A Digital Gym System

—adding digital service to a traditional gym

Group 9

Group Members:

Name	QM ID	BUPT ID
Xiangrong Xing	190014571	2018212603
Shiran Yang	190014157	2018212592
Yitai Cheng	190015110	2018212608
Linfei Huang	190013596	2018212615
Yifan Zhang	190014973	2018212627
Yuanyuan Gao	190015280	2018212576

Table of Contents

1 Introduction

The software we developed is used for online gym operations under the new normal of the epidemic. This software can provide fitness enthusiasts with teaching videos of different categories and difficulties, as well as private live lessons. It can also provide gym staff with administrator functions to manage the system.
As an agile software development team, we work in accordance with the steps introduced in the course. We strive to ensure that the software meets the potential needs of customers and complete all requirements in the handout. The following content is about how we manage the development process.

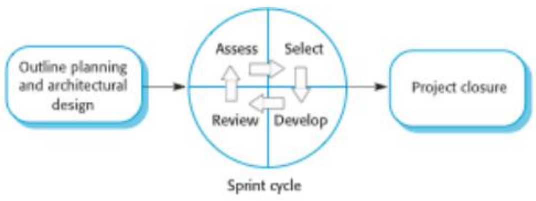
2 Project Management

2.1 Planning

In order to make our teamwork progress smoothly and efficiently, we divided the team of six students into three groups according to each person's strengths. The first group has three people with strong coding skills. They wrote the code of the video interface, the private live training interface and the administrator interface. The second group has two people. They wrote the codes for the welcome interface and the purchase interface, and integrated all the codes together at the end. The third group has a person who is good at design and summary, responsible for prototype design, testing and report writing, and wrote part of the video interface code.
In addition to the respective work of these three groups, we also hold a meeting every Tuesday where all members of the group must attend. At the meeting, we summarized the work of the previous week and made arrangements for the next week to ensure that everyone's progress was reasonable.

2.2 Project Management Techniques

To complete the software more effectively, we use Scrum approach, which is adapted to incremental development and the particular strengths of agile methods. The Scrum approach requires us to arrange short daily stand-up meetings, track the backlog of work to be done, record decisions, and measure progress against the backlog.



2.3 Project Management and Communication Tools

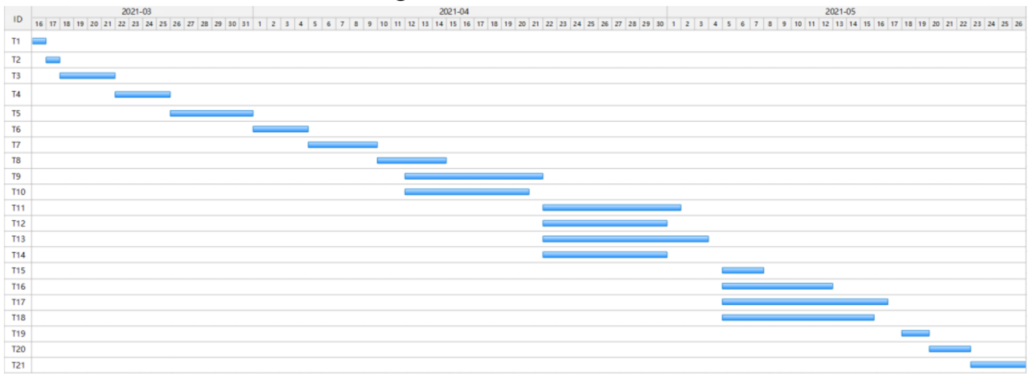
For daily communication, we have formed a WeChat group, where we inform the meeting time and supervise each other.
For file sharing, we update the new version on QMplus HUB, and it can be checked by teaching assistants.

2.4 Time Estimating

The task chart shows project breakdown into tasks:

Stage	Activity	Date	Duration	Dependencies
Requirements	T1: thoughts sharing	3.16	1	
	T2: teamwork plan	3.17	1	
	T3: user stories and product backlog	3.18-3.21	4	T1,T2
	T4: prototype	3.22-3.25	4	T3
Analysis and design	T5: identify entities, boundary and control classes	3.26-3.31	6	
	T6: architecture design	4.1-4.4	4	T5
	T7: design pattern	4.5-4.9	5	T4
Implementation	T8: welcome interface v1	4.10-4.14	5	T7
	T9: video interface v1	4.12-4.21	10	T7
	T10: administrator interface v1	4.12-4.20	9	T7
	T11: video interface v2	4.22-5.1	10	T9
	T12: administrator interface v2	4.22-4.30	9	T10
	T13: private live training interface v1	4.22-5.3	12	T7
	T14: purchase interface v1	4.22-4.30	9	T7
	T15: welcome interface v2	5.5-5.7	3	T8
	T16: video interface v3	5.5-5.12	8	T11
	T17: private live training interface v2	5.5-5.16	12	T13
	T18: purchase interface v2	5.5-5.15	11	T14
Testing	T19: test plam	5.18-5.19	2	
	T20: partial test and adjustment	5.20-5.22	3	T19
	T21: final test	5.23-5.26	4	T20

The Gantt chart shows schedule against calendar time:



2.5 Decision Making

Every Tuesday, the team leader made an appointment for the seminar room and organized a meeting. The specific time could be adjusted according to the situation. The meeting is mainly divided into two parts. In the first part, each of us briefly reported what we did last week. If we encounter difficult issues, we can raise them in the meeting and then discuss them together. In the second part, we made arrangements for the work of the next week. In this process, the team leader must strictly control the overall progress to ensure the rationality of the work.

2.6 Adapting to Changes

Since we are an Agile software development team, we should cater to changing customer requirements. In each iteration, we need to update the backlog and implement new functions within the specified time.

3 Requirements

3.1 Fact-finding Techniques

1. Background Reading

Background reading is the first step in discovering requirements. At this stage, we learn about the gym business through the following two ways. The first is to read the handout and specify the functions required to be implemented. The second is to download KEEP, a well-known fitness application, to learn about the functions of existing system. Through these two methods, we determined that the software includes welcome, video, private live training, purchase, and administrator functions.

2. Interviewing

Interviewing is the most direct way to obtain customer needs. At this stage, we interviewed some fitness enthusiasts among our classmates. For different people, their preferences and abilities are different. So, our software should provide users with options of different types and different difficulties.

3. Document Sampling

Document sampling helps to achieve non-functional requirements. We collect copies of documentation, obtain details, and identify patterns in requirements for the system. Then, we get to know that the historical information of each user needs to be recorded to ensure that the user information will not be reset after they log in again.

3.2 User Stories and Iterations Planning

1. User Stories

More details are in the appendix.

Story ID: O02

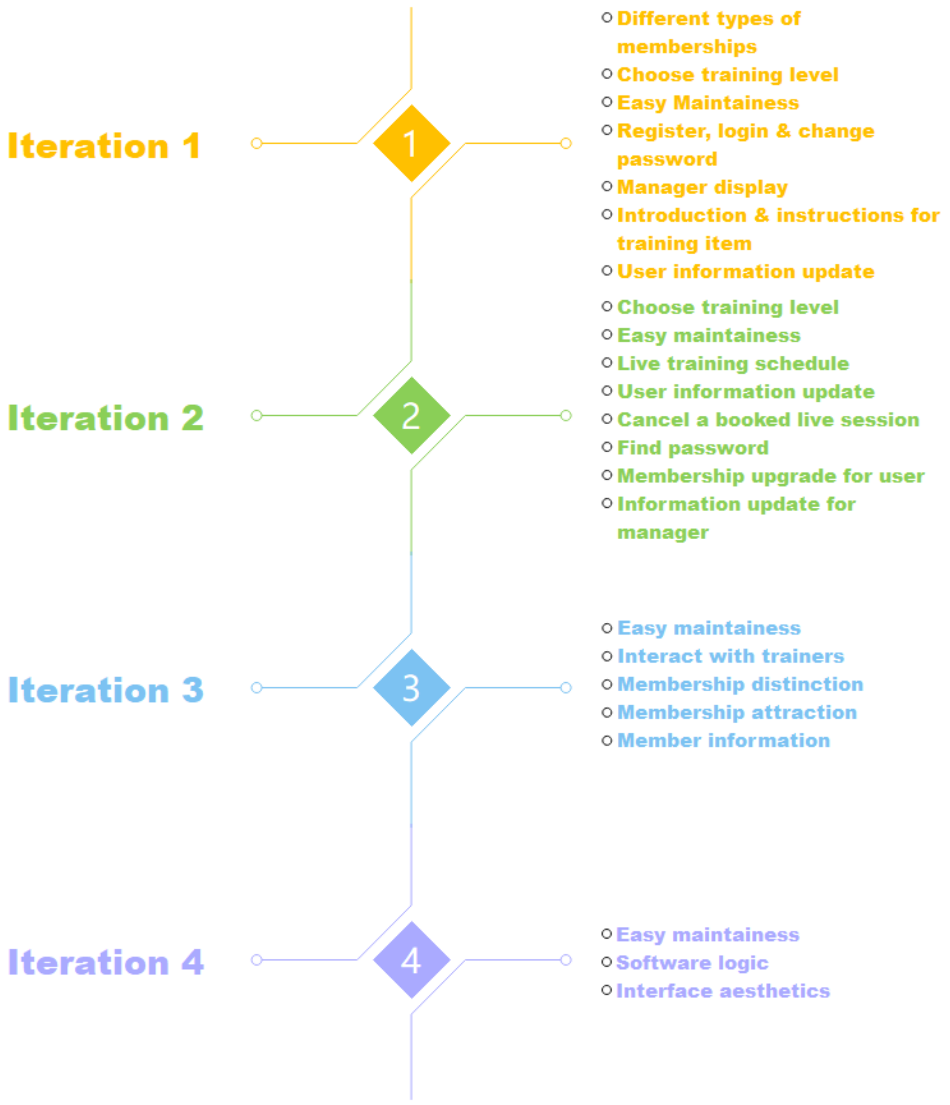
Story name: Different types of memberships

Description:
As a gym owner, I want to provide several different types of memberships. So that I can provide video courses to all members and live training with a personal trainer to VIP members.

Priority (high 1 2 3 4 5low) : 1
Date: 3.30-4.13

Acceptance Criteria:
1. Membership should be divided into ordinary members and VIP members.
2. Video lessons are available to all members.
3. Only VIP members can get live training with a personal trainer.

