

---

# SOFTWARE DESIGN DOCUMENT

for  
iSport



Release 1.0

Prepared by

1650940 Jiang Xiaohu  
1650932 Xu Jingnan  
1651058 Wang Yicheng

School of Software Engineering  
Tongji University

December 27, 2019

---

# Table Of Contents

<b>1</b>	<b>Introduction</b>	<b>6</b>
1.1	Purpose . . . . .	6
1.2	Scope . . . . .	6
1.3	Acronyms, Abbreviations and Definitions . . . . .	7
1.4	Overview . . . . .	10
1.5	Reference Material . . . . .	10
<b>2</b>	<b>System Overview</b>	<b>12</b>
2.1	Product Perspective . . . . .	12
2.2	Product Functionality . . . . .	13
2.3	Users and Characteristics . . . . .	15
2.4	Operating Environment . . . . .	15
2.5	Design and Implementation Constraints . . . . .	17
2.6	User Documentation . . . . .	17
2.7	Assumptions and Dependencies . . . . .	17
<b>3</b>	<b>System Architecture</b>	<b>19</b>
3.1	Architectural Design . . . . .	19
3.2	Description of Achitecture Goals . . . . .	19
3.3	Design Rationale . . . . .	20
<b>4</b>	<b>Principal Components</b>	<b>22</b>
4.1	Component:User Info Management . . . . .	22
4.1.1	Component Description . . . . .	22
4.1.2	Responsibilities . . . . .	23
4.1.3	Component Design Detail Description . . . . .	23

<b>5</b>	<b>Class Interfaces</b>	<b>32</b>
5.1	Class Student . . . . .	32
5.1.1	Public Constructor Student . . . . .	32
5.1.2	Public Method GetStudentID . . . . .	32
5.1.3	Public Method GetDurationOfSession . . . . .	32
5.1.4	Public Method IsSessionExpired . . . . .	32
5.1.5	Public Method CreateCourseItem . . . . .	33
5.1.6	Public Static Method CreateCourse . . . . .	33
5.1.7	Public Method GetEnrolledCourses . . . . .	33
5.1.8	Public Method JoinCourse . . . . .	33
5.2	Class Course . . . . .	33
5.2.1	Public Constructor Course . . . . .	33
5.2.2	Public Method GetStudents . . . . .	33
5.2.3	Public Method GetCourseItems . . . . .	34
5.2.4	Public Method GetChat . . . . .	34
5.3	Class CourseItem . . . . .	34
5.3.1	Public Method RemoveCourseItem . . . . .	34
5.3.2	Public Method ModifyCourseItem . . . . .	34
5.4	Class Document . . . . .	34
5.4.1	Public Constructor Document . . . . .	34
5.4.2	Public Method Download . . . . .	35
5.4.3	Public Method GetCourseItemID . . . . .	35
5.5	Class Chat . . . . .	35
5.5.1	Public Constructor Chat . . . . .	35
5.5.2	Public Method GetChatVersion . . . . .	35
5.5.3	Public Method GetChatMessages . . . . .	35
5.5.4	Public Method SendMessage . . . . .	36
5.5.5	Public Method SendDocument . . . . .	36
5.6	Class ChatMessage . . . . .	36
5.6.1	Public Constructor ChatMessage . . . . .	36
5.6.2	Public Enum MessageType . . . . .	36
5.6.3	Public Method GetType . . . . .	37
5.6.4	Public Method GetContent . . . . .	37
5.6.5	Private Method GetText . . . . .	37
5.6.6	Private Method GetURL . . . . .	37
5.6.7	Public Method GetStudent . . . . .	37

5.6.8	Public Method GetTime . . . . .	37
5.6.9	Public Method Delete . . . . .	37
5.6.10	Public Method GetActiveUsers . . . . .	38
5.6.11	Public Method GetAllUsers . . . . .	38
<b>6</b>	<b>Human Interface Design</b>	<b>39</b>
6.1	Mockup of User Interface . . . . .	39

# Revision History

Name	Date	Reason For Changes	Version
Jiang Xiaohu	2019.11.12	Finish Introduction Part	v1.0
Wang Yicheng	2019.11.14	Finish Overview Part	v1.1
Xu Jingnan	2019.11.16	Finish External Requirements Part	v1.2
Xu Jingnan, Jiang Xiaohu, Wang Yicheng	2019.11.18	Finish Sequence Diagrams for Function Modeling Part	v1.3
Xu Jingnan	2019.11.20	Finish Function Modeling Part	v1.4
Jiang Xiaohu	2019.11.21	Finish Data Modeling Part	v1.5
Wang Yicheng	2019.11.22	Finish Behavior Requirements Part	v1.6
Xu Jingnan	2019.11.23	Finish Nonfunctional Requirements Part	v1.7
Jiang Xiaohu	2019.11.25	Finish Data Dictionary Requirements Part	v1.8

# **1 Introduction**

## **1.1 Purpose**

This design document describes the overall structure of iSport by outlining significant aspects of the systems architecture.

