

**MINISTRY OF**

**EDUCATION AND TRAINING**

**FPT UNIVERSITY**

Capstone Project Document

**Corporate Training System**

|  |  |
| --- | --- |
| **GROUP 02** | |
| **Group members** | Lê Phúc Thịnh – SE61759 (Leader)  Võ Thạch Nguyên – SE61778  Phạm Chánh Hưng – SE61807  Đặng Thế Anh – SE61372 |
| **Supervisor** | Nguyễn Huy Hùng |
| **Ext. Supervisor** | N/A |
| **Capstone Project code** | CTS |

- Ho Chi Minh City, ***8th January, 2018*** -

**

**CAPSTONE PROJECT REGISTER**

Class: Duration time: From 08/01/2018 To 25/04/2018

(\*) Profession: <Software Engineer> Specialty: <ES> <IS>

x

(\*) Kinds of person make registers: Lecturer Students

x

1. Register information for supervisor (if have)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Full name** | **Phone** | **E-Mail** | **Title** |
| Supervisor 1 | Nguyễn Huy Hùng |  | hungnh@fpt.edu.vn | Mr. |

2. Register information for students (if have)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Full name** | **Student code** | **Phone** | **E-mail** | **Role in Group** |
| Student 1 | Lê Phúc Thịnh | SE61759 | 0961173093 | [thinhlpse61759@fpt.edu.vn](mailto:thinhlpse61759@fpt.edu.vn) | Leader |
| Student 2 | Võ Thạch Nguyên | SE61778 | 01233461809 | [nguyenvtse61778@fpt.edu.vn](mailto:nguyenvtse61778@fpt.edu.vn) | Member |
| Student 3 | Phạm Chánh Hưng | SE61807 | 0936210777 | [hungpcse61807@fpt.edu.vn](mailto:hungpcse61807@fpt.edu.vn) | Member |
| Student 4 | Đặng Thế Anh | SE61372 | 0981434867 | [anhdtse61372@fpt.edu.vn](mailto:anhdtse61372@fpt.edu.vn) | Member |

3. Register content of Capstone Project

(\*) 3.1. Capstone Project name:

English: Corporate Training System

Vietnamese: Hệ Thống Đào Tạo Cho Doanh Nghiệp

Abbreviation: CTS

(\*) 3.2. Main proposal content (including result and product)

1. Theory and practice (document):
   * Student should apply the software development process and the UML 2.0 in modeling the system
   * Software artifacts include User Requirement, Software Requirement Specification, Architecture Design, Detail Design, System Implementation and Testing Document, Installation Guide, sources code, and deployable software packages
   * Server side technique:
     + Database design, OOA, OOD, OOP, MVC, Java or .Net technology, …
     + Apache Lucene, ElasticSearch
   * Client side technique
     + HTML5, CSS, JavaScript, JQuery, Ajax
     + Mobile Platform (iOS, Android)
2. Program:

Building a web platform that allows corporate to deploy their own training programs. The following main features should be implemented:

- Manage corporate brand: each corporate should be about to customize their own brand name, logo, and related information

- Manage users: there are 4 type of roles: Admin, Manager, Author, Learner. Admin and Manager should be able to create accounts for Author and Learners, as well as manually enroll Learners to a specific course. Admin can change user’s roles

- Manage programs: Admin/Manager can create a training program. Each program comprises of sequence of courses. A program can be self-paced or deadline

- Manage training plans: Admin/Manager can create training plans: starting and ending programs/courses

- Manage course: Author can create courses.

Courses can include: video, text, PowerPoint, quiz.

Passing a quiz might be required before proceeding to next lecture/course

Each course belongs to skills

* + - * Manage reports: Admin/Manager can view Learners’ report on learning activities

1. Other products:

* All of management functions of the system must be implemented to support the operating system.

4. Other comment (propose all relative thing if have)

|  |  |
| --- | --- |
| **Supervisor (If have)**  *(Sign and full name)* | HCM city, date 12/12/2017  **On behalf of Registers**  *(Sign and full name)* |

Nguyễn Huy Hùng

# Table of Contents

[A. Introduction 1](#_Toc490427359)

[1. Project Overview 1](#_Toc490427360)

[2. Introduction 1](#_Toc490427361)

[3. Current Situation 1](#_Toc490427362)

[4. Problem Definition 1](#_Toc490427363)

[5. Proposed Solution 2](#_Toc490427364)

[5.1 Featured functions 2](#_Toc490427365)

[5.2 Advantages 2](#_Toc490427366)

[5.3 Disadvantages 2](#_Toc490427370)

[B. Software Project Management Plan 2](#_Toc490427373)

[1. Problem Definition 2](#_Toc490427374)

[1.1 Name of this Capstone Project 2](#_Toc490427375)

[1.2 Problem Abstract 2](#_Toc490427376)

[1.3 Project Overview 3](#_Toc490427377)

[2. Project organization 5](#_Toc490427378)

[2.1 Software Process Model 5](#_Toc490427379)

[2.2 Roles and Responsibilities 6](#_Toc490427380)

[2.3 Tools and Techniques 6](#_Toc490427381)

[3. Project Management Plan 7](#_Toc490427382)

[3.1 Product Backlog 7](#_Toc490427383)

[3.2 Sprint Backlog 7](#_Toc490427384)

[3.3 Deliverables 7](#_Toc490427385)

[3.4 All Meeting Minutes 7](#_Toc490427386)

[C. Software Requirement Specifications 8](#_Toc490427387)

[1. User Requirement Specification 8](#_Toc490427388)

[1.1 Guest Requirement 8](#_Toc490427389)

[1.2 Learner Requirement 8](#_Toc490427390)

[1.3 Teacher Requirement 8](#_Toc490427391)

[1.4 Staff Requirement 8](#_Toc490427392)

[1.5 Admin Requirement 8](#_Toc490427393)

[2. System Requirement Specification 9](#_Toc490427394)

[2.1 System Overview Use Case 9](#_Toc490427395)

[3. Conceptual Diagram 10](#_Toc490427396)

[D. Software Design Description 12](#_Toc490427397)

[1. Design Overview 12](#_Toc490427398)

[2. System Architectural Design 12](#_Toc490427399)

[2.1 Web service architecture description 12](#_Toc490427400)

[2.2 Mobile Application Architecture Description 13](#_Toc490427401)

[3. Component Diagram 14](#_Toc490427402)

[4. Detailed Description 17](#_Toc490427403)

[4.1 Class Diagram 17](#_Toc490427404)

[5. Database Design 19](#_Toc490427405)

[5.1 Entity Relationship Diagram (ERD) 19](#_Toc490427406)

[5.2 Data Dictionary 20](#_Toc490427407)

[6. Algorithms 20](#_Toc490427408)

[6.1 Recommendation 20](#_Toc490427409)

[E. Tasksheet 28](#_Toc490427410)

[F. Appendix 32](#_Toc490427411)

[1. UML Documentation of IBM 32](#_Toc490427412)

[2. React Native 32](#_Toc490427413)

[3. WebRTC 32](#_Toc490427414)

[4. Play Framework 2.5 32](#_Toc490427415)

[5. Redis Cache 32](#_Toc490427416)

[6. Firebase Cloud Message 32](#_Toc490427417)

[7. Firebase Realtime Database 32](#_Toc490427418)

[8. Dictionary API 32](#_Toc490427419)

# List of Tables

[*Table 1: Definitions, Acronyms, and Abbreviations* 11](#_Toc490226345)

[*Table 2: Roles and Responsibilities* 17](#_Toc490226346)

[*Table 3: Tools and Techniques* 18](#_Toc490226347)

[*Table 4: Deliverables* 18](#_Toc490226348)

[*Table 5: Conceptual Diagram Data Dictionary* 22](#_Toc490226349)

[*Table 6: Component Diagram Dictionary* 27](#_Toc490226350)

[*Table 7: Class Diagram Dictionary* 29](#_Toc490226351)

[*Table 8: ERD dictionary* 31](#_Toc490226352)

# List of Figures

[*Figure 1: Scrum Process* 16](#_Toc490226092)

[*Figure 2: Use case overview* 20](#_Toc490226093)

[*Figure 3: Conceptual Diagram* 21](#_Toc490226094)

[*Figure 4: Web service Architecture* 24](#_Toc490226095)

[*Figure 5: Mobile Application Architecture* 25](#_Toc490226096)

[*Figure 6: Component Diagram* 26](#_Toc490226097)

[*Figure 7: Class Diagram* 28](#_Toc490226098)

[*Figure 8: Entity Relationship Diagram* 30](#_Toc490226099)

[*Figure 9: Percentage of completed lessons flow chart* 36](#_Toc490226100)

[*Figure 10: Build Dataset Flow Chart* 36](#_Toc490226101)

[*Figure 11: Build New User Dataset Flow Chart* 37](#_Toc490226102)

[*Figure 12: User- based Recommendation Flow Chart* 38](#_Toc490226103)

# Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| MOLA | Mobile Based Language Learning Application |
| UI | User Interface |
| UX | User Experience |
| MVVM | Model – View – View Model |
| ORM | Object Relation Mapping |
| TDD | Test Driven Development |

*Table 1: Definitions, Acronyms, and Abbreviation*

# Introduction

## Project Overview

* Project name: **Corporate Training System**
* Project Code: **CTS**
* Product Type: **Website application**
* Start Date: **8th January, 2018**
* End Date: **April, 2018**

# Introduction

Nowadays, online learning turns out to be more and more practiced. Many corporations start to post their training programs and courses online for their employees. It represents an easy and comfortable method to achieve knowledge in almost every field, from law and accounting, to even human sciences, such as psychology and physics. We can see that e-learning is a great alternative for corporations, especially for companies which can’t afford the time or human resources to train their employees. Currently, many corporations are having problems to find a suitable system that can provide them an environment to train staffs, manage training plans and view employees report easier and more effective.