2. Iterations Planning

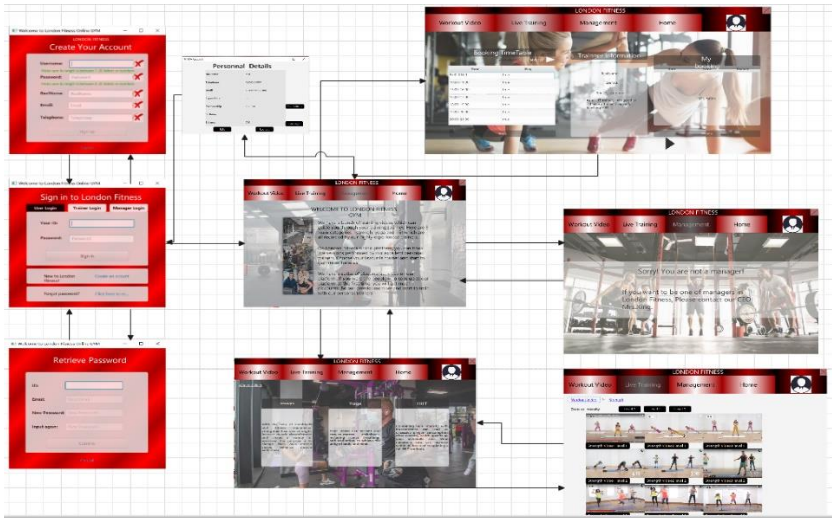


3.3 Adapt to Changes

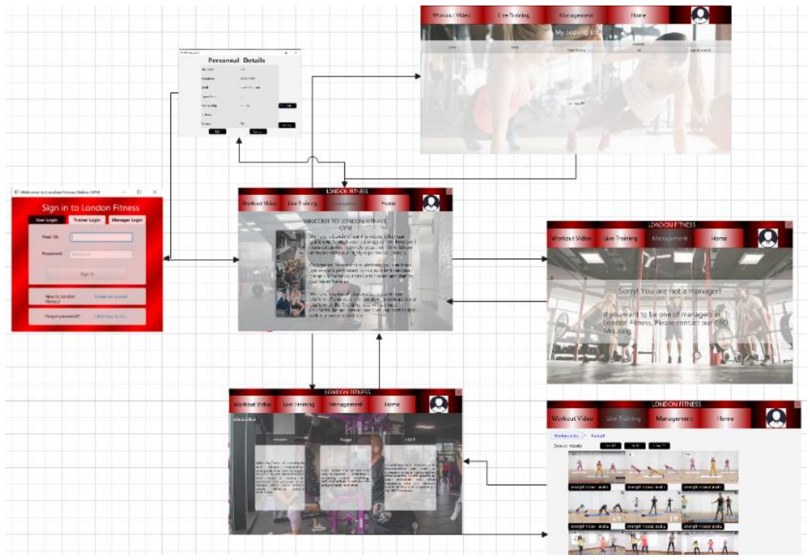
During the five iterations, we continued to refine the requirements. In this process, we encountered many problems, which slowed down our progress and sometimes even exceeded the expected time.
In solving this problem, weekly meetings have played an important role. In the meeting, we can discuss the problems together and quickly come up with solutions. In addition, we have also strictly controlled the progress of work through meetings.

3.4 Prototype

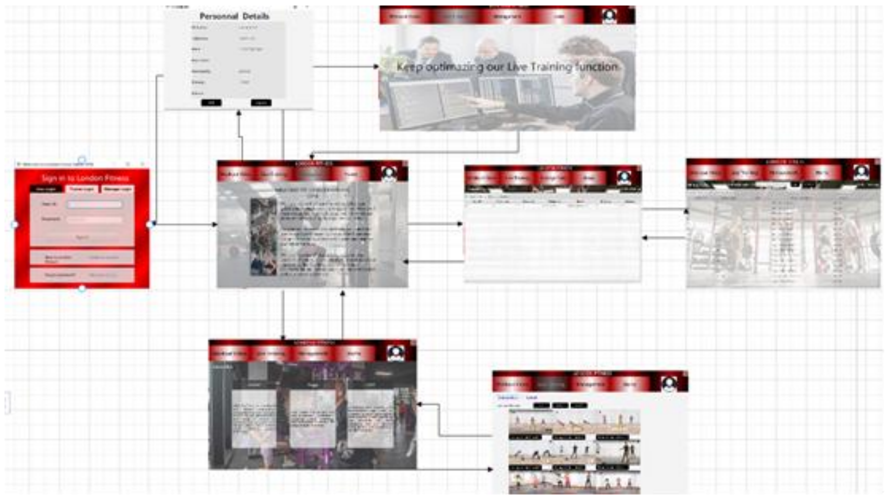
1. Prototype for User



2. Prototype for Trainer



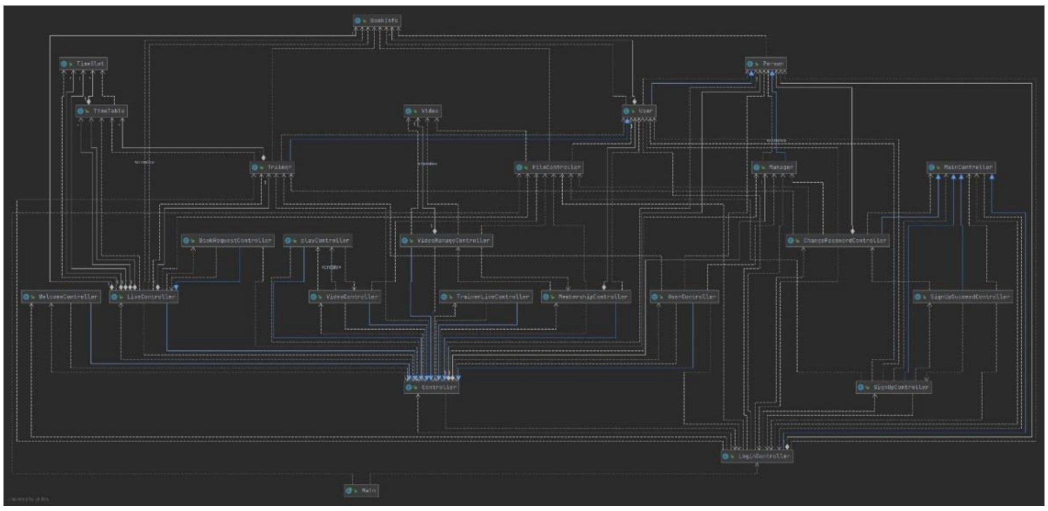
3. Prototype for Manager


Personal Details
Keep optimazing our Live Training function
Sign in to London Fitness

4 Analysis and Design

4.1 Class Diagram and Justification

4.1.1 Class Diagram



4.1.2 Justification for Adaptability to Change

We define a controller class which all other controller classes inherits. Using this inheritance characteristic, we define specific functions for users, administrators and coaches in subclasses. In this way, when we modify or add the corresponding part of the function, we can only modify the code of the corresponding subclass without affecting the overall code.
For modularity, we separate the functionality of a program into video, administrator

information management, user information display, user login and information registration, and private training appointment modules. And each module contains everything necessary to execute its desired functionalities.
Besides, we can add and delete video changes by reading files through the administrator interface, and display the operation results clearly on the interface to meet the needs of any market and customers.

4.2 Refine the Requirements

The refinement of requirements is the elaboration of detail for all abstractions. Through group meetings, we refine the different functions according to the user and manager categories. In actual situation, some details of the functions are not suitable or possible to realize. Then we modify some aspects of the functions. And we iterate the user story and modify the acceptance criteria of some user stories. Start and end time are also adjusted according to the actual situation. We also wrote four functions that are not listed in the product backlog and had added them to the product backlog later.

4.3 Design of the Software

1. **Entity Class**
   Entity classes includes Video, User, TimeTable, TimeSlot, BookInfo and BookInfoForTrainer.

More details are in the appendix. For example:

TimeTable	List<TimeSlot> getTimeSlotList()	Getter function of information of time slot
	void setTimeSlotList(List<TimeSlot>)	Setter function of information of time slot

2. **Control Class**
   Control classes includes Controller, PageController, FileController, yogaController, VideoController, UserController, ChangePasswordControlller, LoginController, SIgnUpController, BookRequestController, InformationRevisionController, LiveController, MembershipController, playController, TrainerLiveController, PurchaseMembershipController, RechargeController, StrengthControlle and HIITController,
   More details are in the appendix. For example:

Class	Method	Description
PageController	List<Object> replaceStage (String , Stage)	Replace the current stage
	Initializable	Replace the current scene content

	replaceSceneContent (String , Stage)	

4.4 Design Principle

1. Simple Responsibility Principle (SRP)

Based on the principle, we tried to make one class focus on carry out the single function, so that the change of the behavior in one class won't influence the functions in other classes. This point can be shown obviously in many aspects of our program. For the class "FileController", the only responsibility it deals with is the data process of the json file. For the class "playController", the only responsibility it deals with is playing the fitness video. And also, for the class "RechargeController", the only responsibility it deals with is allowing the user to recharge money. These all meet the Simple responsibility principle that every class in a system perform a single responsibility. Moreover, although we introduce methods from other classes to help one class, most of the code inside one class works mainly with code inside the same class, which demonstrate our code with high cohesion and low coupling.

2. Open-Close Principle (OCP) & Don't Repeat Yourself (DRY)

OCP means we can make a module behave in new and different ways as requirements change or to meet new needs but we should be able to do this in a way that does not require changing the code of the module. Based on the principle, writing a subclass is a way of modifying the way the code of the superclass works without changing that actual code. We tried to put the general code in parent class, and when requirements change or to meet new needs, we just add new functionality by writing the subclass of the general class. Our codes meet OCP in other way too. A variable of a class type can be set to refer to an object whose type is a subclass. That can be seen in our code as we always set Trainer and Manager to their superclass Person. In this way, an existing method with a parameter of a particular class will work with an object of a subclass of that class as an argument without any need to make changes to it. We also tried to ensure other classes won't directly change the code of one class by declaring the fields in the class as private. For example, in the iteration 2, we write the class yogaController and HIITController. The functionality of them is same, so they share a lot of similar code. So, in iteration 4, we put the general similar code into the class VideoController, and make yogaController and HIITController inherit it, so that the total length of code is much smaller. If we want to add more new features and set corresponding control class like BoxingController, we don't need to modify VideoController. In this way, we achieve the OCP principle. Moreover, we provide static methods for controller layer to invoke. For example, file reading and file writing methods are positioned in one class. When other classes need to read data from or write data to the json file, instead of writing the corresponding code again, we just invoke the static methods in FileController class to perform the operation. We use this principle in our design to avoid duplicate code and make our code keep high quality.

3. Liskov Substitution Principle (LSP)

The LSP principle says that methods should not be overridden in a way that changes assumptions about their behavior. This principle reminds us that the subclasses must always be substitutable for the class they extend. In our program, although we use a lot of inheritance to reduce the code reuse problem, we tried to declare objects whose actual type and apparent type are correspondence so that we can prevent the code behaving in an unexpected way. This allows us to realize the Liskov Substitution Principle.

4. The Dependency-Inversion Principle (DIP)

The DIP can be stated as: High-level modules should not depend on low-level modules, instead, both should depend on abstractions; abstractions should not depend on details; instead, details should depend on abstractions. In our program, the initialize method only depends on the abstraction Initializable. It leaves the details to the classes which implement Initializable interface. The abstraction Initializable means that some code will be implemented before the fxml file is loaded, it does not consider what implementation will be carried out. The detailed decisions on what implementation will be done are left to classes which implement the interface Initializable, which means the details of these classes are dependent on the interfaces they implement.

