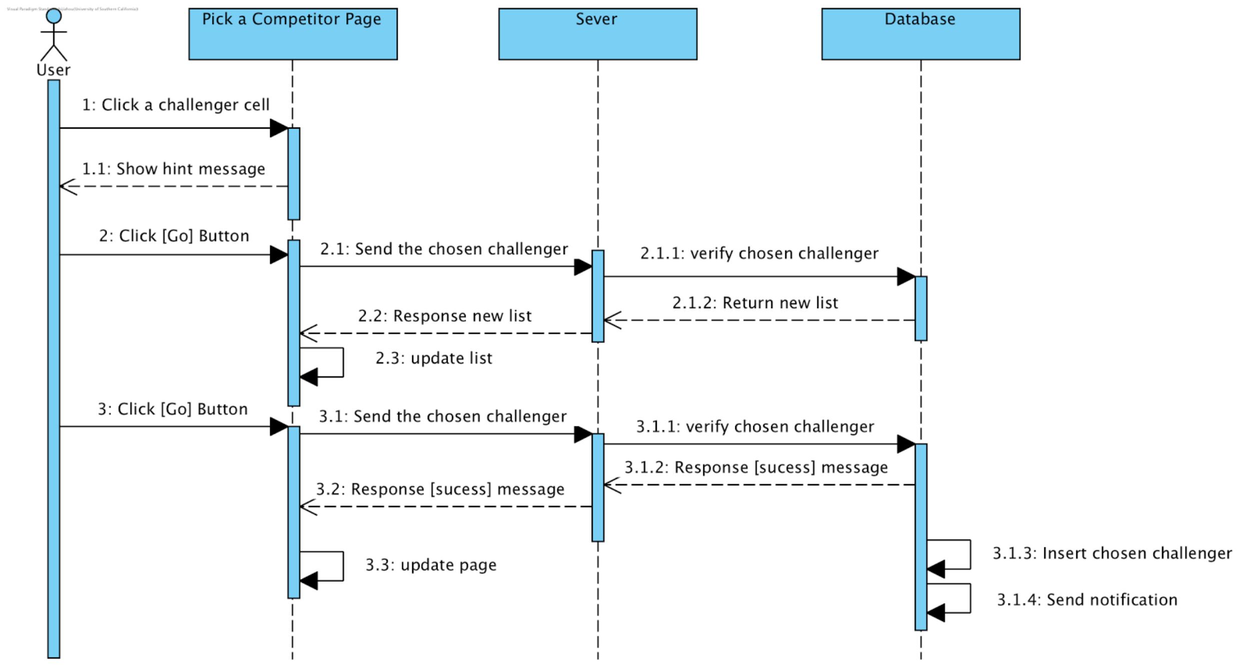
Figure 7: Design Class Diagram

Table 27: Design Class Description

Class	Type	Description
Challenghomepage	Controller	This implement the challenge function(Challenghomepage)
User	Controller	This is the user for the who app
FriendList	Entity	This is the list of friend for user to send challenge.
ContactList	Entity	This is the list of contact for user to send challenge.
ChallengeOtherPage	boundary	This is the page for challenge other
Camera	boundary	This is a tool for user to do the challenge

4.1.3. Process Realization


User
Challenge HomePage
Challenge Others Page
Friend List
Contact List
Message
Database
1: Enter the Challenge page
1.1: Click [Challenge friends] Button
1.1.1: Select friends from friend list
1.1.1.1: Send selected users' information
1.1.1.1.1: Send notification
1.1.1.1.2: return [success] message
1.1.2: return [success] message
1.1.3: Back to the homepage
1.1.4: Select friends from personnel Contact List
1.1.4.1: Send friends' info
1.1.4.1.1: Send message
1.1.4.1.2:
1.1.5: return [success] message
1.1.6: Back to the homepage


User
Pick a Competitor Page
Sever
Database
1: Click a challenger cell
1.1: Show hint message
2: Click [Go] Button
2.1: Send the chosen challenger
2.1.1: verify chosen challenger
2.1.2: Return new list
2.2: Response new list
2.3: update list
3: Click [Go] Button
3.1: Send the chosen challenger
3.1.1: verify chosen challenger
3.1.2: Response [sucess] message
3.2: Response [sucess] message
3.3: update page
3.1.3: Insert chosen challenger
3.1.4: Send notification