In this document, we will introduce a solution for this problem. We build a system, which allows any corporates to post their training program with ease and privacy on our website. Through our system, staffs can access to training anywhere and anytime, making learning and training more effective, faster and easier. For corporations’ admin and manager, they can manage their programs effortlessly, and view their employees’ statistics.

Besides, this document also shows our working process and our perspective in the system, designs, architecture and workflow.

# Current Situation

Corporations with an online training system find out their employees are training faster and better than those who do not have access to online learning. However, it seems not easy for each company to build their own training system.

Whenever a corporate wants to develop a system to train their employees, they usually hire a software company to build a system that suitable for their needs with a cheap price. Nevertheless, it is hard to seek a software company to acquire the two requirements above.

In conclusion, corporates train their employee can be summarized through the following ways:

1. Corporations train their employees in a traditional way: Trainers/mentors train employees directly through meetings and classes.
2. Corporations train their employees online by:
   1. Build their own system to train their employees.
   2. Share their training programs online for their employees.
   3. Use the current available training system.

# Problem Definition

From above current situation, there are some disadvantages:

* Traditional training is not efficiency for corporations.

For current available training system:

* There are not many systems for corporates to train its employees online.
* Learner cannot communicate with author to understand the content of that course if they get trouble in learning.
* Learner has to wait for their manager to accept the request before they can begin to learn a course.
* When an author creates a course, they do not need to receive permission from manager or administrator. Therefore, that course may have a bad influence on learners or inappropriate content.
* An author maybe wonders what plan or program that their courses belongs to.

# Proposed Solution

Our proposed solution is to build a system named Corporate Training System (CTS) who has troubles with finding a place to train their employees online.

Corporate Training System is a web application with following functions:

### 5.1 Feature functions

* + Manage plan, programs, skills and courses.
  + Manage users.
  + Manage corporate brand and information.
  + Participate to a plan or program.
  + Enroll in a course.
  + Learn a lecture.
  + Take quiz and view quiz result.
  + Make statistic report.
  + View learners’ report on learning activities
  + Manage personal information
  + Manage discussion.

### 5.2 Advantages and Disadvantages

* Advantages:
  + - Necessary for all corporate
    - New idea
    - Easy to use application
    - Friendly interface
* Disadvantages:
  + - Finding a suitable price for customer.
    - The cost in web application maintenance is high
    - Users’ demand change

# Software Project Management Plan

## Problem Definition

### Name of this Capstone Project

* Official name: Football Field Reservation System
* Vietnamese name: Hệ thống đặt chỗ đá bóng
* Abbreviation: FFRS

### Problem Abstract

* The system provides a mobile application for user to support reserve field online. User have 2 options to reserve field.

+ If user has enough people to play a match, user can search field where user expected to play, view timeline of field. User chooses field have free time as expected and reserve field.

+ If user wants to play with other users (we call other users is opponents), system will suggest opponent have same level with user who look for. Rating point of user is rated by opponent, field owner after match finish. Based on result, system will calculate to identify ability of user.

* Field owners will use web application to manage their field. With system, field owner can provide promotion to attract players. When having field reservation request, system will arrange and choose suitable field automatically.
* In addition, we build web application for admin to manage profit. Besides, the admin will have right to lock account for bad users.

### Project Overview

#### Current situation

Below are the problems encountered in this project:

* New technique: Some team members are new to the techniques used in the project. The team need an amount of time to get familiar with those techniques.
* Design interface: The system serves users of varying degrees. So, we need an amount of time to analysis and evaluate how to build the easy to use user interface.
* Payment: The system processes payment online so that user must have electronic wallet.

#### The Proposed System

The system will have three sub-systems:

* An API application to serve API for mobile application and web application. API application is a center to process all business logic.
* A mobile application for users to perform find opponents, reserve field, feedback opponent and field, view promotion from field owner, manage and exchange discount voucher from bonus points, view notify from system.
* A web application for field owners and administrators. Field owner perform manage fields and field timeline, promote sale off price or free services, manage profits, view notify about field reservation request. Admin manage profits and lock account of bad users.

We build an algorithm to arrange field reservation request between the fields with other effectively because many field owner own multiple fields.

We use Paypal payments portals to perform transactions in the system, using Google API to get directions for user and find nearby location of field on the map.

##### Mobile Application:

* For user:

+ Manage profile

+ Find field

+ View timeline of field to reserve

+ Find opponent

+ Reserve field

+ Manage field reservation request

+ Payment online

+ View rewards, exchange voucher from bonus points

+ View notifications

+ View reservation history

* For guest

+ Become football team

##### Web Application

* For admin:

+ Get report

+ Lock user’s account

* Staff:

+ Request lock user’s account/field owner’s account

+ Set standard price

+ Accept request become field owner

+ Distribute voucher

* For field owner:

+ Manage profile

+ Create/Get/Disable fields

+ Get free field at a specific time

+ Get timeline of field

+ Get notification about field reservation requested

+ Manage time enable

+ Disable time slot of field

+ Manage income

+ Manage promotion

* For guest:

+ Become a field owner

##### API Application

The server system takes responsibility to response all the requests and also manages and processes data.

* Provide APIs for Mobile Application, Web Application
* Suggest opponent, field
* Perform scheduled tasks

#### Boundaries of the System

The system does:

* Allow user to find fields.
* Allow user to find opponents.
* Allow user to reserve fields.
* Allow user to create a matching opponents request.
* Allow user to cancel field reservation request.
* Suggest opponents with similar level.
* Suggest field have free time as expected.
* Allow field owner manage their field.
* Notify to field owner when field is reserved.
* Notify to user when request is accepted.
* Allow admin to manage profit.
* Allow admin to view report and lock user account.

#### Future plans

Current system is concentrated on core business flow. Therefore, some supporting features are restricted for the development team. These features may be expanded in the future:

* Organize tournament to attract more users and based on the results to assess skill of users more exactly.

#### Development Environment

##### Hardware requirement

* **For server:**

|  |  |  |
| --- | --- | --- |
|  | **Minimum Requirements** | **Recommended** |
| Internet Connection | Cable (4 Mbps) | Cable (8 Mbps) |
| Operating System | Ubuntu 12.04 LTS | Ubuntu 16.04 LTS |
| Computer Processor | Intel® Core i3 1.4GHz | Intel® Core i5 2.50GHz |
| Computer Memory | 2 GB RAM | 4 GB RAM or higher |
| *Table 1: Hardware requirement for Server* | | |

* **For mobile:**

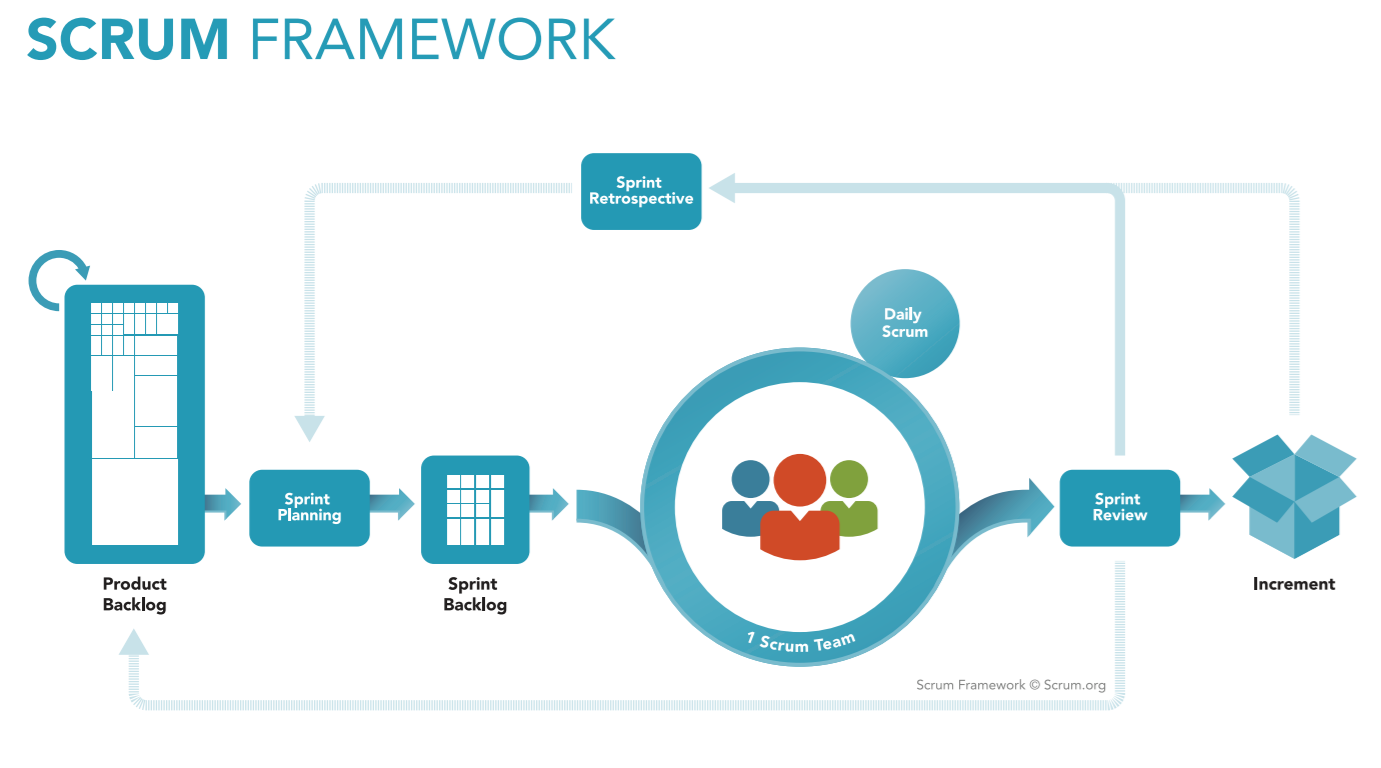
|  |  |  |
| --- | --- | --- |
|  | **Minimum Requirements** | **Recommended** |
| Internet Connection | Wi-Fi or 3G | Wi-fi or 3G, 4G |
| Operating System | Android 6 | Android 7 |
| Mobile Processor | Qualcomm Snapdragon 400 4 Cores 32-bit | [Qualcomm Snapdragon 430 8 Cores 64 bit](https://www.thegioididong.com/hoi-dap/qualcomm-snapdragon-430-844784) |
| Mobile Memory | 1 GB RAM | 2 GB RAM or higher |
| *Table 2: Hardware requirement for Mobile* | | |