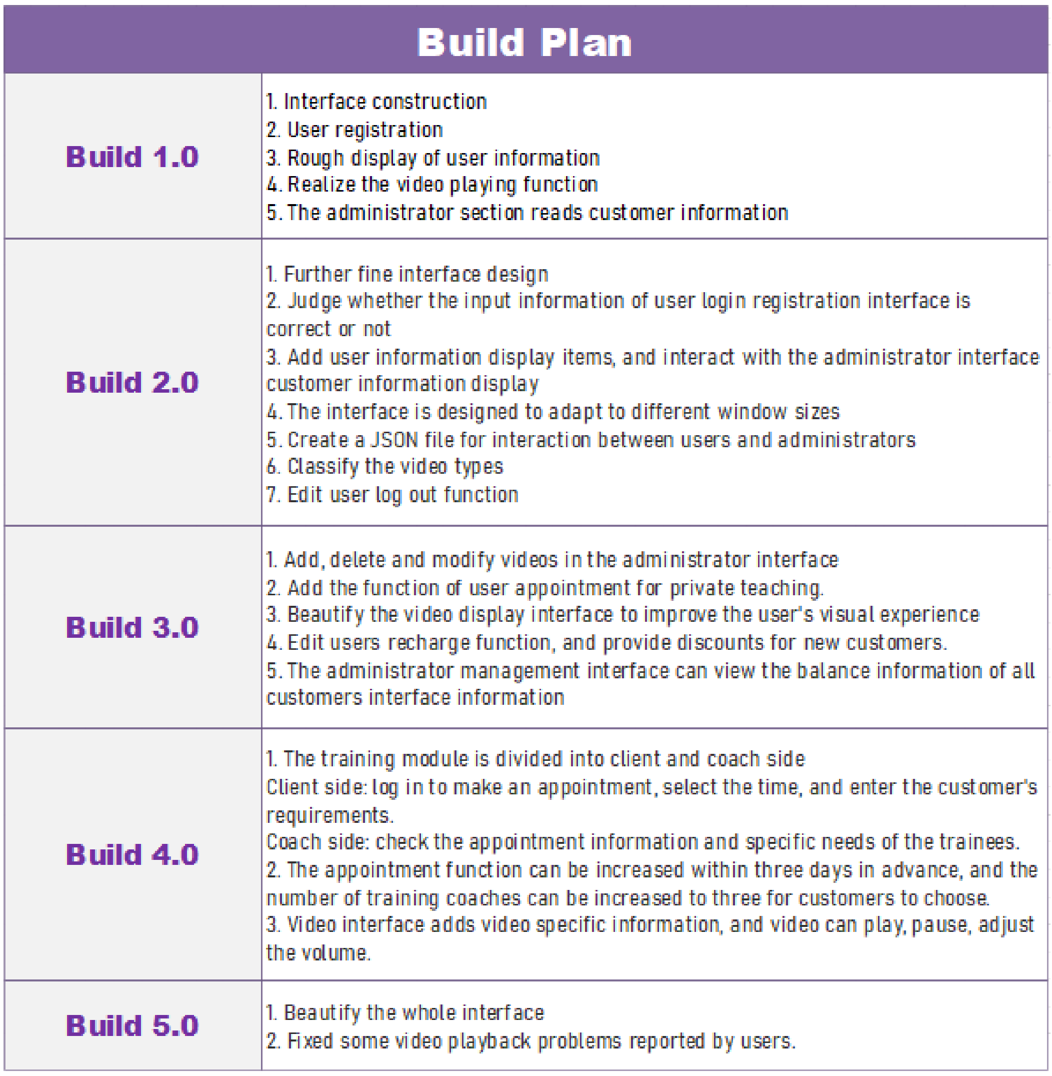
5 Implementation and Testing

5.1 Implementation

5.1.1 Implementation Strategy

We build our software incrementally in the manageable steps. For high-level user requirements, it is implemented in an earlier version. Our program uses JavaFX to build GUI blocks, and uses Java code to interact with blocks. We follow a two-week iteration. In each iteration, we constantly update the backlog, actively obtain user feedback, analyze user requirements, and implement code quickly to update the new version. We design the software system in terms of modules which makes code easy for reuse.

5.1.2 Build Plan



Build Plan	
Build 1.0	1. Interface construction 2. User registration 3. Rough display of user information 4. Realize the video playing function 5. The administrator section reads customer information
Build 2.0	1. Further fine interface design 2. Judge whether the input information of user login registration interface is correct or not 3. Add user information display items, and interact with the administrator interface customer information display 4. The interface is designed to adapt to different window sizes 5. Create a JSON file for interaction between users and administrators 6. Classify the video types 7. Edit user log out function
Build 3.0	1. Add, delete and modify videos in the administrator interface 2. Add the function of user appointment for private teaching. 3. Beautify the video display interface to improve the user's visual experience 4. Edit users recharge function, and provide discounts for new customers. 5. The administrator management interface can view the balance information of all customers interface information
Build 4.0	1. The training module is divided into client and coach side Client side: log in to make an appointment, select the time, and enter the customer's requirements. Coach side: check the appointment information and specific needs of the trainees. 2. The appointment function can be increased within three days in advance, and the number of training coaches can be increased to three for customers to choose. 3. Video interface adds video specific information, and video can play, pause, adjust the volume.
Build 5.0	1. Beautify the whole interface 2. Fixed some video playback problems reported by users.

5.2 Testing

5.2.1 Testing Strategy

Testing strategy in our group includes for aspects: What tests to run, how to run them, when to run them and how to determine whether the testing effort is successful.

- 40% of our tests are automated and the remainder is manual.
- Each subject use case should be tested for its normal flow and behavior and also two alternative flows.
- These should ensure that they cover incorrect user input.
- Our success criteria: 90% of test cases passed.
- No high priority defects unresolved.

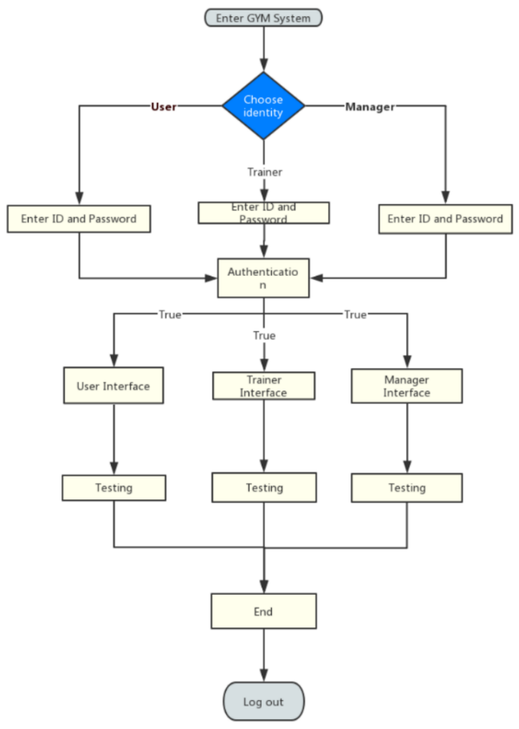
5.2.2 Testing Techniques

1. Black-box Testing

Black-box testing is to test the functional requirements of the software. It attempts to find errors including inputs causing anomalous behavior and outputs which reveal the presence of defects.
During the incremental development of the software, new functionalities will be added into the prototype. Black-box implements the increasing cases to test whether the additional functionality is correct. Based on the black-box testing, the iteration process and the whole structure can be updated.

2. White-box Testing

White-box testing is to ensure that all statements and conditions have been executed at least once and test the internal logic of the application. White-box testing is implemented at the component level. All independent paths within a module have been exercised at least once.
During the process of software development, we use the basic path testing to implement the testing.



5.2.3 TDD

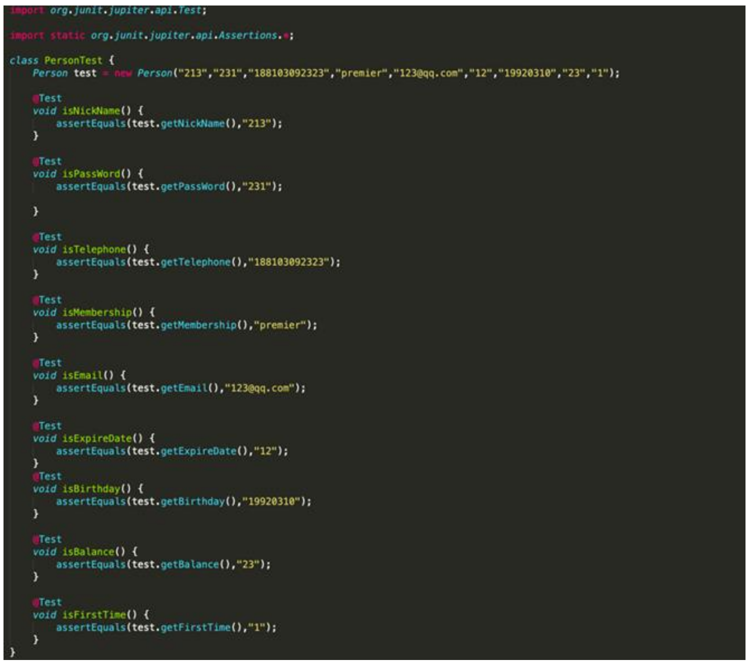
We use the Test-Driven Development in our project to ensure the validity of the system. This is to let the code to meet the specification by writing test prior to the production code. In our development, as the architecture of our software is Boundary-Controller-

Entity, so we used JUnit to test each part of the code to make sure they can connect with each other successfully.
We use JUnit as a tool to check whether the system is operated as I expected. Only when these methods work correctly can we develop the other things. and the TDD circle is as following:

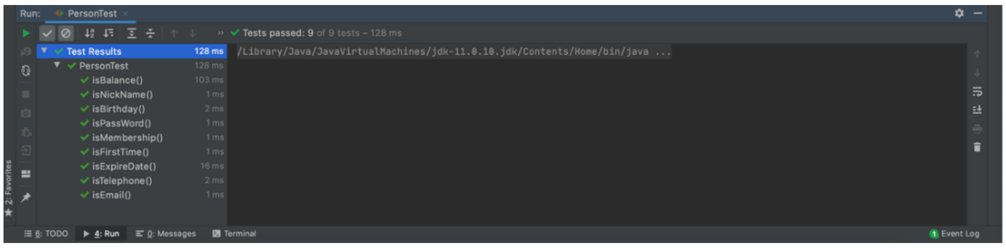
- Write a specification, in code and in the form of a unit test. The test verifies a functional unit of your code.
- Demonstrate test failure.
- Write code to meet the specification.
- Demonstrate test success.
- Refactor the code, to ensure that the system still has an optimally clean code base.

Once it finishes, it will execute the loop again and again.