The purpose of this document is to present a detailed description of iSport. It shows how the software system will be structured to satisfy the requirements.

## **1.2 Scope**

This software system will be a web based system for sports fans, professional athletes, and patients who are under recovering training.

This system will be designed to maximize the exercising efficiency by providing tools to assist in checking and correcting user's wrong postures and recommending training courses customized for users, which would otherwise have to be expensive, time-consuming and labor intensive. By maximizing the users training efficiency and convenience the system will meet the needs of sports fans, athletes and injured patients while remaining easy to understand and use.

More specifically, this system is designed to allow a user to imitate the standard exercising postures while observe and correct their mistakes simultaneously with the help of a website.

The software will collect some professional courses in the database, including static and dynamic trainings which means doing exercise according to

a set of images or a video and iSport will recommend suitable trainings for users on the basis of their training performance. Courses are classified into exercising courses and recovering courses, aiming to help athletes and patients respectively.

Both visual and audio notification are used in every course of the system to provide eye-catching, user-friendly and clear instructions; the feedback of one's training is proposed once the training is over and the report can be browsed in the report page.

The selection and deletion of one user's favorable course is supported in personal information webpage and one can comment training he/she has taken on comment webpage to provide suggestions to other users.

The personal information registering, changes is allowed via the application options. The system also contains a relational database containing a list of users, training images and videos.

## 1.3 Acronyms, Abbreviations and Definitions

Table 1.1: Definitions

<b>Definitions, Acronyms, and Abbreviations</b>	
Term	Definitions
User	Someone who interacts with iSport including sports fans, athletes and injured patients who need recovery training.
Sports fans	One of iSport's potential customers who love sports and want to get professional instructions when exercising. Some of them may can't afford the expense of personal coaching or don't have time to go to the gym.
<b>Continued . . .</b>	

Table 1.1: (continued)

<b>Definitions, Acronyms, and Abbreviations (continued)</b>	
Athletes	One of iSport’s potential customers who want to get real-time exercising feedback to improve their performance or who want to get some relaxing training in their spare time to keep a good competitive state.
Injured Patients	One of iSport’s potential customers who need recovering training after some treatments, e.g. surgeries. On the one hand, some of them may can’t afford the doctor’s expensive medical instructions for recovering training. on the other hand, there is no enough doctors or nurses who can instruct and supervise the patients’ recovering exercising. But without professional instructions training can be useless or even leads to secondary trauma.
Admin/Administrator	System administrator who is given specific permission for managing and controlling the system, e.g. updating the user’s information, uploading new training courses.
User Info	User’s basic information including user’s avatar, account name, tel-number and email address.
Courses	Training courses including normal exercising training and recovering training.
Exercise Courses	Training courses which serve the sports fans and athletes.
Recover Courses	Training courses which serve the injured patients.
Static Courses	Training courses which instruct the users photo by photo.
Dynamic Courses	Training courses which instruct the users according to a standard video.
<b>Continued . . .</b>	



Table 1.1: (continued)

<b>Definitions, Acronyms, and Abbreviations (continued)</b>	
Appraisal Subsystem	Remark the user's performance by using a grade from 0 -100
Comment Subsystem	User comment on the training courses they have taken to provide reference for other users.
Recommendation Subsystem	A subsystem which will provide some courses for users according to their recent performance.
Exercise Tips	There will be sports tips in the webpage of iSport to prevent users from athletic injuries.
Sport Report	A web page to feedback the user's exercising performance.
Audio Notification	An audio notification will be shown when the user is doing exercise to encourage the user to hold on or notify the user to correct their postures.
Visual Notification	A visual notification will be shown when the user is doing exercise, if the user's posture is standard, then the web-frame will turn green to suggest the user to hold on, otherwise the web-frame will be red.
DataBase	A relational database containing a list of user info, training images and videos.
Detection Subsystem	Subsystem to detect the user's postures and draw the user's skeleton. The main model of detection subsystem is PoseNet.
Comparison Subsystem	Subsystem to compare the postures of the user and that of the standard. The subsystem aims to check if the user pass the posture.
Correction Subsystem	Subsystem to calculate where the postures' wrong part are, e.g. left-arm, right-leg, head.
<b>Continued . . .</b>	

Table 1.1: (continued)

<b>Definitions, Acronyms, and Abbreviations (continued)</b>	
Clients	Group who delegate the development of iSport to the developers and will take charge of the later management of iSport.
Developers	Develop team including project managers, programmers, testers who are responsible for the development of iSport and its later maintenance and updating.
<b>The End</b>	

## 1.4 Overview

This document is an overview of the software architecture of iSport in high detail. We start by providing all the principal components that the application is built on as well as their responsibilities to the success of the application. Next, we provide diagrams to show the hierarchy of our classes and the architectural design. Finally, we develop a general API of the major class methods used to build the functionality of our application. There are also mockups of our UI design included in our design document.