## Project organization

### Software Process Model

This project is developed using Scrum model – part of an agile framework for Software development project. Our team choose Scrum model because of the following reasons:

* In the team, there are 2 members not playing football. So, all members must work together in order to discuss about business logic and avoid misunderstanding.
* In the project there are many new technologies that need to be learned. With the Scrum model, the team can learn and develop in parallel to meet deadline.
* There is no leader, no hierarchy in team, so team members work cheerfully, stimulating the initiative and creativity of each member.
* The project implements a new idea, so maybe product owner change requirement or extend scope. The team will adapt to change better.



*Figure 1: Scrum Process*

(<https://www.scrum.org/resources/what-is-scrum>)

### Roles and Responsibilities

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Role in Scrum** | **Name** | **Responsibilities** |
| 1 | Product owner | Kiều Trọng Khánh | * Specify scope and user requirement. * Supervise the development progress. * Provide professional techniques and business analysis support. |
| 2 | Scrum master | Mai Minh Quý | * Create Sprint Backlog and Product Backlog. * Make sure the Scrum teams understand and follow the process. * Always be present to answer questions and give advice when product owner or scrum member needs. * Help the team master scrum artifacts such as: Sprint Backlog, Product Backlog, ... |
| 3 | Scrum team member | Mai Minh Quý  Trương Hữu Thành  Phan Minh Huấn  Phạm Trung Hiếu | * Designing database * Clarifying requirements * Prepare documents * GUI design * Coding * Testing |
|  | | | |

*Table 2: Roles and Responsibilities*

### Tools and Techniques

|  |  |
| --- | --- |
|  | Tool/Technique |
| Mobile Application | Android SDK, Firebase |
| Web Portal | CSS, HTML5, React JS, Java Script, Firebase |
| Back-end | SpringBoot framework, Java, JPA |
| IDE | IntelliJ 2017.2.4, Android Studio 2.3.3, Visual Studio Code |
| Database | MySQL Server 5.7.19, MySQL Workbench 6.3.9 |
| Modeling tools | Star UML 2.8.0 |

*Table 3: Tools and Techniques*

## Project Management Plan

### Product Backlog

All product backlog could be found here

### Sprint Backlog

All sprint backlog could be found here

### Deliverables

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Deliverable** | **Deliverable date** | **Deliverable location** | **Note** |
| 1 | Introduction, Project Management Plan, Concept Diagram, Class Diagram, Entity Relationship Diagram, Use Case Overview, Mock UI |  |  | Sprint 1 |
| 2 | Study Spring Boot Framework, Study React, Design User Interface for Mobile Programming, Mobile Architecture, Web Service Architecture |  |  | Sprint 2 |
| 3 | Code core flow RESTful API Web services, Web application and Mobile application |  |  | Sprint 3 |
| 4 | Code core flow Web application and Mobile application; Study firebase console services; User Requirement Specification |  |  | Sprint 4 |
| 5 | Code low priority functions, Conceptual Diagram, Design Overview, System Architectural Design, Component Diagram. |  |  | Sprint 5 |
| 6 | Testing, Entity Relationship Diagram, Database Diagram, Algorithms |  |  | Sprint 6 |
|  | | | | |

*Table 4: Deliverables*

* For each Sprint, deliverables are potentially shippable products, which can be a part of document or prototype implemented based on the project’s core flow.
* Each Sprint has a fixed duration of two weeks.

### All Meeting Minutes

All meeting documents could be found here

# Software Requirement Specifications

## User Requirement Specification

### Guest Requirement

Guest is a person who loves to play football. Guest can do the following functions:

* Sign up
* Sign in

### User Requirement

User is a person who represents a football team to reserve field. User can do the following functions:

* Manage account:
  + Get profile’s information
  + Edit profile
* Search field
* Reserve field (reserve friendly match: user finds and reserves a field)
* Manage matching requests:
  + Create reservation requests
  + Cancel reservation requests
  + Reserve tour match (user finds opponent, chooses suitable field from suggest of system and reserves a match)
* Get history (user views history about created reservation request and created matching request)
* Rating field/opponent
* Send report (report bad user or bad field owner)
* Set favorites field.
* Add opponent to blacklist
* Deposit (deposit money to user’s wallet in system)

### Field Owner Requirement

Field owner is a person who has football fields and let users reserve his/her fields by using our application. The following functions are available for field owner to use:

* Manage account:
  + Get profile’s information
  + Edit profile
* Manage field:
  + Create field
  + Disable field
  + Create promotions (discount reservation fee or free services for user)
* Manage active time:
  + Create active time
  + Update active time
* Send report (report bad user)
* Set field for match
* Get bills by date

### Admin Requirement

Admin is an employee from our company. Admin is responsible for managing service provider system and accounts. Admin can do following functions:

* Lock users’ accounts

### Staff Requirement

Staff is an employee from our company. Staff solves the problem for the user and field owner. Staff can do following functions:

* Accept the field owner’s account creation request (when field owner creates an account, staff will authenticate the actual situation of field owner to approve field owner’s account)
* Get report from user (view reports from user and field owner)
* Send lock account request (request admin to lock account of bad user, field owner)
* Set standard price of field (control reservation field fee to avoid devaluation)
* Change user’s balances (when user makes a transaction at a company office, staff will use this function to solve money problems)

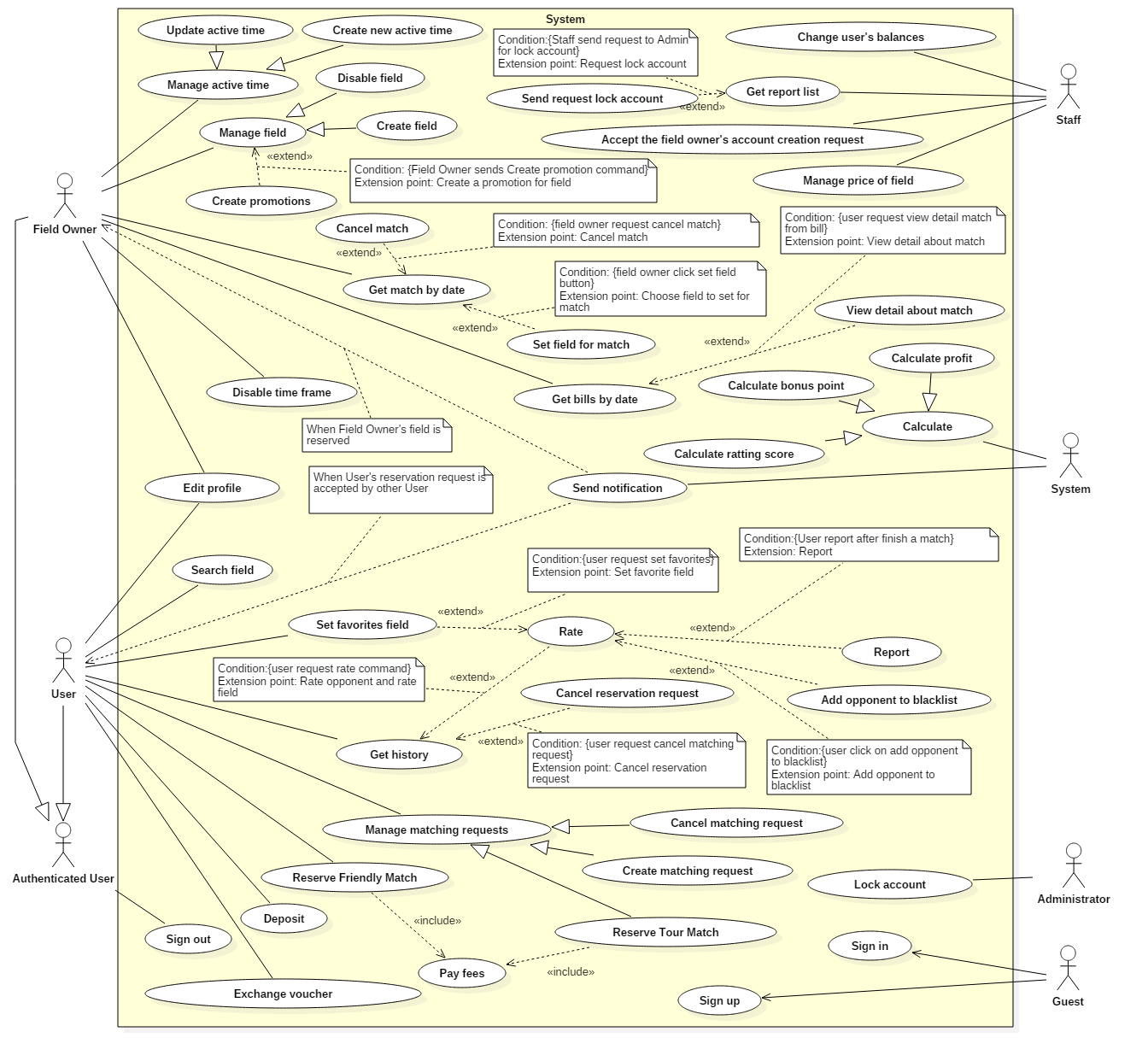
### System Requirement

System performs functions that run underground and connects to external system (deposit money, notification system). System do following functions:

* Calculate bonus point (based on rating of user)
* Calculate rating score (based on rating of user)
* Send notification (when user reserves field, found suitable opponents…; field owner has reservation request)

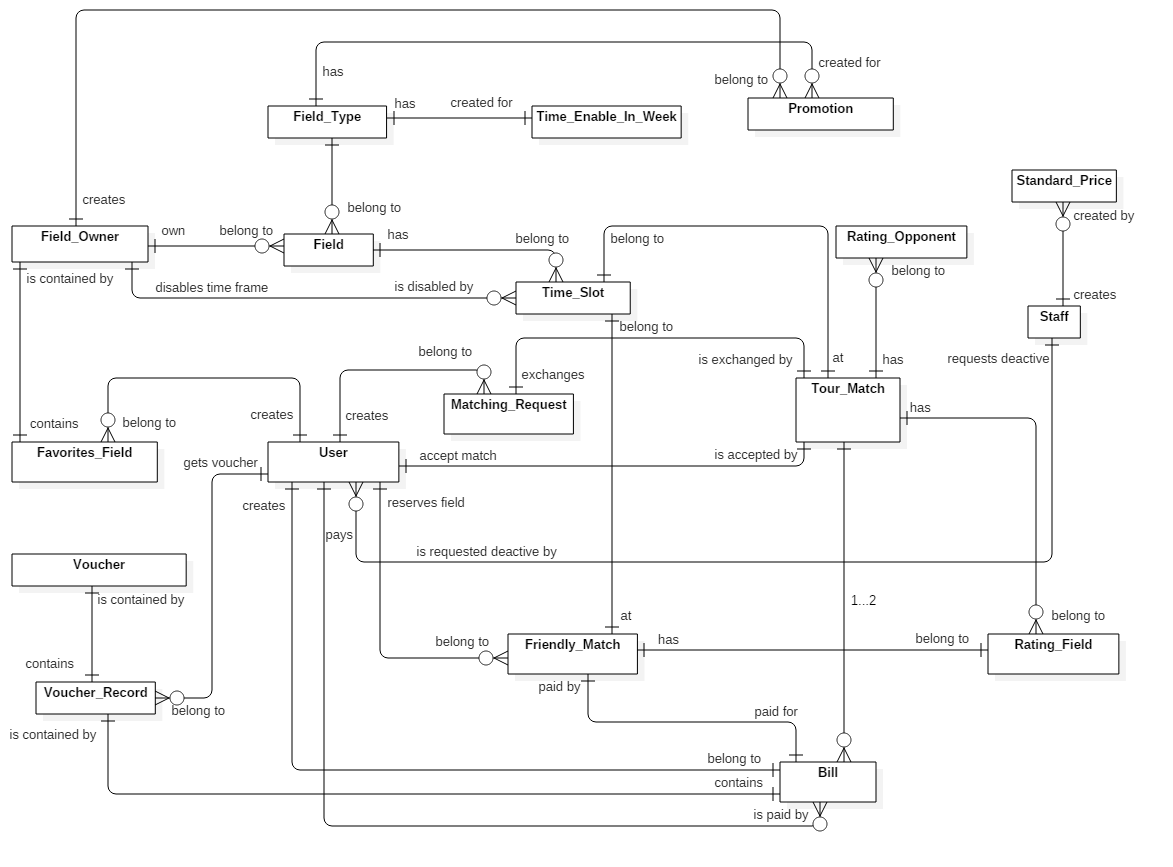
## System Requirement Specification

### System Overview Use Case



*Figure 2: Use case overview*

## Conceptual Diagram

n

*Figure 3: Conceptual Diagram*

**Data Dictionary**

|  |  |
| --- | --- |
| **Entity Data dictionary: describe content of all entities** | |
| **Entity Name** | **Description** |
| Staff | Contain the staff information |
| Standard Price | Contain standard price for field in system |
| Field Owner | Contain the field owner information |
| Field | Contain the field information |
| Field Type | Contain the field type information |
| Time Enable In Week | Contain field’s active time information of field owner |
| Promotion | Contain promotion information |
| User | Contain the user information |
| Time Slot | Contain time slot information (start time, end time, price…) |
| Matching Request | Contain matching request information of user |
| Friendly Match | Contain information about match that reserved by a team |
| Tour Match | Contain information about match that reserved by 2 team |
| Voucher | Contain information about voucher of system |
| Voucher Record | Contain information about voucher get by user |
| Bill | Contain information about reservation field fees |
| Rating Field | Contain information about rating field |
| Rating Opponent | Contain information about rating opponent |

*Table 5: Conceptual Diagram Data Dictionary*

# Software Design Description

## Design Overview

This document describes the technical and user interface design of FFRS System. It includes the architectural design, the detailed design of common functions and business functions and the design of database model.

* The architectural design describes the overall architecture of the system and the architecture of each main component and subsystem.
* The detailed design describes static and dynamic structure for each component and functions. It includes class diagram, class explanations and sequence diagrams for each use cases.
* The database design describes the relationships between entities and details of each entity.
* Document overview:
  + Section 2: gives an overall description of the system architecture design.
  + Section 3: gives component diagrams that describes the connection and integration of the system.
  + Section 4: gives the detail design description which includes class diagram, class explanation and sequence diagram to details the application functions.
  + Section 5: describes screen design.
  + Section 6: describes a fully attributed ERD.
  + Section 7: describes algorithms.

## System Architectural Design

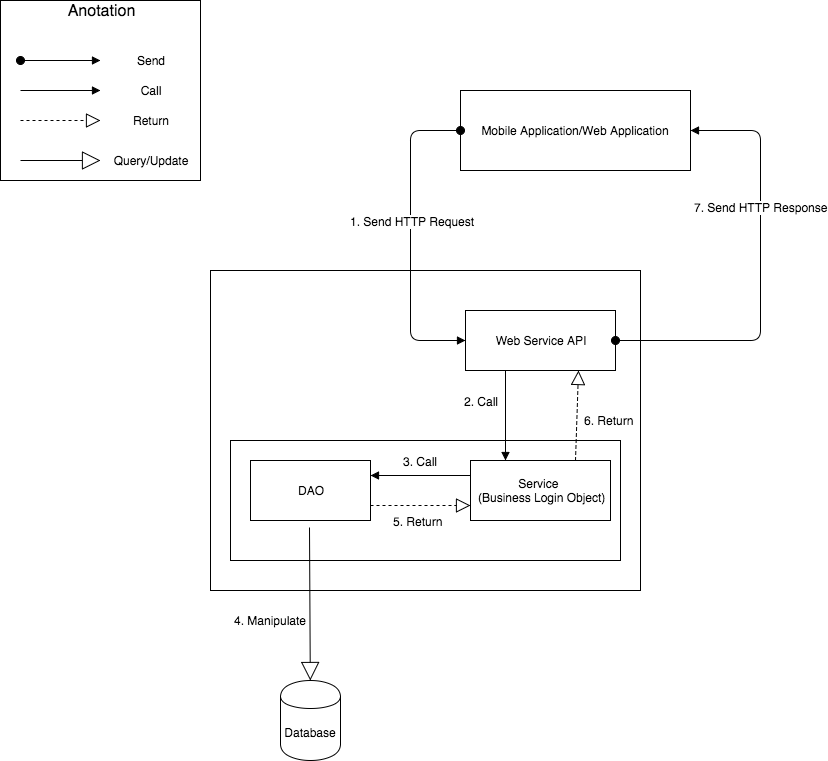
### System Architecture Design

We choose Restful Web Service model as main model because of following advantages:

* + In the future, our application may expand for other platforms as well. Web service provides reusable API that can be called by any platforms, so that our system could easily expand.
  + Beacon is a newly developed technology; its SDK is changing frequently. When does, we only need to update mobile application without affecting other components.

Web Service is developed from Spring MVC architecture style beacause:

* + It provides comprehensive infrastructure support for developing Java enterprise level applications. Spring also enables the developer to create high performing, easily testable, reusable and loose coupling enterprise Java application.
  + Spring enables the developers to develop the enterprise applications using POJOs (Plain Old Java Object). The benefit of developing the applications using POJO is, that we do not need an enterprise container such as an application server, but we have the option of using a robust servlet container.
  + Spring provides a very clean division between controllers and JavaBean models.