For example, for the class Person.java, we implement the TDD for it and the screen shots are shown as follows:

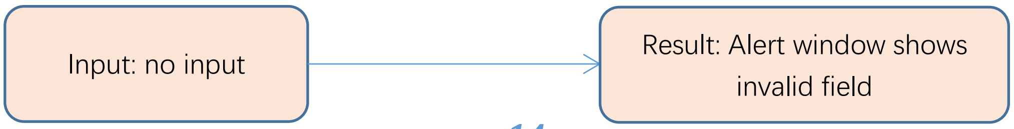


```java
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

class PersonTest {
    Person test = new Person("213","231","188103092323","premier","123@qq.com","12","19920310","23","1");

    @Test
    void isNickName() {
        assertEquals(test.getNickName(),"213");
    }

    @Test
    void isPassWord() {
        assertEquals(test.getPassWord(),"231");

    }

    @Test
    void isTelephone() {
        assertEquals(test.getTelephone(),"188103092323");
    }

    @Test
    void isMembership() {
        assertEquals(test.getMembership(),"premier");
    }

    @Test
    void isEmail() {
        assertEquals(test.getEmail(),"123@qq.com");
    }

    @Test
    void isExpireDate() {
        assertEquals(test.getExpireDate(),"12");
    }
    @Test
    void isBirthday() {
        assertEquals(test.getBirthday(),"19920310");
    }

    @Test
    void isBalance() {
        assertEquals(test.getBalance(),"23");
    }

    @Test
    void isFirstTime() {
        assertEquals(test.getFirstTime(),"1");
    }
}
```

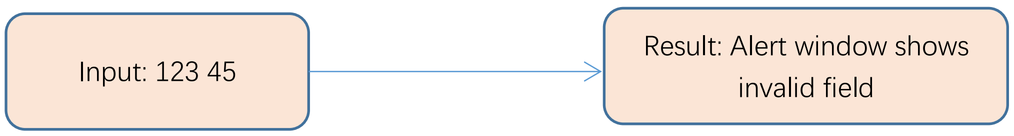


```
Run:  PersonTest
✓ Tests passed: 9 of 9 tests – 128 ms
▼ ✓ Test Results        128 ms   /Library/Java/JavaVirtualMachines/jdk-11.0.10.jdk/Contents/Home/bin/java ...
  ▼ ✓ PersonTest        128 ms
      ✓ isBalance()     103 ms
      ✓ isNickName()      1 ms
      ✓ isBirthday()      2 ms
      ✓ isPassWord()      1 ms
      ✓ isMembership()    1 ms
      ✓ isFirstTime()     1 ms
      ✓ isExpireDate()   16 ms
      ✓ isTelephone()     2 ms
      ✓ isEmail()         1 ms
```

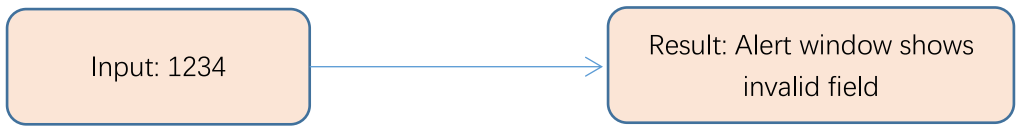
For example, to test the funtion of restricting user information revision for telephone, we use partition testing method.
Partition 1: Input the telephone with null field



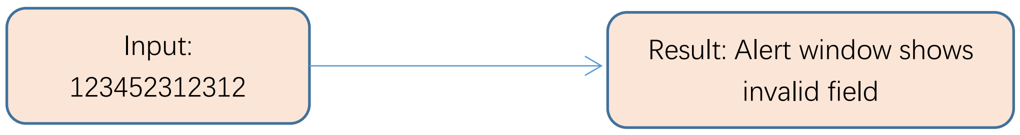
Partition 2: Input the telephone with blank



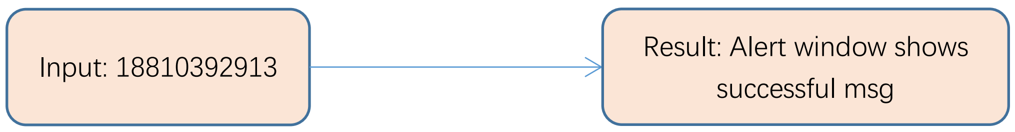
Partition 3: Input the email with length less than 5



Partition 4: Input the email with length greater than 11



Partition 5: Input the email with length 11



5.2.4 Test Case

For example, component level：Person.java

Person	Correct input	Incorrect Input
Input	Correct user info	Incorrect user info
Result	User information can be correctly printed on the screen	Invalid information is printed on the screen
Condition	No condition exists	No condition exists
Disciplined techniques	Black box testing	Black Box testing

For example, system level：FileController.java

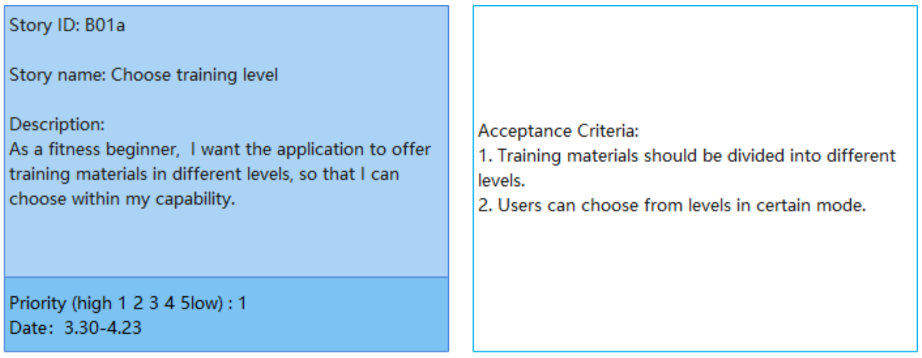
Fileoutput	Correct input	Incorrect Input
Input	Correct json file address	Incorrect json file address
Result	User information can be correctly extracted from json file	No information will be extracted
Condition	No condition exists	No condition exists
Disciplined techniques	Black box testing	Black Box testing

6 Appendix

6.1 Reference

[1] Lecture sild EBU6304 Project Management
[2] Lecture sild EBU6304 Processes and Agile
[3] Lecture sild EBU6304 Design Principles
[4] Lecture sild EBU6304 Implementation and Testing

6.2 User Story



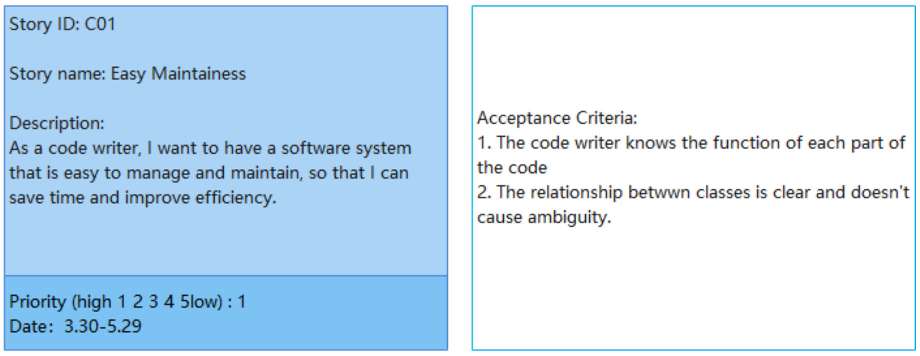
Story ID: B01a

Story name: Choose training level

Description:
As a fitness beginner, I want the application to offer training materials in different levels, so that I can choose within my capability.

Priority (high 1 2 3 4 5low) : 1
Date: 3.30-4.23

Acceptance Criteria:
1. Training materials should be divided into different levels.
2. Users can choose from levels in certain mode.



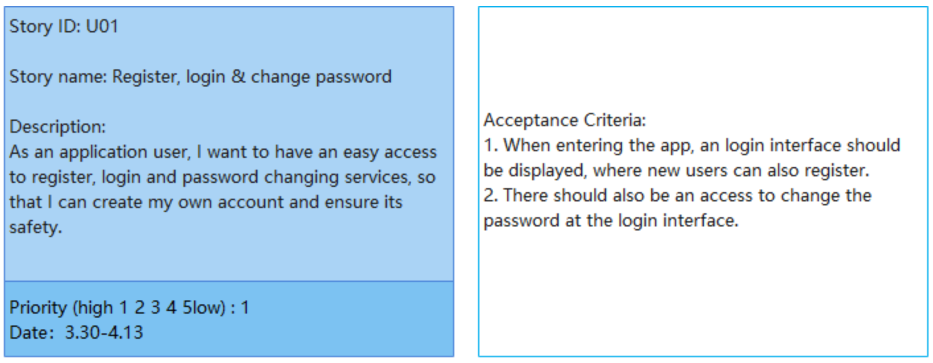
Story ID: C01

Story name: Easy Maintainess

Description:
As a code writer, I want to have a software system that is easy to manage and maintain, so that I can save time and improve efficiency.

Priority (high 1 2 3 4 5low) : 1
Date: 3.30-5.29

Acceptance Criteria:
1. The code writer knows the function of each part of the code
2. The relationship betwwn classes is clear and doesn't cause ambiguity.



Story ID: U01

Story name: Register, login & change password

Description:
As an application user, I want to have an easy access to register, login and password changing services, so that I can create my own account and ensure its safety.

Priority (high 1 2 3 4 5low) : 1
Date: 3.30-4.13

Acceptance Criteria:
1. When entering the app, an login interface should be displayed, where new users can also register.
2. There should also be an access to change the password at the login interface.

Group 9

Story ID: M01 Story name: Manger display Description: As a manager of the membership, I want to have an interface where I can clearly see the type of membership and other information, so that I can manage users better. Priority (high 1 2 3 4 5low) : 1 Date： 3.30-4.13	Acceptance Criteria: 1. When entering the app, an login interface should be displayed, where new users can also register. 2. There should also be an access to change the password at the login interface.
Story ID: O03 Story name: Live training schedule Description: As a gym owner, I want to have live training schedules for trainers. So that VIP members can only book live training during the trainer's free time to avoid time conflicts. Priority (high 1 2 3 4 5low) : 1 Date： 4.14-4.23	Acceptance Criteria: 1. Set schedules for trainers who offers live training. 2. Users can only book live sessions during the trainer’ s free time. 3. Booking information should be stored in a file.
Story ID: P02 Story name: Interact with trainers Description: As a user proficient in gym I want to contact with trainers on my targets and physical abilities So that trainers can design specific exercises for me. Priority (high 1 2 3 4 5low) : 2 Date： 4.24-5.22	Acceptance Criteria: When the user books a live session, the detailed information of the user can be sent to the trainer.
Story ID: B02 Story name: Introduction & instructions for training item Description: As a fitness beginner, I want the application shows brief introduction and instructions for each training item, So that I can know whether I need and how can I practice. Priority (high 1 2 3 4 5low) : 2 Date： 3.30-4.13	Acceptance Criteria: 1. Verify that there are relating words or sentences to describe the training item in the choosing interface. 2. Make sure that clear and accurate instructions will be given for each training item.
Story ID: O01 Story name: Membership distinction Description: As a gym owner, I want to ensure that different type of users have differnet access permission to different sections, so that the function can be used by appropiate user group. Priority (high 1 2 3 4 5low) : 2 Date： 4.24-5.22	Acceptance Criteria: Verify that the system can distinguish user type according to the log in information and set different access based on it.

Group 9

Story ID: O04

Story name: Membership attraction

Description:
As a gym owner, I want that the application offers some discount, so that we can always attract customers.

Priority (high 1 2 3 4 5low) : 2
Date: 4.24-5.22

Acceptance Criteria:
There should be a discount offered when topping up for the first time.

Story ID: U02

Story name: User information update

Description:
As an application user, I want that it will be possible for me to change some of my user information, so that I can update my newest physical conditions or personal preferences.