## 1.5 Reference Material

### Standard References

The standards we have followed are as follows:

- [1] T. Russell, A. Brizee, E. Angeli, and R. Keck, Mla formatting and style guide, The Purdue OWL, 2010.
- [2] Barnard, H Jack and Metz, Robert F and Price, Arthur L et al., A recommended practice for describing software designs: IEEE standards project 1016, 1986.

[3] R. S. Pressman, Software engineering: a practitioners approach. Palgrave Macmillan, 2005.

## **Writing Tools References**

The writing tools we have used are as follows:

[4] L. Lamport, LATEX: a document preparation system: users guide and reference manual. Addison-wesley, 1994.

[5] S. Wong, Staruml tutorial, Connexions Web site, Sep, 2007.

[6] P. O. Team, Process on tools, <https://www.processon.com/support>.

## 2 System Overview

### 2.1 Product Perspective

ISport is a web system developed by isport team, aiming at posture correcting with the assist of camera.

The user-case diagram in the following figure illustrates the user-case in the system. The system is expected to evolve over at least three releases, ultimately allowing for complete streamlining of the posture correcting process, fitness classes and rehabilitation classes for learning.

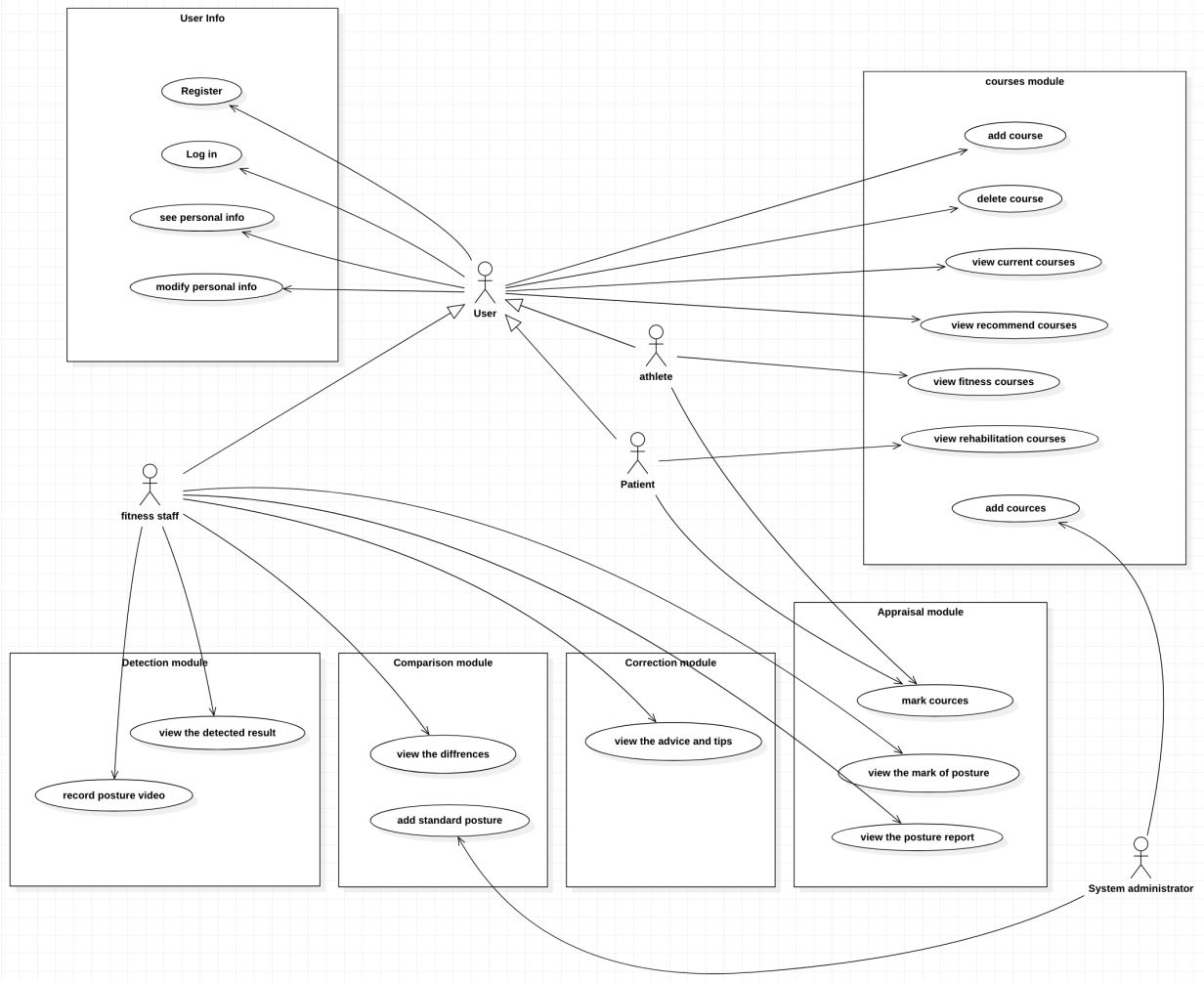


Figure 2.1: overall user-case diagram

## 2.2 Product Functionality

ISport contains these following key features:

Let the user register their account of the website through their mobile phone or e-mail.

Let the user log in to the system by correctly complete the log-in form. If the username is not in the database, the user will be prompted to register his/her account.

User can modify his/her account information after logging in, including the username, e-mail address, phone number, gender, age and so on.

Show all the sorted fitness courses in a list for user to choose in the fitness courses view page.

Show a list of courses that the user has selected to learn before

Show a list of recommended courses for the user by user's previous behavior

Show the comment and evaluation of each courses in their detailed page

Provide the add button for user to add a course he/she want to learn in the future and add the course to his/her courses list.

Provide the delete button for user to remove a course he/she doesn't want to learn and remove the course from his/her courses list.

Generate user's exercise report for watching and analyzing in the report page

Give the user a space to leave his/her evaluation about the courses learnt before

Capture and Collect the posture data of the user in front of the computer and store the data in the server.

Compare user's posture with the standard one and calculate the similarity between the postures.

Show tips on the user's screen to notice the wrong posture of the user and help to correct them.

Rate the user's posture on a scale of zero to ten to let the user know whether his posture is standard or not

System administrator can upload new courses to the system for users to choose and learn.