**Figure 1 - System Architecture Design**

### Mobile Application Architecture Design

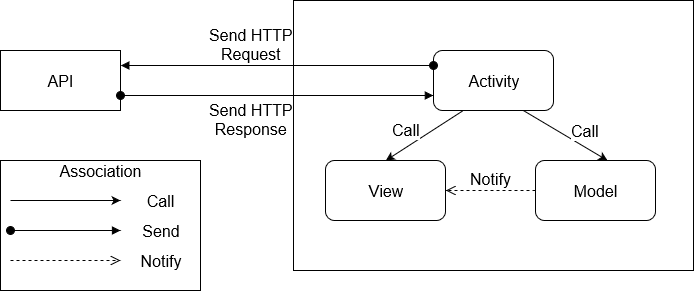
The application is developed as an Android native application. In general, the application architecture conforms to Android architecture.

*Reference: Android Developer Guider – Application Fundamentals*

[*http://developer.android.com/guide/components/fundamentals.html*](http://developer.android.com/guide/components/fundamentals.html)

Our team chooses Android because of following reasons:

* + Android is currently the most common mobile platforms in the world, and it doesn’t have any signs of slowing down. As we want to maximize our potential users, choosing Android will give us the highest popularity market.
  + Developing other mobile platforms (such as iOS or Windows Phone) may require specific expensive devices and tools. On the other hand, Android development can be done on either Mac, Windows or Linux devices with free tools.
  + Native Android applications are developed using Java programing language. Our team is strong at Java and chooses it as the main programing language.



**Figure 1 – Mobile Application Architecture Design**

### Web Application Architecture Design

* + In general, use Redux when you have reasonable amounts of data changing over time, you need a single source of truth, and you find that approaches like keeping everything in a top-level React component's state are no longer sufficient.
* However, it's also important to understand that using Redux comes with tradeoffs. It's not designed to be the shortest or fastest way to write code. It's intended to help answer the question "When did a certain slice of state change, and where did the data come from?", with predictable behavior. It does so by asking you to follow specific constraints in your application: store your application's state as plain data, describe changes as plain objects, and handle those changes with pure functions that apply updates immutably. This is often the source of complaints about "boilerplate". These constraints require effort on the part of a developer, but also open up a number of additional possibilities (such as store persistence and synchronization).
  + In the end, Redux is just a tool. It's a great tool, and there's some great reasons to use it, but there's also reasons you might not want to use it. Make informed decisions about your tools, and understand the tradeoffs involved in each decision.

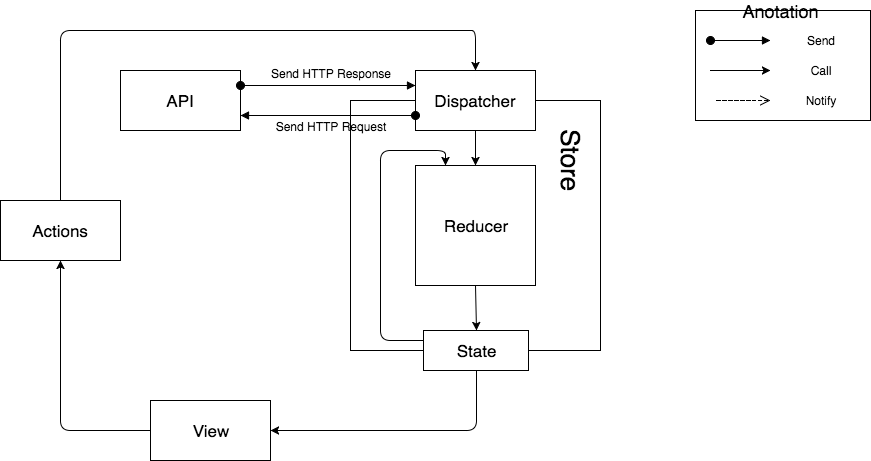
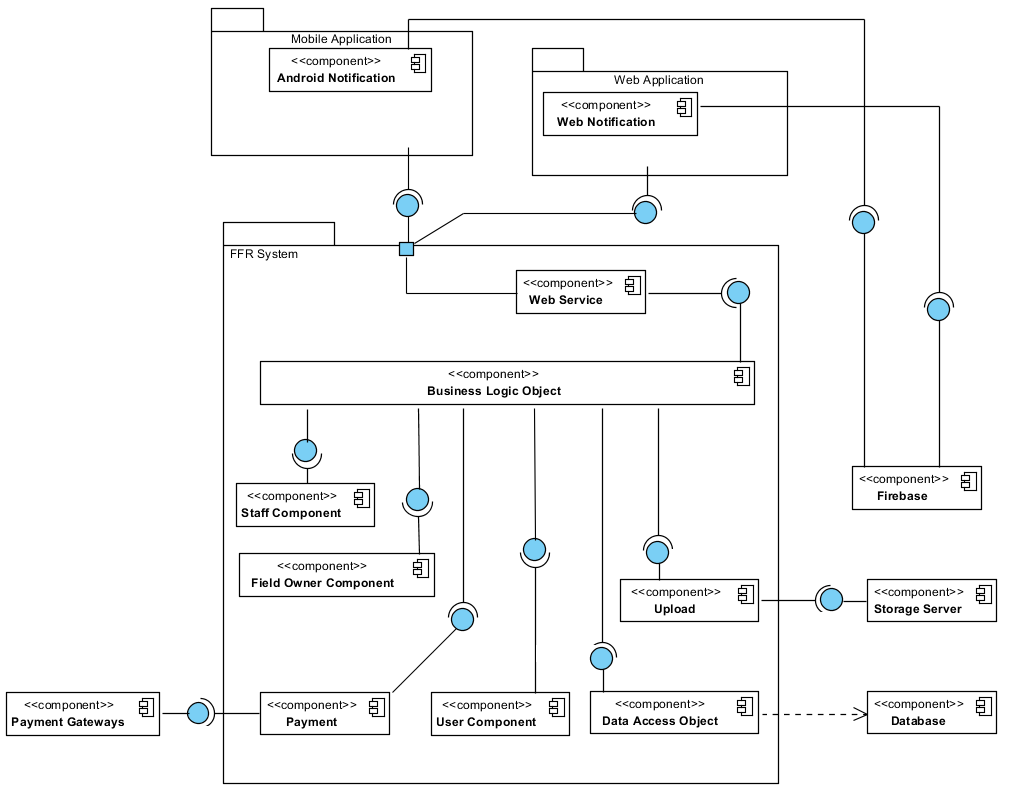