Priority (high 1 2 3 4 5low) : 2
Date: 3.30-4.23

Acceptance Criteria:
1. There should be a UI that displays some of theie personal information for users to check.
2. Users should be able to make some changes within that interface

Story ID: P03

Story name: Cancel a booked live session

Description:
As a user proficient in gym
I want to cancel the live classes before the start time
So that I can have a better user experience.

Priority (high 1 2 3 4 5low) : 3
Date: 4.14-4.23

Acceptance Criteria:
An obvious button for cancel should be displayed in the interface of booking lessons.

Story ID: T01a

Story name: Member information

Description:
As a personal trainer, I want to know exactly the member' s targets and physical abilities, so that I can have a better design of my teaching content and service for my members better.

Priority (high 1 2 3 4 5low) : 3
Date: 4.24-5.22

Acceptance Criteria:
1. The member's physical information is stored in a file
2. The member has a clear training objectives

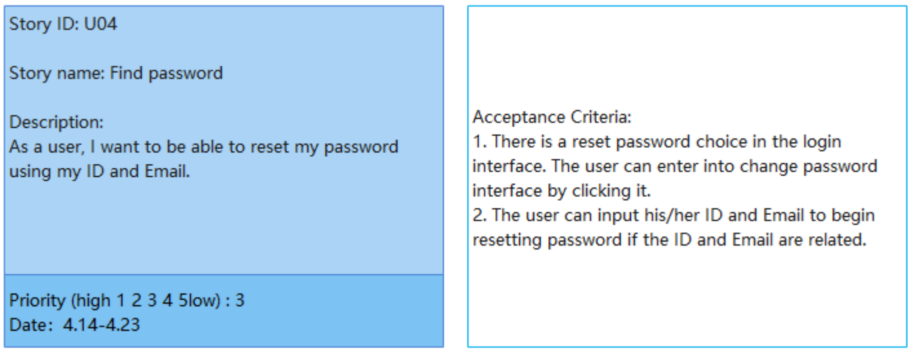
Story ID: U03

Story name: Sign up hint

Description:
As a user, I want to be noticed if I input some illegal characters or violate naming rules

Priority (high 1 2 3 4 5low) : 3
Date: 4.14-4.23

Acceptance Criteria:
When user input illegal characters or violate naming limits, the system should give user some feedback.



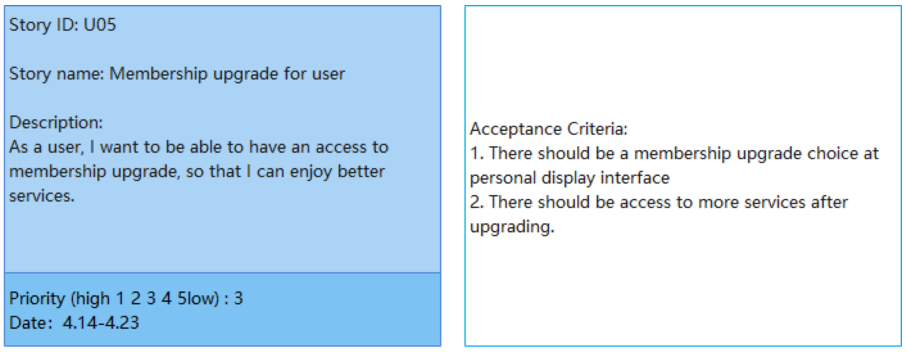
Story ID: U04

Story name: Find password

Description:
As a user, I want to be able to reset my password using my ID and Email.

Priority (high 1 2 3 4 5low) : 3
Date: 4.14-4.23

Acceptance Criteria:
1. There is a reset password choice in the login interface. The user can enter into change password interface by clicking it.
2. The user can input his/her ID and Email to begin resetting password if the ID and Email are related.



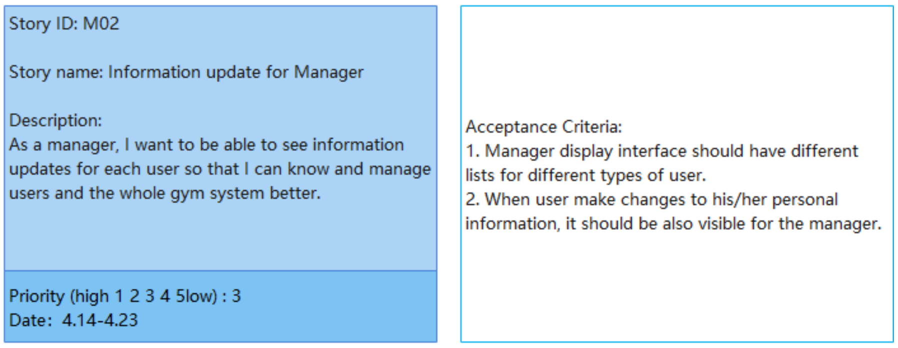
Story ID: U05

Story name: Membership upgrade for user

Description:
As a user, I want to be able to have an access to membership upgrade, so that I can enjoy better services.

Priority (high 1 2 3 4 5low) : 3
Date: 4.14-4.23

Acceptance Criteria:
1. There should be a membership upgrade choice at personal display interface
2. There should be access to more services after upgrading.



Story ID: M02

Story name: Information update for Manager

Description:
As a manager, I want to be able to see information updates for each user so that I can know and manage users and the whole gym system better.

Priority (high 1 2 3 4 5low) : 3
Date: 4.14-4.23

Acceptance Criteria:
1. Manager display interface should have different lists for different types of user.
2. When user make changes to his/her personal information, it should be also visible for the manager.

6.3 Design of Software

1. Entity Class

Class	Method	Description
Video	void setVideoID(String)	Setter function of video id
	String getVideoID()	Getter function of video id
	void setLevel(String)	Setter function of level
	String getLevel()	Getter function of level
	void setType(String)	Setter function of type
	String getType()	Getter function of type
	void setVideoLink(String)	Setter function of video link

	String getVideoLink()	Getter function of video link
	void setStatus(String)	Setter function of status
	String getStatus()	Getter function of status
	void setVideoDescription(String)	Setter function of video description
	String getVideoDescription()	Getter function of video description
User	void setBalance(String)	Setter function of balance
	String getBalance()	Getter function of balance
	void setFirstTime(String)	Setter function of first time
	String getFirstTime()	Getter function of first time
	List<BookInfo>getBookInfoList()	Getter function of Booking information
	void setBookInfoList(List<BookInfo>)	Getter function of Booking information
	void setUserID(String)	Setter function of user id
	String getUserID()	Getter function of user id
	String getMembership()	Getter function of membership
	void setMembership(String)	Setter function of membership
	String getRealName()	Getter function of real name
	void setRealName(String)	Setter function of realname
	String getUserName()	Getter function of user name
	void setUserName(String)	Setter function of user name
	String getPassword()	Getter function of password
	void setPassword(String)	Setter function of password
	String getTel()	Getter function of telephone
	void setTel(String)	Setter function of telephone
	String getEmail()	Getter function of email address
	void setEmail(String)	Setter function of email address
	String getExpireDate()	Getter function of expire date

	void setExpireDate(String)	Setter function of expire date
	String getBirthday()	Getter function of birthday
	void setBirthday(String)	Setter function of birthday
	List<BookInfoForTrainer> getBookInfoForTrainerList()	Getter function of booking information of trainer
TimeTable	List<TimeSlot> getTimeSlotList()	Getter function of information of time slot
	void setTimeSlotList(List<TimeSlot>)	Setter function of information of time slot
TimeSlot	boolean isBookingFlag()	Getter function of the booking flag
	String getDuration()	Getter function of the duration
	void setBookingFlag(boolean)	Setter function of the booking flag
	void setDuration(String)	Setter function of the duration
BookInfo	String getTrainer()	Getter function of the trainer
	void setTrainer(String)	Setter function of the trainer
	String getDate()	Getter function of the date
	void setDate(String)	Setter function of the date
	String getTime()	Getter function of the time
	void setTime(String)	Setter function of the time
BookInfoForTrainer	String getUserRequest()	Getter function of the user request
	void setUserRequest(String)	Setter function of the user request
	String getDate()	Getter function of the date
	void setDate(String)	Setter function of the date
	String getTime()	Getter function of the time
	void setTime(String)	Setter function of the time
	String getUserName()	Getter function of the user name
	void setUserName(String)	Setter function of the user name
	String getUserTel()	Getter function of the user phone number
	void setUserTel(String)	Setter function of the user phone number

2. Control Class

Class	Method	Description
PageController	List<Object> replaceStage (String , Stage)	Replace the current stage
	Initializable replaceSceneContent	Replace the current scene content

	(String , Stage)	
FileController	Map<String, Object> parseFile (String)	Read data from json file
	void updateFile (String, Map<String, Object>)	Write data to the json file
	User getInfo(String , String)	Read information from json file
	void writeFile(String , User)	Write information to the json file
	Video getVideoInfo(String, String)	Read video information from json file
	void writeVideoFile(String, Video)	Write video information to the json file
yogaController	void select1()	Select yoga level 1
	void select2()	Select yoga level 2
	void select3()	Select yoga level 3
	void back(MouseEvent)	Handle the event of clicking back button
	void refresh(MouseEvent)	Handle the event of clicking refresh button
VideoController	void onStrengthButtonClick (MouseEvent)	Handle the event of clicking Strength button
	void onHIITButtonClick (MouseEvent)	Handle the event of clicking HIIT button
	void onyogaButtonClick (MouseEvent)	Handle the event of clicking yoga button
	void showVideos(String)	Show the videos
	void selectLevel1(String)	Select videos of level 1
	void selectLevel2(String)	Select videos of level 2
	void selectLevel3(String)	Select videos of level 3
UserController	void showPersonDetails(User)	Show the detail information of the person
	void handleEditPerson()	Handle the event of clicking edit button
	boolean showPersonEditDialog (User)	Show the dialog of the stage
	void handleRecharge()	Handle the event of clicking recharge button
	void handleMembership()	Handle the event of clicking membership button
	boolean showRechargePage (User)	Display the recharge page

	boolean showMembershipPage (User)	Display the membership page
	void logOut(MouseEvent)	Log out the current page
ChangePassword Controlller	void onChangePassword ConfirmClicked()	Handle the event of clicking password confirm button
	boolean ifUserExist (String, String)	Check the existence of the user
	void changePassword (String, String, String)	Change the password for the user
	void gotoSuccessfulPage()	Jump to the successful page
	void onCancelClicked (MouseEvent)	Handle the event of clicking cancel button
LoginController	void onCreateClicked (MouseEvent)	Handle the event of clicking create button
	boolean ifUserExist (String,String)	Check the existence of the user
	boolean ifTrainerExist (String ,String)	Check the existence of the trainer
	void trainerLoginButtonClick()	Handle the event that trainer clicks login button
	void loginButtonClick()	Handle the event of clicking login button
	void gotoWelcome (User)	Jump to the welcome page
	void onResetClicked (MouseEvent)	Handle the event of clicking reset button
SIgnUpController	void onCancelClicked()	Handle the event of clicking cancel button
	void onSignUpButtonClicked (MouseEvent)	Handle the event of clicking sign up button
	void gotoSignUpSucceed (String, String)	Jump to the successful page
	boolean checkInput (String, String)	Checkin the user inputs
	void onUsernameInput (KeyEvent)	Handle the event of inputting user name
	void onPasswordInput (KeyEvent)	Handle the event of inputting password
	void onTelInput (KeyEvent)	Handle the event of inputting telephone
	void onEmailInput (KeyEvent)	Handle the event of inputting email