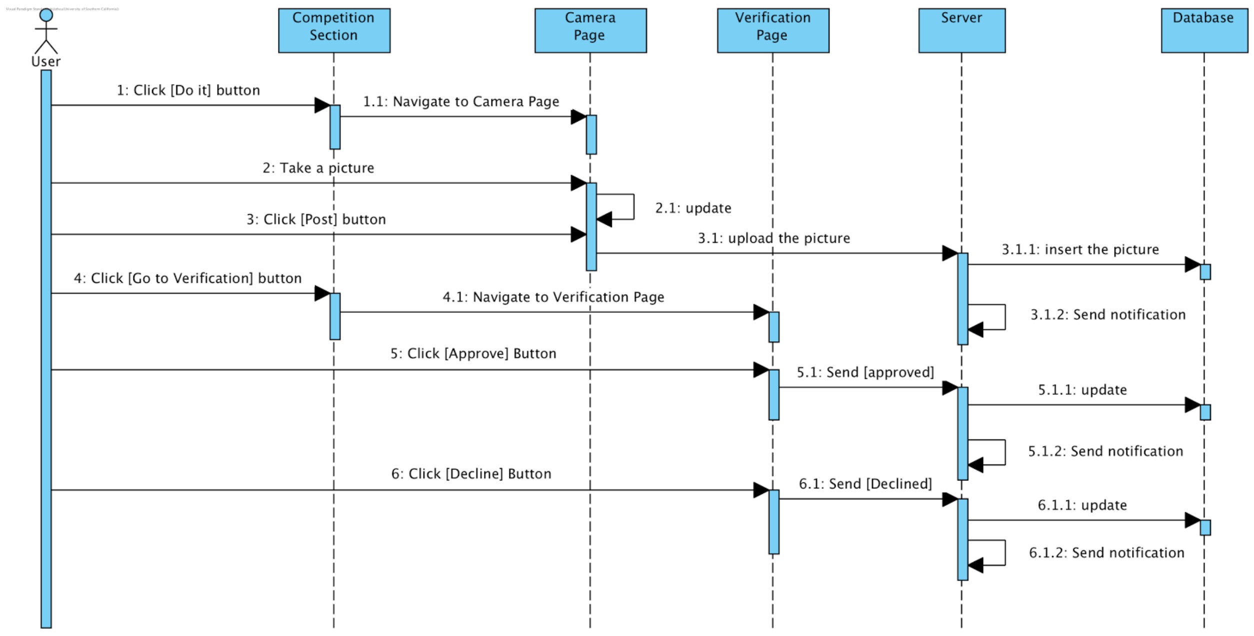


Figure 8: Process Realization Diagram

4.2. Design Rationale

The challenge game part we are designing is one of the functions that should be added in the current map. Thus, in the map, there is a button, which guides us to the page of challenge game part. When user enter into the challenge game part, there is a sign for us to challenge our friends. When we enter the page that we choose a friend to challenge with, there is two types of list we can choose. One is the friend list that we have in this app, the other is the contact that we get from user's phone. When user send the invitation, the user who get the invitation could see a button that guide us to the page we choose one of the challenger who have sent the invitation. Then when the user make a pair and finish the challenge by posting the result, the challenge will have a confirm button for user to confirm. Then user could confirm opponent's result, and then opponent will get the points after that.

For the who architecture, it is a three-tier architecture pattern.

1. UI display(iPhone)
2. Logic of the whole challenge game(AWS)
3. Database(MongoDB)

The UI display could show basic page connection of the challenge part without the logic of the game. In the server part, which is Amazon Web Server, we add the logic of the challenge game part, so that we could help users page turn correctly. And also, we need connect the database, which is MongoDB, with the server, so to store the user's profile, post and game information.

5. Architectural Styles, Patterns and Frameworks

Table 28: Architectural Styles, Patterns, and Frameworks

Name	Description	Benefits, Costs, and Limitations
Three-tire architecture	This is a architecture style that contain what I mention before, which are UI display, Amazon Web Service, MongoDB. They can not work without any other part in our project. UI display do not have any logic of the challenge game, and it also does not have any data (e.g. post, profile, game information) that store in UI display. It has to connect with the server, which is AWS (Amazon Web Service), and this kind of backend could help the user guide into the correct page, or connect the MongoDB to get the data from it to return it back to the UI display, so these parts is a three-tire architecture for the project.	Benefit: it could help the who project maintain very well because the server side could provide a very stable service for the project. Also, the who architecture could handle many case in the project, like one of the three lays can not work, it could not have much more damage for the other layer Cost: Free Limitation: We need to change all the layer's code if we want to change a feature.
Client-server	For our project, Amazon Web Service is the server that provide the function for the project, and the client could run the app by the service provided by AWS. It is also a part of the three-tire architecture.	Benefit: separate the frontend and the backend. Cost: free Limitation: can not maintain as many as users.