Figure 2 – Web Application Architecture Design

## Component Diagram

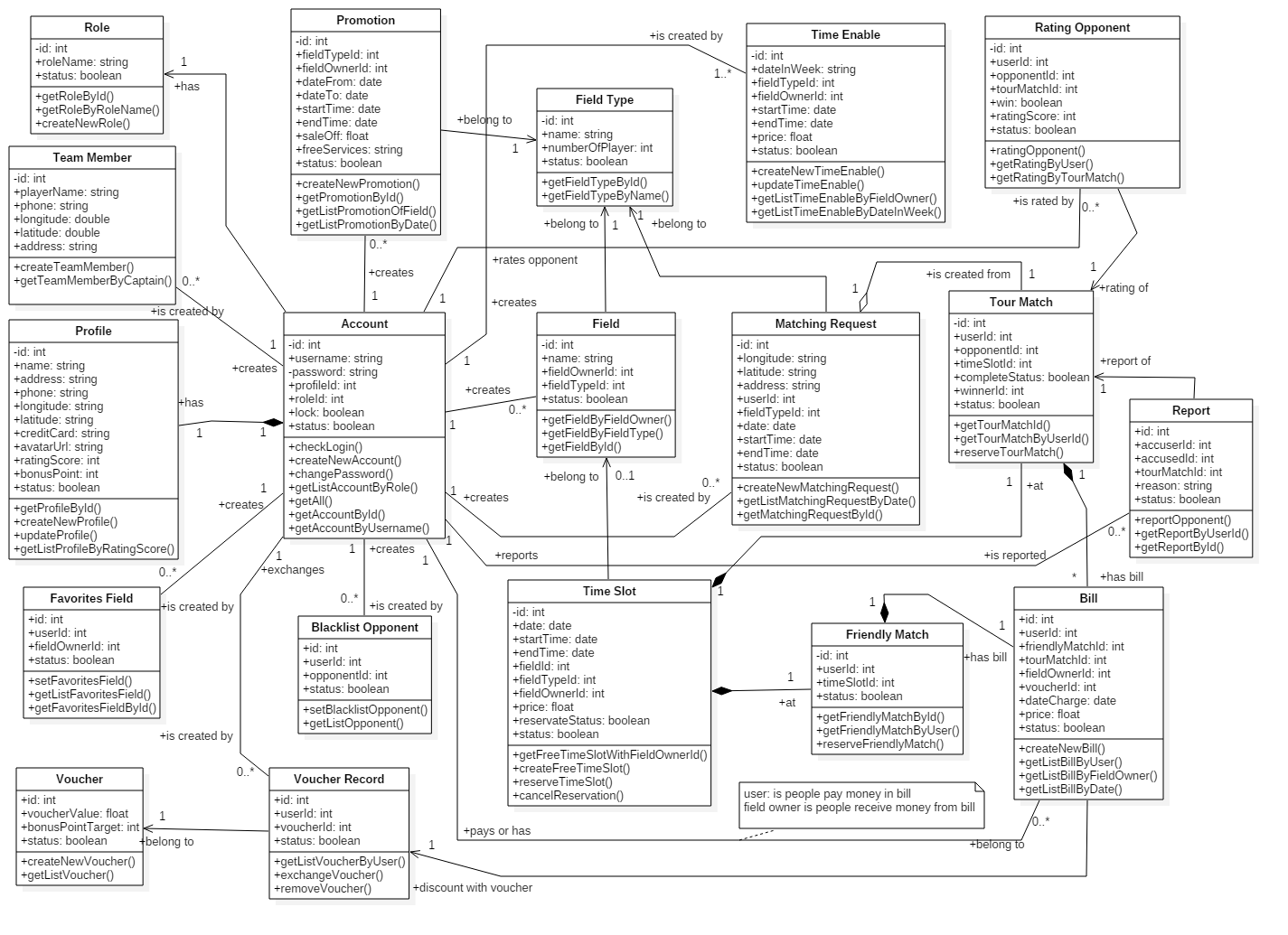


**Figure 10: Component Diagram**

|  |  |
| --- | --- |
| Component Dictionary: Describes components | |
| Business Logic Object | Common objects handle domain business operations for each component. |
| User Component | Component handles user activities in the system. |
| Field Owner Component | Component handles field owner activities in the system. |
| Staff Component | Component handles staff activities in the system. |
| Web Service | Provide API for mobile application and web application to interact with the system. |
| Upload | Component handles upload file to storage server. |
| Data Access Object | Component interacts between the system and database. |
| Payment | Component to interacts between the system and payment gateways. |
| FFR System | FFR System package: back-end system. |
| Payment Gateways | Payment Gateways: Paypal, Nganluong. |
| Database | Component represents for MySQL Database. |
| Storage Server | Component saves information: image avatar. |
| Firebase | Component supports notification between Android Application and Web Application. |
| Android Application | Android Application package. |
| Web Application | Web Application package: View, Controller. |
| Android Notification | Component supports android application to receive notifications. |
| Web Notification | Component supports web application to receive notifications. |

## Detailed Description

### Class Diagram



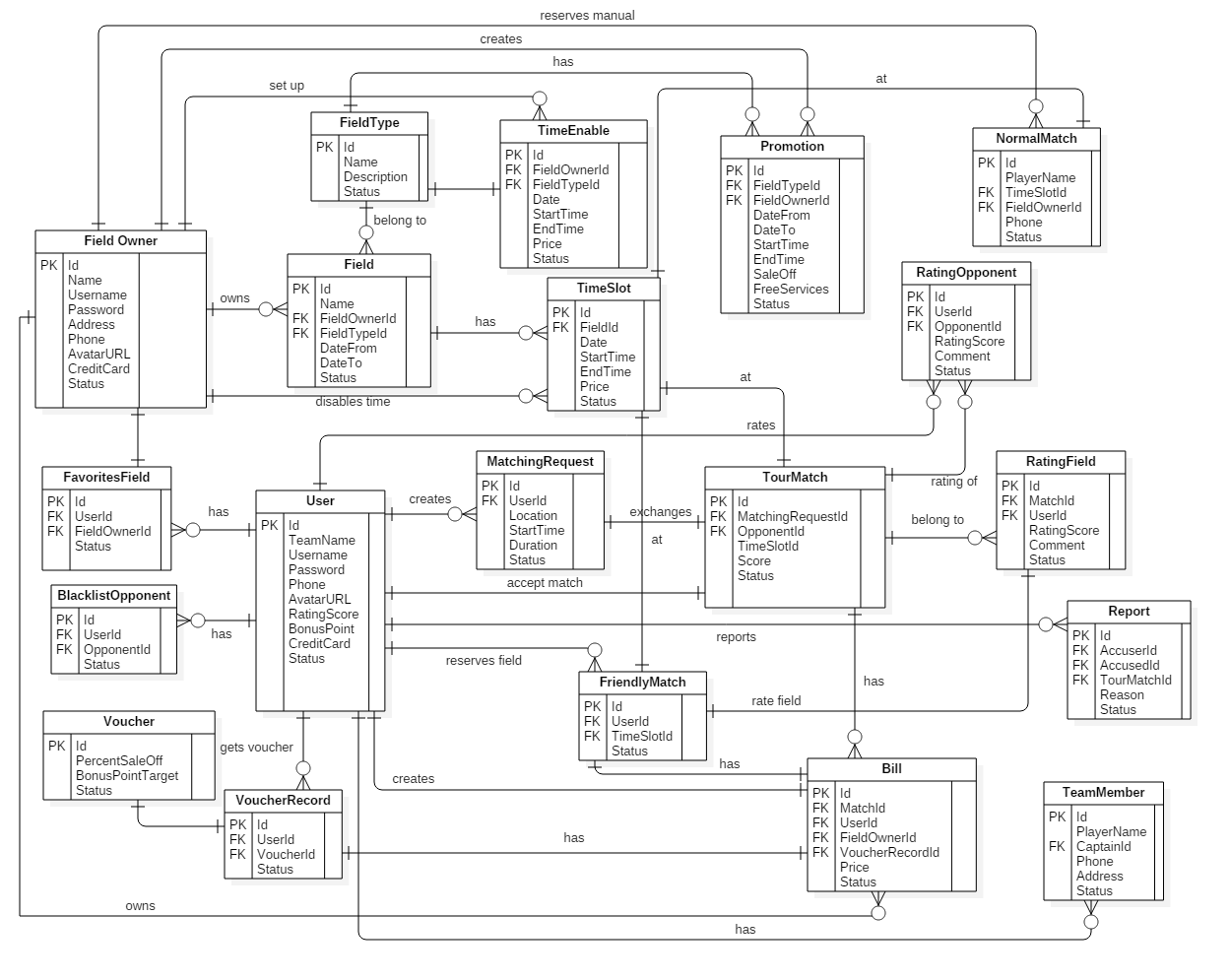
*Figure 7: Class Diagram*

|  |  |  |
| --- | --- | --- |
| **Class name** | **Mapping column with conceptual diagram** | **Description** |
| Role Entity | Role | Contains information of role in the system: field owner, user, staff… |
| Account Entity | Account | Contains information about username, password, status of account. |
| Profile Entity | Profile | Contains information detail of user in system. |
| Field Entity | Field | Contains information about field of field owner. |
| Field Type Entity | Field Type | Contains number of players, field type such as: 5 vs 5, 7 vs 7… |
| Matching Request Entity | Matching Request | Contains information about mataching request of user. |
| Tour Match Entity | Tour Match | Contains information about tour match (match is created by matching opponent). |
| Friendly Match Entity | Friendly Match | Contains information about match. |
| Time Slot Entity | Time Slot | Contains information of all time slot in the system. |
| Rating Opponent Entity | Rating Opponent | Contains information about rating opponent. |
| Bill Entity | Bill | Contains information about payment reservation of request. |
| Promotion Entity | Promotion | Contains information about promotion is created by field owner. |
| Favorites Field Entity | Favorites Field | Contains information about favorites field of user. |
| Blacklist Opponent Entity | Blacklist Opponent | Contains information about blacklist opponent. |
| Team Member Entity | Team Member | Contains information about team member of user |
| Voucher Entity | Voucher | Contains information about discount of voucher |
| Voucher Record Entity | Voucher Record | Contains information about voucher record is exchanged by user. |
|  | | |

*Table 7: Class Diagram Dictionary*

## Database Design

### Entity Relationship Diagram (ERD)



*Figure 8: Entity Relationship Diagram*

### Data Dictionary

|  |  |  |
| --- | --- | --- |
| Entity Data Dictionary: describe content of all entities | | |
| Entity name | **Description** | **Mapping column with Conceptual diagram** |
| User | Contain the user information | user |
| Field Owner | Contain the field owner information | fieldOwner |
| Field | Contain the field information | field |
| FavoritesField | Contain the favorite field information | favoriteField |
| FieldType | Contain the field type information | fieldType |
| TimeEnable | Contain the time enable information | timeEnable |
| TimeSlot | Contain the time slot information | timeSlot |
| FriendlyMatch | Contain the friendly match information | friendlyMatch |
| TourMatch | Contain the tour match information | tourMatch |
| NormalMatch | Contain the normal match information | normalMatch |
| MatchingRequest | Contain the matching request information | matchingRequest |
| RatingField | Contain the rating field information | ratingField |
| RatingOpponent | Contain the rating opponent information | ratingOpponent |
| Report | Contain the report information | report |
| BlacklistOpponent | Contain the blacklist opponent information | blacklistOpponent |
| Bill | Contain the bill information | bill |
| Promotion | Contain the promotion information | promotion |
| Voucher | Contain the voucher information | voucher |
| VoucherRecord | Contain the voucher record information | voucherRecord |
| TeamMember | Contain the team member information | teamMember |

*Table 8: ERD dictionary*

## Algorithms

### Manage time

#### Definition

The field management system is based on free time of the field, so that the System can quickly know which field owner can satisfy the user’s request.

#### Define problem

The User may select the time frame arbitrarily and then reserves the field. The problem is how system can manage time of fields. System has to calculate free time quickly and effectively.

When a large number of users reserve field, the field’s timeline may be broken into fragments. Therefore, system has to choose field owners who received matching requests.

#### Solution

The field management system is based on free time of the field, so that the system can decide which field owner can satisfy the user’s request. If H (0, ..., k, ..., 23) is the set of vacant fields for one hour from k (H (k), the number of vacant fields between k and k + 1 hour).

Let F(1, .., i, n) is the set of fields of a field owner. User A reserve field from x to y hour(s).

Step 1: The System will check H (k) with x <= k <= y-1 with all H (k)> 0, there exists at least 1 vacant field from x to y.