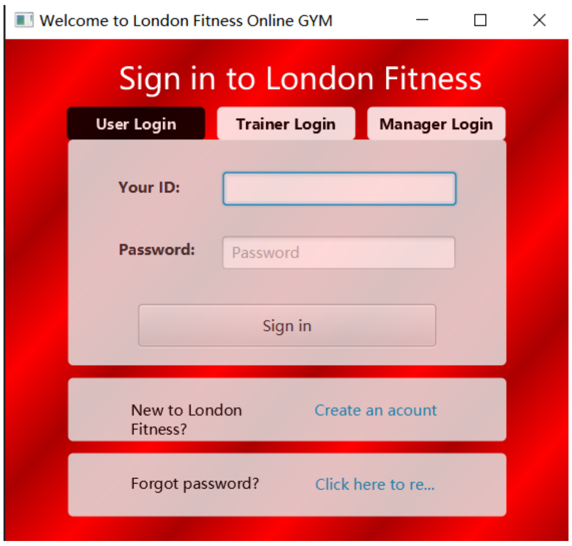
	void onRealNameInput(KeyEvent)	Handle the event of inputting realname
BookRequest Controller	void returnToBookingPage (MouseEvent)	Handle the event of clicking return button
	confirmBooking (MouseEvent)	Handle the event of clicking confirm button
InformationRevision Controller	void setPerson(User)	Set the particular person
	boolean isOkClicked()	Judge if the ok button is clicked
	void handleOk()	Handle the event of clicking OK button
	void handleCancel()	Handle the event of clicking cancel button
	boolean isInputValid()	Check if the input is valid
LiveController	void showTable()	Show the table of the interface
	void setDateForTable()	Set the date field in the table
	String getDate()	Get the date
	void loadTrainerInfo(int)	Load the trainer information
	void loadBookInfoTable()	Load the booking information
	void changeTimetable(int)	Change the timetable
	void NextButtonClicked (MouseEvent)	Handle the event of clicking Next button
	void formerButtonClicked (MouseEvent)	Handle the event of clicking Former button
	void goToNextDay (MouseEvent)	Handle the event of clicking Next date button
	void goToFormerDay (MouseEvent)	Handle the event of clicking Former date button
	void onBookButtonClicked (MouseEvent)	Handle the event of clicking Book button
	void handleUpdate()	Handle the event of update interface
	void updateTrainer BookingInfo()	Update the booking information of the trainer
	void updateMyBookingInfo()	Update the booking information of the user
	void onDeleteButtonClicked (MouseEvent)	Handle the event of clicking Delete button
MembershipController	void showList()	Show the list in the interface
	void upgrade(String)	Upgrade information based on the json file
	void VideoManageLinkClick()	Handle the event of clicking Video Manage link
	void ManagerLinkClick()	Handle the event of clicking

		Manager link
playController	String DurationToString (Duration)	Convert the time to String
	void start(Stage, String)	Start to play the video
TrainerLiveController	void checkRequest()	Check the booking request
	void loadBookInfoTable()	Load booking information table
PurchaseMembership Controller	void setPerson(User)	Set the particular person
	void handlePurchase1()	Handle the event of clicking purchase 1 month
	void handlePurchase2()	Handle the event of clicking purchase 3 months
	void handlePurchase3()	Handle the event of clicking purchase 1 year
	boolean isButtonClicked()	Check if any button is clicked
RechargeController	void setPerson(User)	Set the particular person
	Void handleRecharge1()	Handle the event of clicking recharge 100 dollars
	Void handleRecharge2()	Handle the event of clicking recharge 200 dollars
	void handleRecharge3()	Handle the event of clicking recharge 500 dollars
	boolean isButtonClicked()	Check if any button is clicked
StrengthController	void select1()	Select hiit level 1
	void select2()	Select hiit level 2
	void select3()	Select hiit level 3
	void back(MouseEvent)	Handle the event of clicking back button
	void refresh(MouseEvent)	Handle the event of clicking refresh button
HIITController	void select1()	Select hiit level 1
	void select2()	Select hiit level 2
	void select3()	Select hiit level 3
	void back(MouseEvent)	Handle the event of clicking back button
	void refresh(MouseEvent)	Handle the event of clicking refresh button
Controller	void setUser(User)	Set the particular user
	void setStage(Stage)	Set the stage
	void changed(ObservableValue <? extends Number>, Number , Number>	Change the size of interface
	void onVideoButtonClick	Handle the event of clicking video

	(MouseEvent)	button
	void onLiveButtonClick (MouseEvent)	Handle the event of clicking live button
	void onMemberButtonClick (MouseEvent)	Handle the event of clicking membership button
	void onHomeButtonClick (MouseEvent)	Handle the event of clicking home button
	void onUserClick (MouseEvent)	Handle the event of clicking user button
	void onVideoMouseEntered (MouseEvent)	Handle the event of entering video button
	void onVideoMouseExited (MouseEvent)	Handle the event of exiting video button
	void onLiveMouseEntered (MouseEvent)	Handle the event of entering live button
	void onLiveMouseExited (MouseEvent)	Handle the event of exiting live button
	void onHomeMouseEntered (MouseEvent)	Handle the event of entering home button
	void onHomeMouseExited (MouseEvent)	Handle the event of exiting home button
	void onMembershipMouseEntered (MouseEvent)	Handle the event of entering membership button
	void onMembershipMouseExited (MouseEvent)	Handle the event of exiting membership button
	Initializable replaceSceneContent (String)	Replace the content of the scene

6.4 Screenshots

Sign in interface:


Welcome to London Fitness Online GYM

Sign in to London Fitness

User Login | Trainer Login | Manager Login

Your ID:

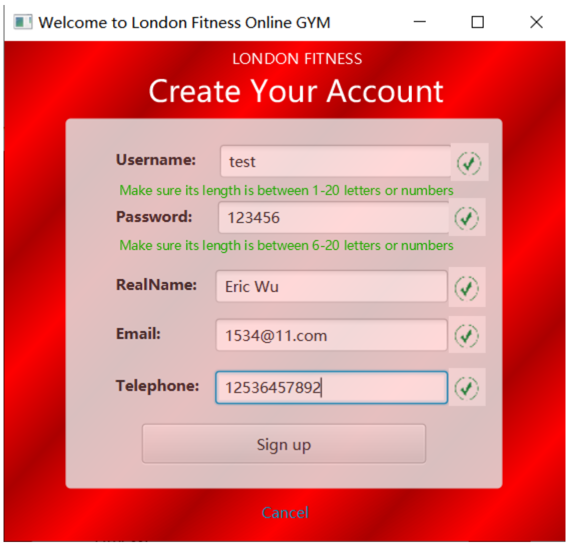
Password: Password

Sign in

New to London Fitness? Create an acount

Forgot password? Click here to re...

Sign up interface:


Welcome to London Fitness Online GYM

LONDON FITNESS

Create Your Account

Username: test

Make sure its length is between 1-20 letters or numbers

Password: 123456

Make sure its length is between 6-20 letters or numbers

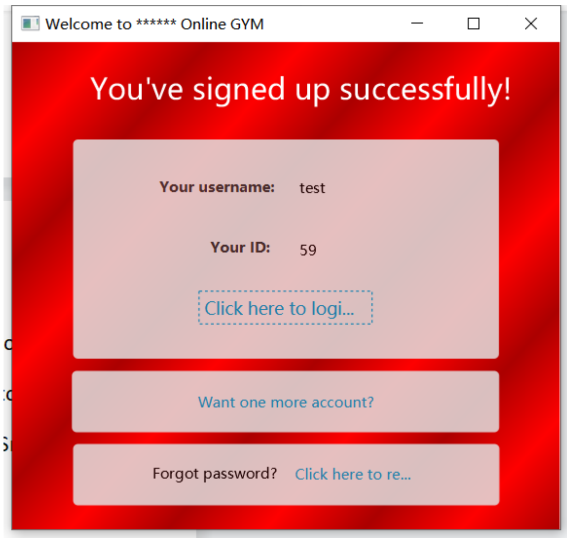
RealName: Eric Wu

Email: 1534@11.com

Telephone: 12536457892

Sign up

Cancel


Welcome to ****** Online GYM

You've signed up successfully!

Your username: test

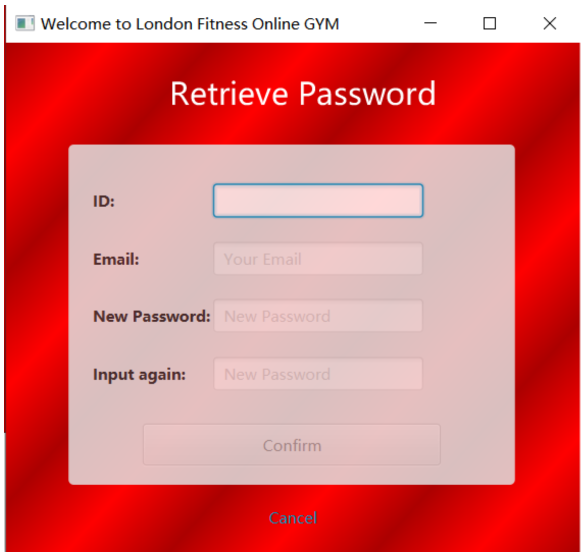
Your ID: 59

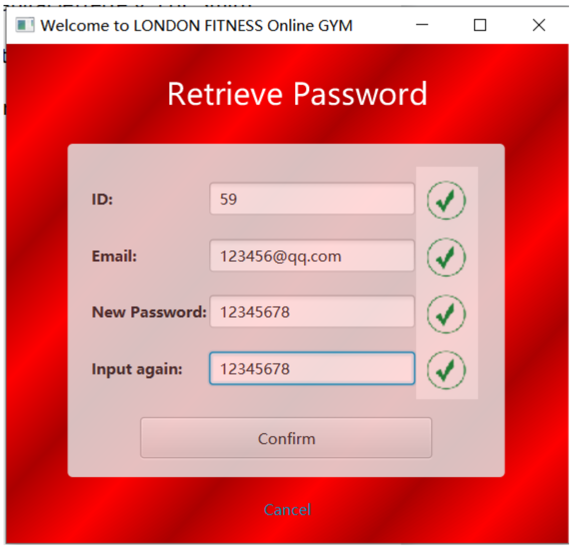
Click here to logi...

Want one more account?

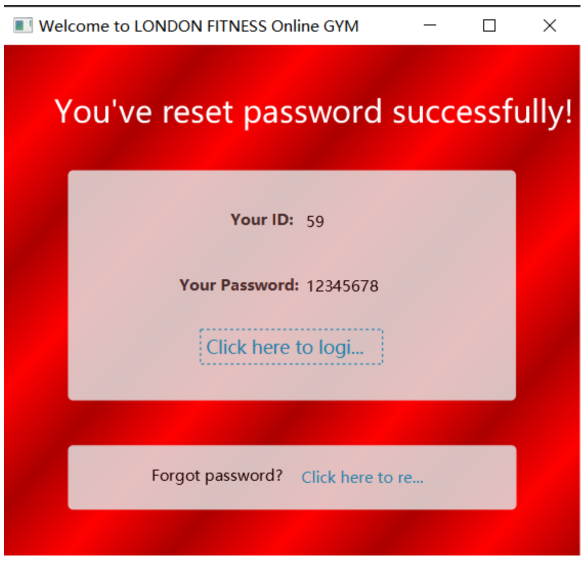
Forgot password? Click here to re...