## 2.3 Users and Characteristics

Users	Desc
People who want to correct their posture	Normal user is expected to register and log in the system, upload their posture and get the feedback advice from the system
People who want to be fitness	People who want to be fitness is expected to register and log in the system, choose fitness classes for themselves and learn it on the website
People who need rehabilitation	People who need rehabilitation is expected to register and log in the system, choose fitness classes for themselves and learn it on the website
System Administrator	System Administrator has the privilege to update posture information in the database. The Administrator does not directly interact with the website

## 2.4 Operating Environment

### Hardware

#### 1. Server

CPU: Intel Dual-Core 2.4GHz

Memory: 8GB

External Storage: 500G SSD

Quantity: 1

## 2. Client(Minimum Configuration)

CPU: Single-Core 1GHz

Memory: 2G

External Storage: 20GB HDD

## Software

### 1. Server

Operating System: Ubuntu 18.04 LTS

Database: MySQL

Software and Library : Python, Spring, TensorFlow, Nginx

### 2. Client

Operating System: Windows/macOS/Linux/Android/IOS

Software: Browser

Recommand Browser	Version
Google Chrome	44+
Mozilla Firefox	40+
Apple Safari	7+
Microsoft Edge	12+
Microsoft Internet Explorer	11+
Opera Opera	31+



## 2.5 Design and Implementation Constraints

1. Memory: Server will have 500GB internal hard drive. Softwares and database cannot exceed this amount. System administrator must notice this limitation. And each user should follow the rule that the video data uploaded each time shall not be greater than 100M
2. Language requirements: software must be multilingual, including the following languages: English, Chinese
3. Number of user: each video uploaded must be one person. Each time this system can only deal with one person's data.

## 2.6 User Documentation

Along with this system: iSport, a user manual need to be written to help users understand how to operate the system. It would be written for non-technical individuals and the level of content would differ considerably from a system administration guide, which is more detailed and complex. The user manual would follow common user documentation styles to be simple.

Trying to use step-by-step instructions for users who firstly log in to the website, by showing messaging structures, quick references, tips and glossary of terms.

User document can be written in HyperText Markup Language (HTML) or Portable Document Format (PDF) , which must describe the use of the software system.

## 2.7 Assumptions and Dependencies

It is assumed that the website will work correctly with every third-party operating system and compatible across all of the major browsers.

Assumed that the web server always runs well without down or not responding.

ISport provides two kinds of classes for user to choose: fitness classes and rehabilitation classes.