Step 2: Suppose F(i) is the field that Field owner choose for user A.

For (j = x, j = j +1, j<=y)

If (F(i) is free at j hour) then

User can reserve this field as normal till j hour

If (F(i) is reserved by other user) then

1. Field owner find other field F(ii) which has free time at j hour
2. Swap all reserve request of F(i) and F(ii) from j hour
3. After swap F(i) is free at j hour

#### Example

Two fields of field owner “Cây Trâm”:

+ Field 1: 06AM to 22PM

+ Field 2: 06AM to 22PM

Reserve request:

+ User A reserved Field 1, time frame is: 06AM to 08AM

+ User B reserved Field 2, time frame is: 08AM to 10AM

New reserve request:

+User C reserves time frame 07AM to 09AM

**Field owner set field for Use**r:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field** | **06 - 07** | **07 - 08** | **08 - 09** | **09 - 10** | **10 - 11** | **11 - 12** |
| Field 1 | User A | User A | 0 | 0 | 0 | 0 |
| Field 2 | 0 | 0 | User B | User B | 0 | 0 |

**After system apply algorithm:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field** | **06 - 07** | **07 - 08** | **08 - 09** | **09 - 10** | **10 - 11** | **11 - 12** |
| Field 1 | User A | User A | User B | User B | 0 | 0 |
| Field 2 | 0 | User C | User C | 0 | 0 | 0 |

#### Complexity

The complexity is: O(n2)

### Distance

#### Definition

The situation is: when system finds fields around 1 location A (latitude1, longitude1), with k(km) radius.

If location B (latitude2, longtitude2) is within a radius of k(km) from A, the precondition is:

x, y are constants.

The main purpose is we calculate the approximately x, y then input to the system. Based on x, y, system can quickly find out which fields is the most suitable. X, y constants help system early removal fields can not afford.

#### Define problem

Calculating the distance between 2 points is one of the most used functions in our system (user reserves field, system finds opponent, system matches 2 users).

Calculating the distance between 2 points takes so many steps.

#### Solution

We calculate x, y as 2 constants. System based on x, y can calculate faster.

For any two points on a sphere, the haversine of the central angle between them is given by:

hav is the Havesin fuction:

* *d* is the distance between the two points
* *r* is the radius of the sphere,
* *φ*1, *φ*2: latitude of point 1 and latitude of point 2, in radians
* *λ*1, *λ*2: longitude of point 1 and longitude of point 2, in radians

Note that φ and λ can be converted from radians to degrees by multiplying by  as usual.

A (latitude1, longitude1), B (latitude2, longtitude2)

The latitude of Ho Chi Minh City, Vietnam is 10.762622, and the longitude is 106.660172.

* Let latitude1 = latitude2 = 10.76, d=5(km), r= 6.367.449(m)
  + - longitude2 = longitude1±0.0455 (in decimal)
* Let longitude1 = longitude2, d=5(km), r=6.367.449(m)
  + - latitude2 = latitude1±0.045 (in decimal)

After testing, the erroneous is less than 5%.

#### Example

A(10.8547079,106.6257702) (Google maps define as Công viên phần mềm quang trung)

B (latitude, longitude) has longitude the same as A, latitude as A+0.0455

B(10.9002079,106.6257702)

The distance between A and B is 5.04km (using google maps calculator)

C (latitude, longitude) has latitude the same as A, longitude as A+0.045

C (10.8547079, 106.6707702)

The distance between A and C is 4.98km (using google maps calculator)

#### Complexity

The complexity is: O(n)

### Matching 2 users

#### Definition

After 2 users create matching request, if 2 requests are matched, system will suggest list of fields for user.

#### Define problem

Usually, 2 requests will be matched by comparing time. However, just comparing time is not enough. We don’t know if they are as the same level (for example, a team is skilled and the other is not so good at playing football).

There are many other reasons why we have to build different criterions.

#### Solution

The system not only match users based on time but also rely on the user's rank, distance between two users and the list of their favorite fields to make a list of fields suitable for both.

Set favorite field and blacklist functions make the matching system more accurate and preferable to users. After completing a match, two users will rate the match, which enables the system to calculate 2 users’ rank more accurately.

The favorite field has high priority on the suitable list of fields. If a user is in the blacklist of other user, they can not match each other.