Password retrieval interface:





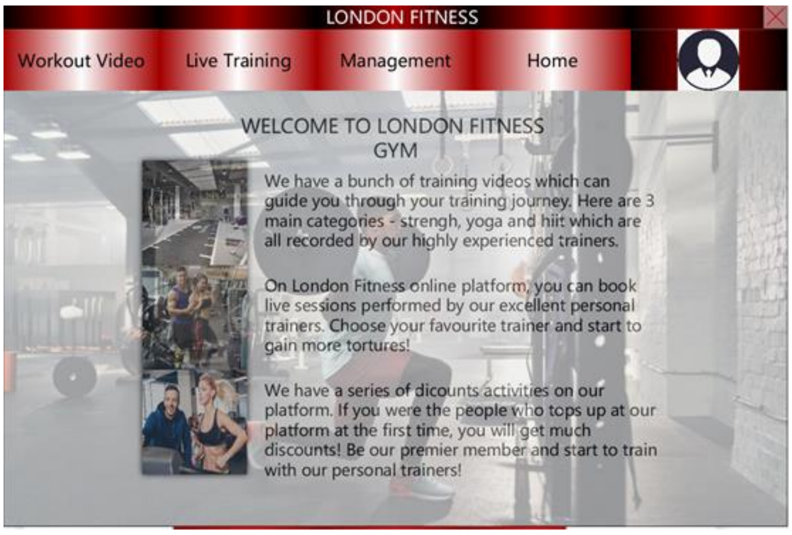
ID: 59
Email: 123456@qq.com
New Password: 12345678
Input again: 12345678


Welcome to LONDON FITNESS Online GYM
You've reset password successfully!
Click here to logi...
Forgot password?
Click here to re...

Your ID: 59
Your Password: 12345678

Group 9

Welcome interface:



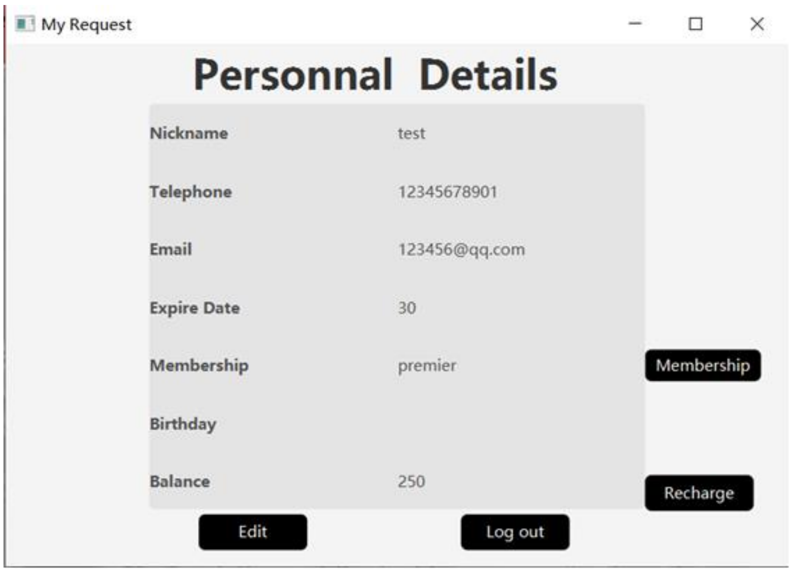
WELCOME TO LONDON FITNESS GYM

We have a bunch of training videos which can guide you through your training journey. Here are 3 main categories - strengh, yoga and hiit which are all recorded by our highly experienced trainers.

On London Fitness online platform, you can book live sessions performed by our excellent personal trainers. Choose your favourite trainer and start to gain more tortures!

We have a series of dicounts activities on our platform. If you were the people who tops up at our platform at the first time, you will get much discounts! Be our premier member and start to train with our personal trainers!

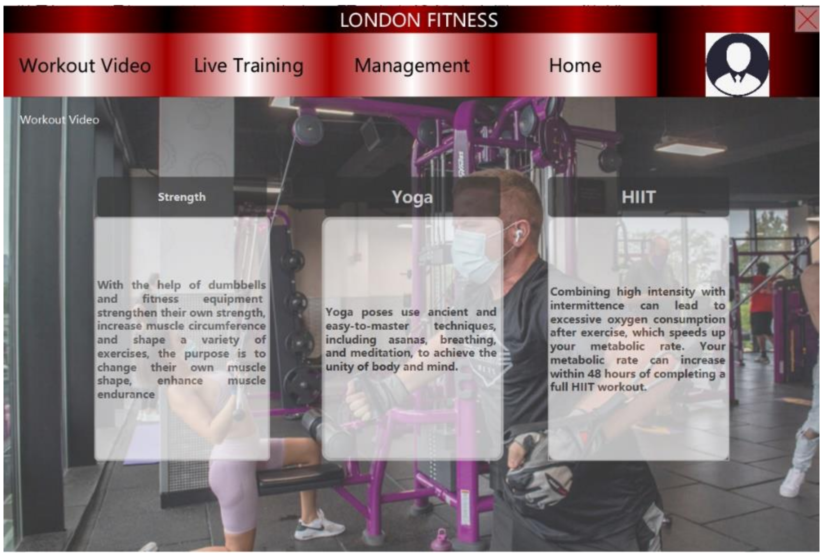
Personal information interface:



Personnal Details

Nickname	test
Telephone	12345678901
Email	123456@qq.com
Expire Date	30
Membership	premier
Birthday	
Balance	250

Workout video interface:



Strength

With the help of dumbbells and fitness equipment strengthen their own strength, increase muscle circumference and shape a variety of exercises, the purpose is to change their own muscle shape, enhance muscle endurance

Yoga

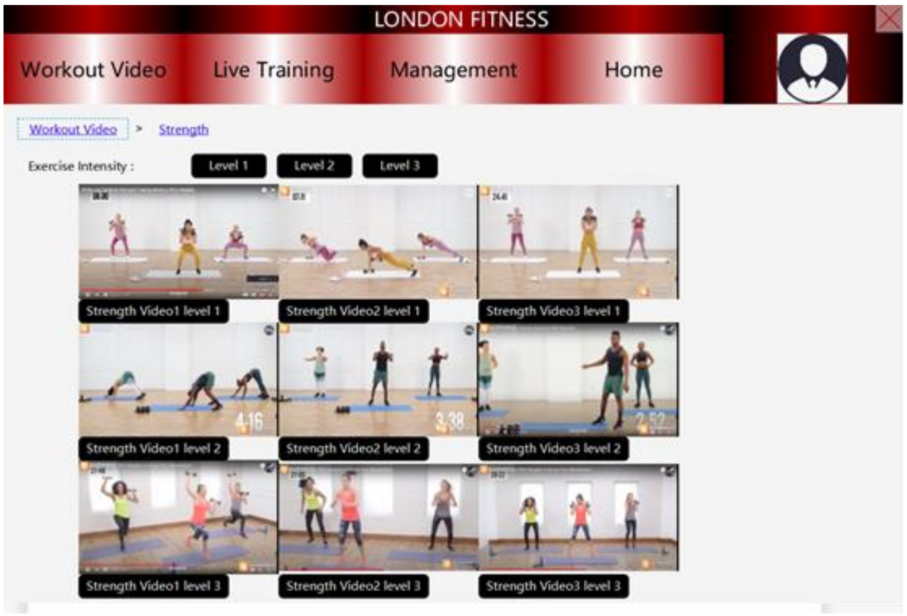
Yoga poses use ancient and easy-to-master techniques, including asanas, breathing, and meditation, to achieve the unity of body and mind.

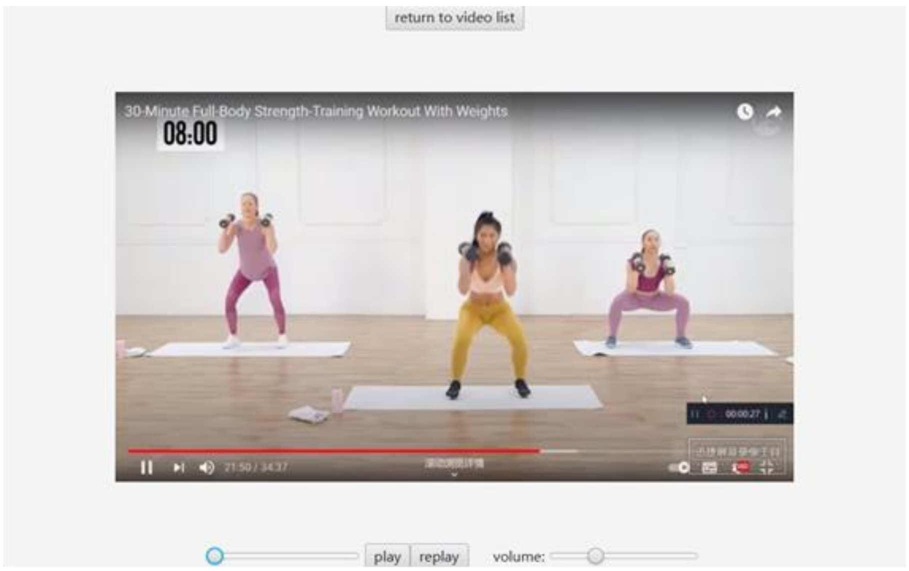
HIIT

Combining high intensity with intermittence can lead to excessive oxygen consumption after exercise, which speeds up your metabolic rate. Your metabolic rate can increase within 48 hours of completing a full HIIT workout.