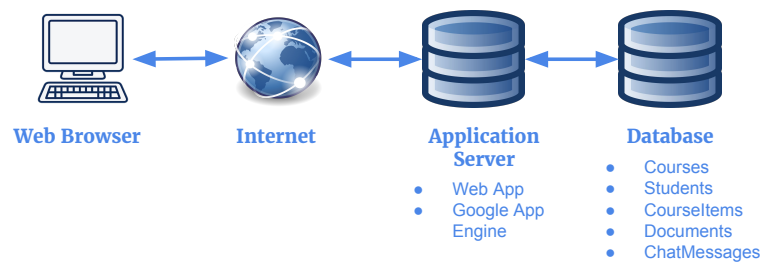
Website visitors who have not been registered are only allowed to watch some demo videos, they are not allowed to upload their posture video data before registering.

In addition to test their posture and attend into courses, members are also allowed to update their information, but they have to log in first.

# 3 System Architecture

## 3.1 Architectural Design

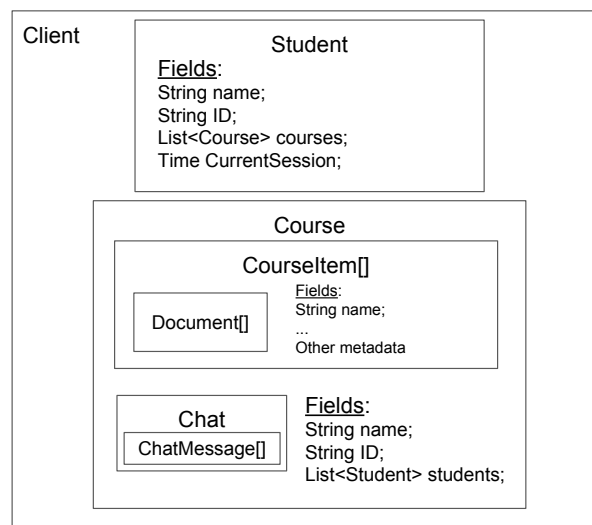
### Global Overview



## 3.2 Description of Architecture Goals

The Class Collaboration Application will be hosted by the Google App Engine and users will access the app via a web browser. The app will have access to a database containing lists of all the Courses created for the app, corresponding CourseItems, Documents, and Chatmessages, as well as the Students who have created accounts. The database will save the information

## Class Structures



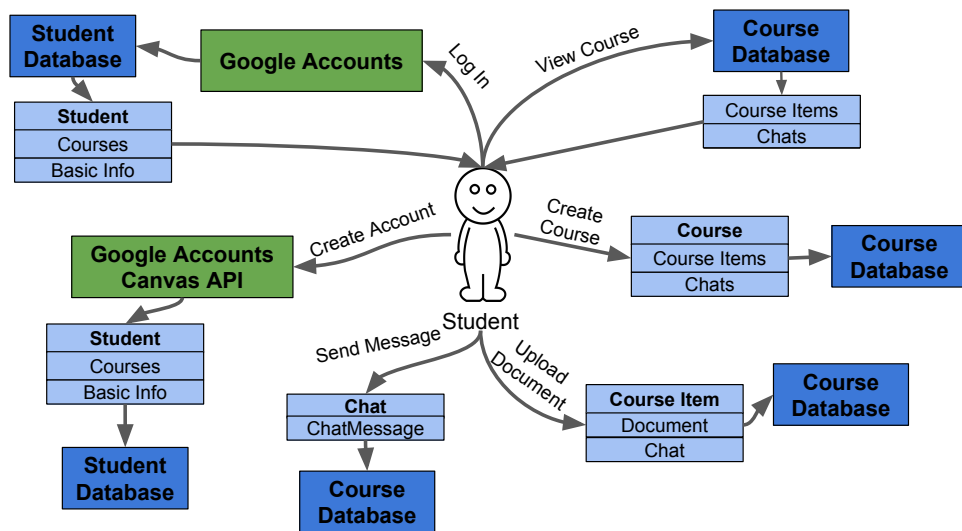
for an indefinite amount of time. Our class structures and main use cases by users are shown below.

### 3.3 Design Rationale

We have decided to use Google App Engine because it is a reliable platform to scale and build web applications. Many web applications are maintained either through IaaS or PaaS. We decided not to use IaaS because, although we can have root access to a VM, we would have to be responsible for managing the resources on the machine including, memory and CPU usage. Since Google App Engine is a PaaS, it manages all of our computational resources for us, so our only responsibility is maintaining the application while Google App Engine would take care of the infrastructure, security and scalability of the Class Collaboration Application.

For each user, we decided to integrate SSO into our application. Using SSO as a way for users to sign into the application through their Case credentials boosts our security capabilities as well as mitigates the risk of 3rd party applications accessing sensitive information about the user. This also improves user experience since the user does not have to create and keep track of another username and password.