# Tasks sheet

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Product Deliverables** | **Task** | **QuyMM** | **ThanhTH** | **HuanPM** | **HieuPT** | **Unit** | **Size** |
| 1 | Report1 - Introduction | Project Information | **O** |  |  |  |  | 1 |
| Introduction |  |  |  | **O** |  | 1 |
| Current Situation |  |  | **O** | **O** |  | 1 |
| Problem Definition | **O** |  |  |  |  | 1 |
| Proposed Solution |  | **O** |  |  |  | 1 |
| Functional Requirements | **O** | **O** |  |  |  | 1 |
| Role and Responsibility | **O** |  |  |  |  | 1 |
| Review and merge document | **O** |  |  |  |  | 1 |
| 2 | Report2- Software Project Management Plan | Problem Definition |  |  | **O** |  |  | 1 |
| Project Organization | **O** |  |  |  |  | 1 |
| Project Management Plan | **O** | **O** |  |  |  | 1 |
| Coding Convention |  |  |  | **O** |  | 1 |
| Review and merge document | **O** |  |  |  |  | 1 |
| 3 | Report 3- Software Requirement Specification | **User Requirement Specification** |  |  | **O** |  |  | 1 |
| **System Requirement Specification** |  |  |  |  |  |  |
| **External Interface Requirements** |  | **O** |  |  |  | 1 |
| **Functional Requirement** | **O** |  |  |  |  | 1 |
| **System Overview Usecase** | **O** |  |  |  |  | 3 |
| **List of Usecase** |  |  |  |  |  |  |
| **Admin** |  |  |  |  |  |  |
| Add Staff |  | **O** |  |  |  | 1 |
| Delete Staff |  | **O** |  |  |  | 1 |
| Lock Account |  |  |  | **O** |  | 1 |
| **Guest** |  |  |  |  |  |  |
| Sign Up |  | **O** |  |  |  | 1 |
| Sign In |  | **O** | **O** | **O** |  | 1 |
| **Field Owner** |  |  |  |  |  |  |
| Get match by date |  | **O** |  |  |  | 2 |
| Create promotion |  |  |  | **O** |  | 1 |
| Create field |  | **O** |  |  |  | 1 |
| Disable field |  | **O** |  |  |  | 2 |
| Create active time |  | **O** |  |  |  | 1 |
| Update active time |  | **O** |  |  |  | 2 |
| Disable time frame | **O** | **O** |  |  |  | 2 |
| Set field for match |  |  |  | **O** |  | 2 |
| Get bills by date |  |  |  | **O** |  | 1 |
| Edit Profile |  |  |  | **O** |  | 1 |
| **User** |  |  |  |  |  | 1 |
| Edit Profile |  |  | **O** |  |  | 1 |
| Search field | **O** |  | **O** |  |  | 1 |
| Reserve friendly match | **O** |  | **O** |  |  | 3 |
| Reserve tour match | **O** |  | **O** |  |  | 4 |
| Get history list | **O** |  |  |  |  | 1 |
| Set favorites field |  |  | **O** |  |  | 1 |
| Rate field |  |  | **O** |  |  | 1 |
| Rate opponent |  |  | **O** |  |  | 1 |
| Report account |  |  | **O** |  |  | 1 |
| Add opponent to blacklist |  |  | **O** |  |  | 1 |
| Exchange voucher | **O** |  | **O** |  |  | 1 |
| Create matching request | **O** |  |  |  |  | 1 |
| Cancel matching request | **O** |  |  |  |  | 1 |
| Cancel reservation request |  |  | **O** |  |  | 2 |
| Deposit |  |  | **O** |  |  | 3 |
| **Staff** |  |  |  |  |  |  |
| Get report list |  |  |  | **O** |  | 1 |
| Send lock account request |  |  |  | **O** |  | 1 |
| Accept the field owner's account creation request |  | **O** |  |  |  | 1 |
| Manage price of field |  | **O** |  |  |  | 1 |
| Change user's balance |  | **O** |  |  |  | 1 |
| **System** |  |  |  |  |  |  |
| Calculate bonus point | **O** |  |  |  |  | 2 |
| Calculate rating score | **O** |  |  |  |  | 2 |
| Send notification | **O** | **O** | **O** |  |  | 4 |
| **Software System Attribute** |  |  |  |  |  |  |
| Usability |  |  |  | **O** |  | 1 |
| Reliability |  |  |  | **O** |  | 1 |
| Availability |  | **O** |  |  |  | 1 |
| Secutiry |  | **O** |  |  |  | 1 |
| Maintainability |  |  | **O** |  |  | 1 |
| Portability |  |  |  | **O** |  | 1 |
| Performance |  |  |  | **O** |  | 1 |
| **Conceptual Diagram** | **O** |  |  |  |  | 4 |
| Review and merge document | **O** |  |  |  |  | 1 |
| 4 | Report 4- Software Design Description | Design Overview | **O** |  |  |  |  | 1 |
| System Architectural Design | **O** |  |  |  |  | 4 |
| Component Diagram | **O** |  |  |  |  | 4 |
| Class Diagram | **O** |  | **O** |  |  | 4 |
| Class Diagram Dictionary |  |  | **O** |  |  | 1 |
| Class Diagram Explanation |  |  | **O** |  |  | 1 |
| **Interaction Diagram** |  |  |  |  |  |  |
| **Activity Diagram** |  |  |  |  |  |  |
| Login |  |  |  | **O** |  | 1 |
| Register User |  |  |  | **O** |  | 1 |
| Create matching request |  |  | **O** |  |  | 1 |
| Reserve friendly match |  |  | **O** |  |  | 1 |
| Reserve tour match |  |  | **O** |  |  | 1 |
| Deposit |  |  |  | **O** |  | 1 |
| **Sequence Diagram** |  |  |  |  |  |  |
| Register Field Owner |  |  |  | **O** |  | 1 |
| Get match by date |  | **O** |  |  |  | 1 |
| Create field |  | **O** |  |  |  | 1 |
| Create active time |  | **O** |  |  |  |  |
| Disable time frame |  | **O** |  |  |  |  |
| Get bills by date |  | **O** |  |  |  |  |
| Set field for match |  | **O** |  |  |  | 1 |
| **Interface** |  |  |  |  |  |  |
| Component Interface | **O** |  |  |  |  | 1 |
| User Interface Design | **O** | **O** | **O** | **O** |  | 1 |
| **Database Design** |  |  |  |  |  |  |
| Entity Relationship Diagram | **O** |  |  |  |  | 4 |
| Data dictionary |  |  |  |  |  | 1 |
| **Algorithms** |  |  |  |  |  |  |
| Manage time | **O** |  |  | **O** |  |  |
| Calculate distance | **O** |  |  | **O** |  |  |
| Matching 2 user | **O** |  |  |  |  | 5 |
| Review and merge document | **O** |  |  |  |  | 1 |
| 5 | Report 5 - Software Implementation and Test Document | **Introduction** |  |  |  |  |  |  |
| Overview | **O** |  | **O** |  |  | 1 |
| Test Approach |  | **O** |  |  |  | 1 |
| **Database Relationship Diagram** |  |  |  |  |  |  |
| Physical Diagram | **O** | **O** | **O** | **O** |  | 3 |
| Data dictionary |  |  | **O** | **O** |  | 1 |
| **Mobile Implementation** |  |  |  |  |  |  |
| **User** |  |  |  |  |  |  |
| Sign Out |  |  | **O** |  |  | 1 |
| Edit Profile |  |  | **O** |  |  | 1 |
| Search field |  |  | **O** |  |  | 1 |
| Reserve friendly match | **O** |  |  |  |  | 1 |
| Reserve tour match |  |  | **O** |  |  | 1 |
| Get reservation request history |  |  |  |  |  | 1 |
| Get matching request history | **O** |  |  |  |  | 1 |
| Set favorites field |  |  | **O** |  |  | 1 |
| Rate field |  |  | **O** |  |  | 1 |
| Rate opponent |  |  | **O** |  |  | 1 |
| Report account |  |  | **O** |  |  | 1 |
| Add opponent to blacklist |  |  |  | **O** |  | 1 |
| Exchange voucher |  |  | **O** | **O** |  | 1 |
| Create matching request | **O** |  | **O** |  |  | 1 |
| Cancel matching request |  |  |  | **O** |  | 1 |
| Cancel reservation request | **O** |  | **O** |  |  | 1 |
| Deposit |  |  | **O** |  |  | 1 |
| Apply voucher to bill |  |  |  | **O** |  | 1 |
| **Web Application Implementation** |  |  |  |  |  |  |
| **Field Owner** |  |  |  |  |  |  |
| Get match by date |  | **O** |  |  |  | 1 |
| Create promotion |  | **O** |  |  |  | 1 |
| Create field |  | **O** |  |  |  | 1 |
| Disable field | **O** | **O** |  |  |  | 1 |
| Create active time | **O** | **O** |  |  |  | 1 |
| Disable time frame | **O** | **O** |  |  |  | 1 |
| Get bills by date |  | **O** |  |  |  | 1 |
| **Staff** |  |  |  |  |  |  |
| Login |  |  |  | **O** |  |  |
| Logout |  |  |  | **O** |  |  |
| Get report list |  |  |  | **O** |  |  |
| Send lock account request |  |  |  | **O** |  | 1 |
| Accept the field owner's account creation request |  |  |  | **O** |  | 1 |
| Manage price of field |  | **O** |  |  |  |  |
| Change user's balance |  | **O** |  |  |  | 1 |
| **Admin** |  |  |  |  |  |  |
| Create Account |  | **O** |  | **O** |  | 1 |
| Update Account |  |  |  | **O** |  | 1 |
| Lock Account |  |  |  | **O** |  | 1 |
| **API Implementation** |  |  |  |  |  |  |
| **User** |  |  |  |  |  |  |
| Sign Out |  |  | **O** |  |  | 1 |
| Edit Profile |  |  | **O** |  |  | 1 |
| Search field | **O** |  |  |  |  | 1 |
| Reserve friendly match | **O** |  |  |  |  | 2 |
| Reserve tour match | **O** |  |  |  |  | 4 |
| Get reservation request history | **O** |  |  |  |  | 1 |
| Get matching request history | **O** |  |  |  |  | 1 |
| Set favorites field |  |  | **O** |  |  | 1 |
| Rate field |  |  | **O** |  |  | 1 |
| Rate opponent |  |  |  | **O** |  | 1 |
| Report account |  |  |  | **O** |  | 3 |
| Add opponent to blacklist |  |  |  |  |  | 1 |
| Exchange voucher |  |  |  |  |  | 1 |
| Create matching request | **O** |  |  |  |  | 1 |
| Cancel matching request | **O** |  |  | **O** |  | 1 |
| Cancel reservation request | **O** |  |  |  |  | 2 |
| Deposit | **O** |  |  |  |  | 1 |
| Apply voucher to bill |  |  |  | **O** |  | 1 |
| **Field Owner** |  |  |  |  |  |  |
| Get match by date | **O** |  |  |  |  | 1 |
| Create promotion |  | **O** |  |  |  | 1 |
| Create field |  | **O** |  |  |  | 1 |
| Disable field | **O** |  |  |  |  | 1 |
| Create active time | **O** |  |  |  |  | 1 |
| Disable time frame | **O** |  |  |  |  | 1 |
| Get bills by date | **O** |  |  |  |  | 1 |
| **Test plan** |  |  |  |  |  |  |
| Features to be tested |  |  |  | **O** |  | 1 |
| Features not to be tested |  |  |  | **O** |  | 1 |
| **System Testing Test Case** |  |  |  |  |  |  |
| Test Flow |  |  |  | **O** |  | 1 |
| **Test Case** |  |  |  |  |  |  |
| **Mobile application test cases** |  |  |  |  |  |  |
| **Guest** |  |  |  |  |  |  |
| Sign in | **O** |  |  |  |  | 1 |
| Sign up |  |  |  | **O** |  | 1 |
| **User** |  |  |  |  |  |  |
| Search Field | **O** |  |  |  |  | 1 |
| Reserve Field |  |  | **O** |  |  | 1 |
| Create matching request |  |  |  | **O** |  | 1 |
| Reserve tour match |  |  |  | **O** |  | 1 |
| Set favorites field |  |  | **O** |  |  | 1 |
| **Field Owner** |  |  |  |  |  |  |
| Create new active time |  | **O** |  |  |  | 1 |
| Create new field |  | **O** |  |  |  | 1 |
| 6 | Report 6 - Software User's Manual | Review and merge document | **O** |  |  |  |  | 1 |
| Installation Guide |  |  |  | **O** |  | 1 |
| **User’s Guide** |  |  |  |  |  |  |
| ***Mobile Application*** |  |  |  |  |  |  |
| **Guest** |  |  |  |  |  |  |
| Sign Up | **O** |  |  |  |  | 1 |
| **User** |  |  |  |  |  |  |
| Search field |  |  | **O** |  |  | 1 |
| Reserve friendly match |  |  | **O** |  |  | 1 |
| Find opponent |  |  | **O** |  |  | 1 |
| Create new matching request |  |  | **O** |  |  | 1 |
| Choose opponent |  |  | **O** |  |  | 1 |
| Deposit |  |  | **O** |  |  | 1 |
| ***Web Application*** |  |  |  |  |  |  |
| **Field Owner** |  |  |  |  |  |  |
| Update active time |  | **O** |  |  |  | 1 |
| Create new promotion |  | **O** |  |  |  | 1 |
| Reserve match |  | **O** |  |  |  | 1 |
| Review and merge document | **O** |  |  |  |  | 1 |
|  |  |  |  |  |  |  |  |  |
| **Notes** |  |  |  |  |  |  |  |  |
| No. | Function Types | Function Point Size |  |  |  |  |  |  |
| 1 | Insert, Delete, Update, Simple Search, Login, Register, simple function or simple query on one table | 1 |  |  |  |  |  |  |
| 2 | Computed functions or joined query | 2 |  |  |  |  |  |  |
| 3 | Complex functions or multiple subquery | 3 |  |  |  |  |  |  |
| 4 | Constraint process and data integrity | 4 |  |  |  |  |  |  |
| 5 | Functions with implementing algorithms | 5 |  |  |  |  |  |  |

# Appendix

## UML Documentation of IBM

<http://www.ibm.com/developerworks/rational/library/769.html?ca=drs->

## React Native

<https://reactjs.org/>

## Spring Boot Framework

<https://projects.spring.io/spring-boot/>

## Gradle Build Tools

<https://gradle.org/>

## Firebase Realtime Database

<https://firebase.google.com/docs/database/>

## Dictionary API

<https://glosbe.com/>