## Student Use Cases



# 4 Principal Components

## 4.1 Component:User Info Management

### 4.1.1 Component Description

Users who use the platform for the first time need to register first. After successful registration, the system will jump to the login page. Users log in to the system through a valid user name and password, and then jump to the home page.

When users view their personal information on the personal page, they need to obtain the information saved in the database. Modifying personal information also needs to submit the modified information to the database of iSport.

All the above behaviors must interact with the database and blockchain to obtain and change the corresponding user personal information. The call relationship between classes is shown in the figure below:

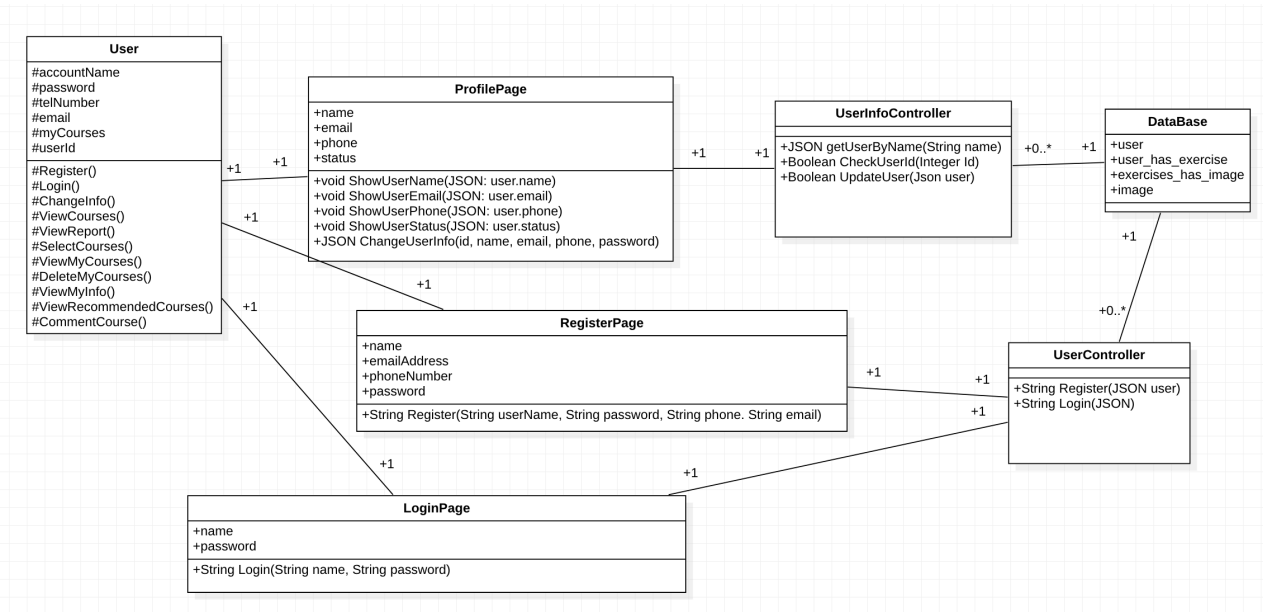


Figure 4.1: Class Diagram of UserInfo Component

### 4.1.2 Responsibilities

- User can register through UserInfo Component.
- User can login iSport by UserInfo Component.
- User can check his/her information through UserInfo Component on Profile Page.
- User can update his/her information through UserInfo Component on Profile Page.

### 4.1.3 Component Design Detail Description

#### Design for User Login in

- Input: User Name; User Password
- Output: Whether the user log in successfully
- Logical Flow:

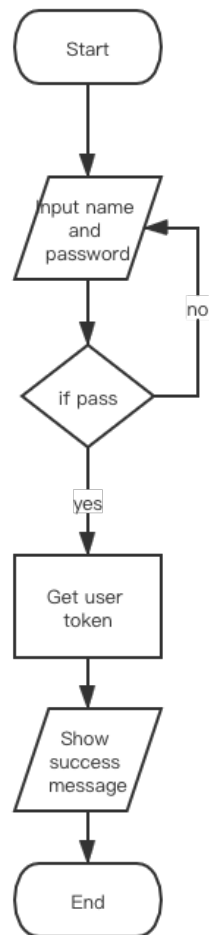


Figure 4.2: Logical Flow for User Log in

- Interface Design



Request Method	Post
Service Routing	http://[host]:6060/user/login
Params	Params1: user name; Params2: user password; Params3: request
Params Type	user name: String; user password:String; request: HttpServletRequest
Description	Check if the user's password is right according to the input user name and password.
Return Type	Json.

- Restriction and Error Handling

1. User has registered before.

## Design for User Register

- Input:

1. User Name : Got by HttpRequest (compulsory)
2. User Password: Got by HttpRequest (compulsory)
3. Email Address: Got by HttpRequest (compulsory)
4. Phone Number: Got by HttpRequest (compulsory)

- Output: Whether the user register successfully

- Logical Flow:

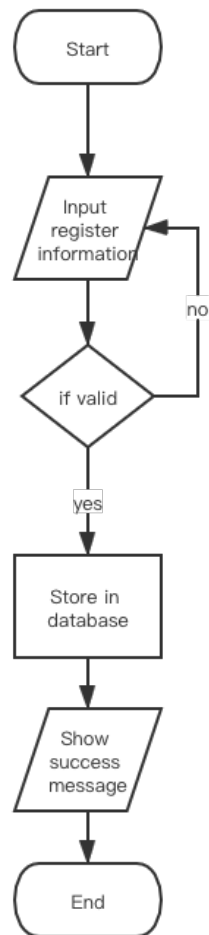


Figure 4.3: Logical Flow for User Register

- Interface Design

Request Method	Post
Service Routing	http://[host]:6060/user/register
Params	Params1: user name; Params2: user password; Params3: phone; Params4: email
Params Type	user name: String; user password:String; phone: String; email: String
Description	Help the user register iSport.
Return Type	Json.

- Restriction and Error Handling

1. User has related information to register iSport.

## Design for User Check Personal Information

- Input: null
- Output: User's profile page
- Logical Flow:

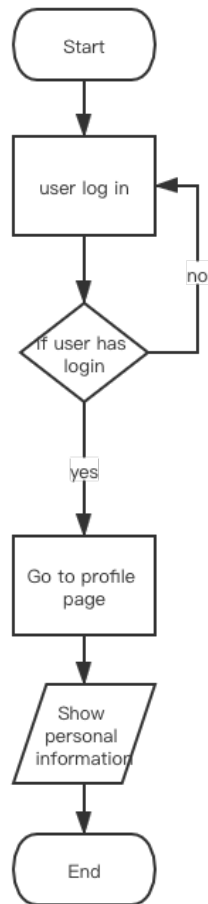


Figure 4.4: Logical Flow for User Checking Info

- Interface Design

Request Method	GET
Service Routing	http://[host]:6060/userName/info
Params	Params1: user name
Params Type	user name: String
Description	Get user's information and show on the profile page.
Return Type	Json.

- Restriction and Error Handling

1. User has personal information stored in database.

## **Design for User Update Personal Information**

- Input:
  1. user name
  2. user password
  3. email
  4. phone
- Output: User's profile page
- Logical Flow:

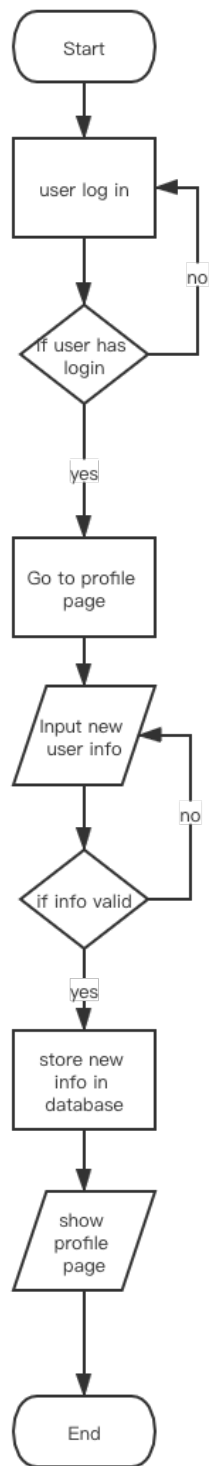


Figure 4.5: Logical Flow for User Updating Info

- Interface Design

Request Method	GET
Service Routing	http://[host]:6060/userId/updateInfo
Params	Params1: user id; Params2: user name; Params3: email; Params4: tel; Params5: password
Params Type	user id: Integer; user name: String; email: String; tel: String password: String
Description	Update user's information and refresh the profile page.
Return Type	Json.

- Restriction and Error Handling
  1. User has personal information stored in database.
  2. User provide valid new personal information.

# 5 Class Interfaces

## 5.1 Class Student

An instance of Student represents a user who can create CourseItems and Courses. A student is also able to upload Documents attached to a CourseItem.

### 5.1.1 Public Constructor Student

*Student(String name, String id, Time lastLogin)*

Creates a student object containing the name and Case ID.

### 5.1.2 Public Method GetStudentID

*String GetStudentID()*

Returns the Case ID of the user.

### 5.1.3 Public Method GetDurationOfSession

*Time GetDurationOfSession()*

Returns the length of the current login session of the user.

### 5.1.4 Public Method IsSessionExpired

*Boolean IsSessionExpired()*

Returns whether the the current login session is expired.



### 5.1.5 Public Method CreateCourseItem

*Boolean CreateCourseItem(dict options, Course course)*

Creates an instance of a CourseItem and returns whether it was successfully made.

### 5.1.6 Public Static Method CreateCourse

*Boolean CreateCourse(String courseID, String CourseName)*

Creates an instance of a Course and returns whether it was successfully made.

### 5.1.7 Public Method GetEnrolledCourses

*List<Course> GetEnrolledCourses()*

Returns the list of courses a student is currently enrolled in on the application.

### 5.1.8 Public Method JoinCourse

*Boolean JoinCourse(Course course)*

Returns whether a student successfully joined to a preexisting course.

## 5.2 Class Course

### 5.2.1 Public Constructor Course

*Course(String courseID)*

Creates a Course object from the supplied courseID. Populates private fields with items from the database queried using courseID.

### 5.2.2 Public Method GetStudents

*List<Student> GetStudents()*

Queries the students table of the database to see which students have added this course to their list of courses.

### 5.2.3 Public Method GetCourseItems

*List<CourseItem> GetCourseItems()*

Queries the database to return a list of all CourseItems associated with the course.

### 5.2.4 Public Method GetChat

*Chat GetChat()*

Returns the instance of the Chat class associated with the Course.

## 5.3 Class CourseItem

An object created by a student that can contain a document relevant to the course. This object is associated with a Course object.

### 5.3.1 Public Method RemoveCourseItem

*Boolean RemoveCourseItem()*

Removes the CourseItem from the database, so that it is no longer available to Students.

### 5.3.2 Public Method ModifyCourseItem

*Boolean ModifyCourseItem(Document document)*

Adds the document to the already existing CourseItem.

## 5.4 Class Document

### 5.4.1 Public Constructor Document

*Course(Integer documentID)*

Creates a Document object associated with the specified document ID. This ID is used to query the database to locate the actual file associated with it.

## 5.4.2 Public Method Download

*Void Download()*

Outputs the binary data for the document along with the proper HTTP header (application/octet-stream) to tell the browser to download the file instead of display it.

## 5.4.3 Public Method GetCourseItemID

*String GetCourseItemID()*

Returns the unique id associated with the course item.

# 5.5 Class Chat

Represents the chat for a course. The Chat object will hold all ChatMessage objects for a course.

## 5.5.1 Public Constructor Chat

*Chat(String courseID)*

Creates a Chat object from the supplied courseID. Populates private fields with items from the database queried using courseID. Since there is one chat per course it is ok to use the courseID as a lookup for Courses as well as Chats.

## 5.5.2 Public Method GetChatVersion

*Integer GetChatVersion()*

Returns what number message the chat has currently advanced to. This is cached whenever possible and queried often by the user to know when to request updates.

## 5.5.3 Public Method GetChatMessages

*List<ChatMessage> GetChatMessages(Integer number = 50)*

Returns that last *number* chat messages associated with this chat (default

50).

### 5.5.4 Public Method SendMessage

*Void SendMessage(String content, Student author)*

Creates a ChatMessage object with a string content and sends it to the chat.

### 5.5.5 Public Method SendDocument

*Void SendDocument(Document doc, Student author)*

Creates a ChatMessage object with a document attached and sends it to the chat.

## 5.6 Class ChatMessage

Represents a single message sent in a Chat. This message can be text or a document.

### 5.6.1 Public Constructor ChatMessage

*Chat(String courseID)*

Creates a ChatMessage object associated with the courseID.

### 5.6.2 Public Enum MessageType

Indicates the chat message type. Currently can be:

- Text
- Document
- Image

### **5.6.3 Public Method GetType**

*MessageType GetType()*

Returns the type of the message.

### **5.6.4 Public Method GetContent**

*Void GetContent()*

Returns the content associated with the ChatMessage.

### **5.6.5 Private Method GetText**

*String GetText()*

Returns the text associated with this ChatMessage

### **5.6.6 Private Method GetURL**

*String GetURL()*

Returns the URL associated with this ChatMessage.

### **5.6.7 Public Method GetStudent**

*Student GetStudent()*

Returns the Student who created the ChatMessage object.

### **5.6.8 Public Method GetTime**

*Time GetTime()*

Returns the time the ChatMessage object was sent.

### **5.6.9 Public Method Delete**

*Void Delete()*

Removes the ChatMessage object from the database. This object will no longer be shown in the Chat object class.

### **5.6.10 Public Method GetActiveUsers**

*List<Student> GetActiveUsers()*

Returns all the active users in the current chat.

### **5.6.11 Public Method GetAllUsers**

*List<Student> GetAllUsers()*

Returns all users currently subscribed to a chat.

# 6 Human Interface Design

## 6.1 Mockup of User Interface