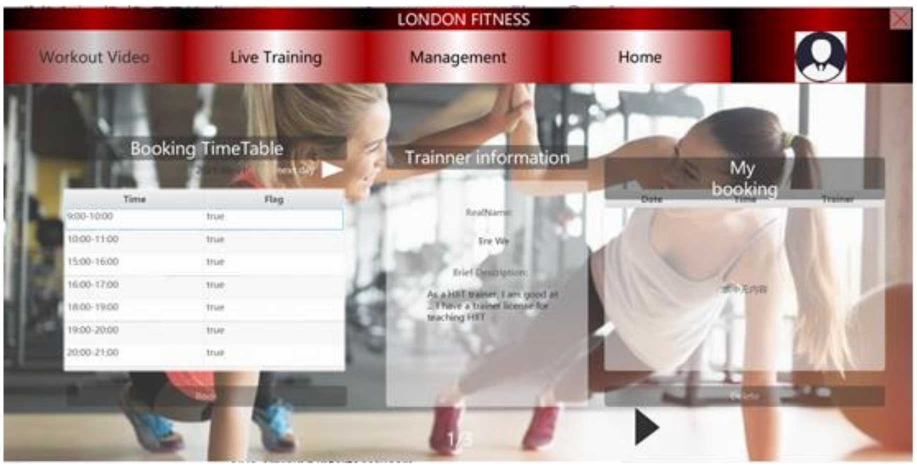
Group 9

Video list interface:



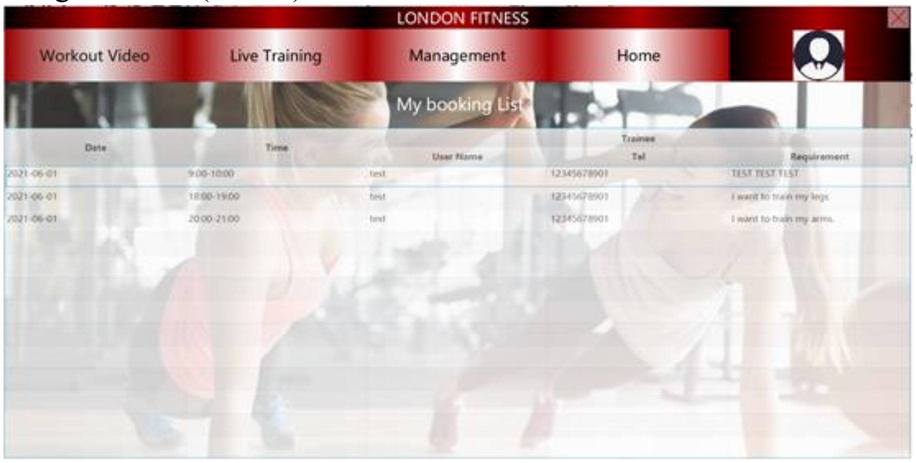


Live training interface (user):



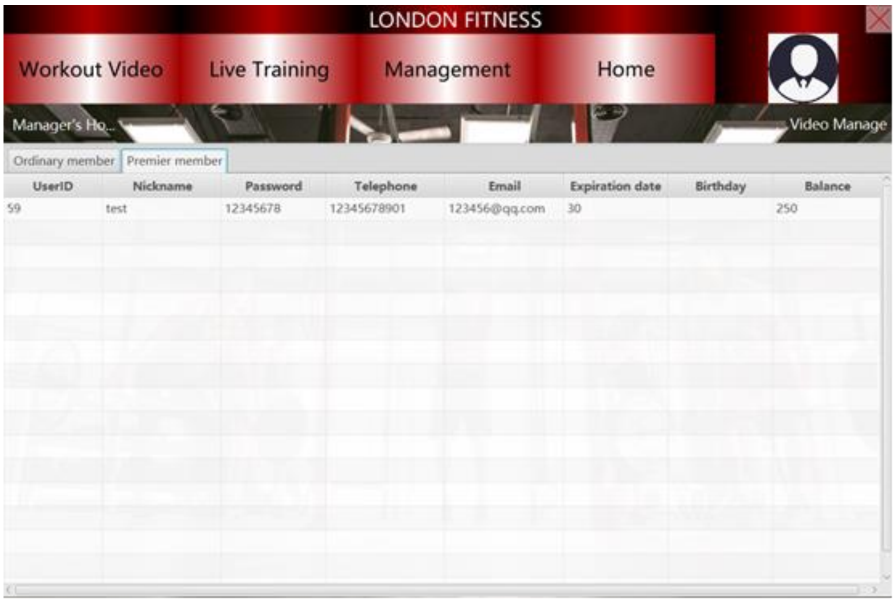
Time	Flag
9:00-10:00	true
10:00-11:00	true
15:00-16:00	true
16:00-17:00	true
18:00-19:00	true
19:00-20:00	true
20:00-21:00	true

Live training interface (trainer):



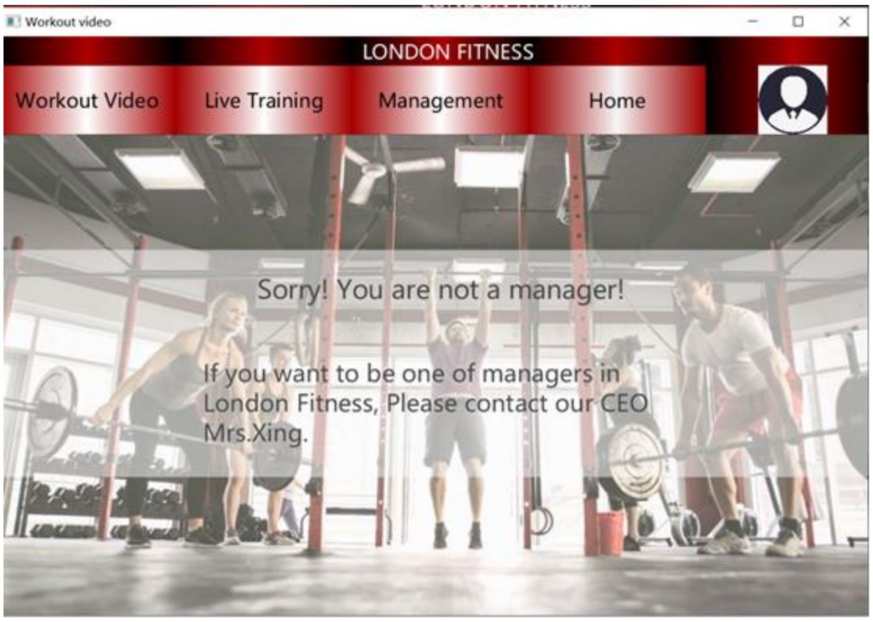
		Trainee		
Date	Time	User Name	Tel	Requirement
2021-06-01	9:00-10:00	test	12345678901	TEST TEST TEST
2021-06-01	18:00-19:00	test	12345678901	I want to train my legs
2021-06-01	20:00-21:00	test	12345678901	I want to train my arms.

Membership management interface:



UserID	Nickname	Password	Telephone	Email	Expiration date	Birthday	Balance
59	test	12345678	12345678901	123456@qq.com	30		250

Administrator interface when not administrator:

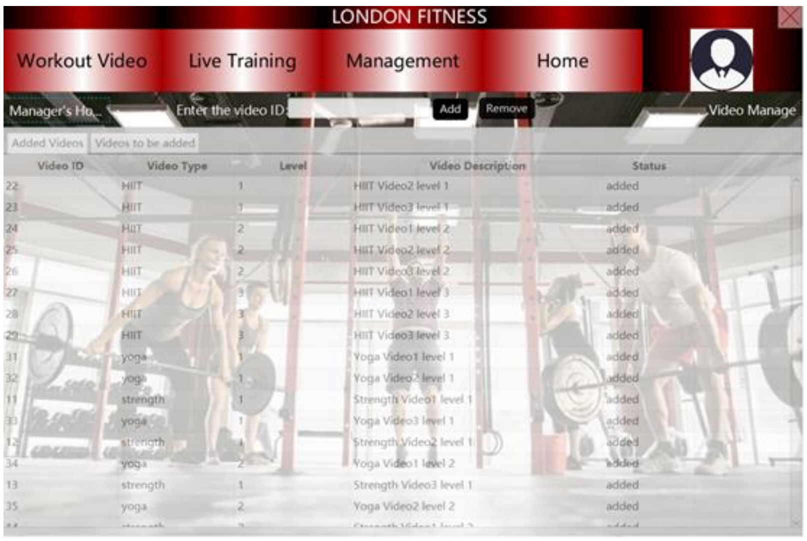


Sorry! You are not a manager!

If you want to be one of managers in London Fitness, Please contact our CEO Mrs.Xing.

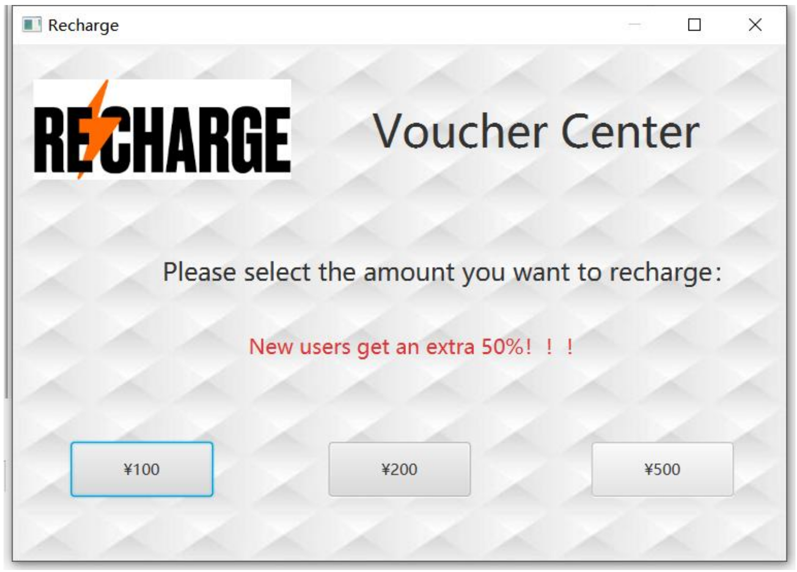
Group 9

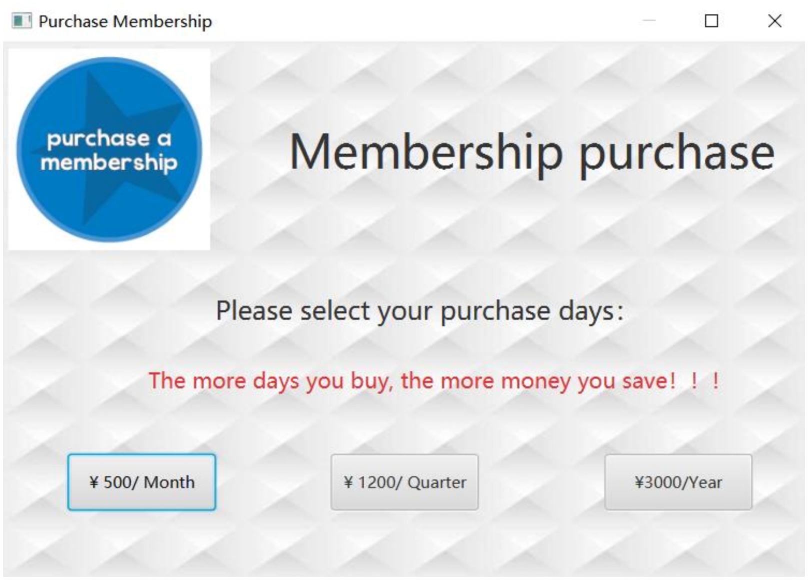
Video management interface:



Video ID	Video Type	Level	Video Description	Status
22	HIIT	1	HIIT Video2 level 1	added
23	HIIT	1	HIIT Video3 level 1	added
24	HIIT	2	HIIT Video1 level 2	added
25	HIIT	2	HIIT Video2 level 2	added
26	HIIT	2	HIIT Video3 level 2	added
27	HIIT	3	HIIT Video1 level 3	added
28	HIIT	3	HIIT Video2 level 3	added
29	HIIT	3	HIIT Video3 level 3	added
31	yoga	1	Yoga Video1 level 1	added
32	yoga	1	Yoga Video2 level 1	added
11	strength	1	Strength Video1 level 1	added
33	yoga	1	Yoga Video3 level 1	added
12	strength	1	Strength Video2 level 1	added
34	yoga	2	Yoga Video1 level 2	added
13	strength	1	Strength Video3 level 1	added
35	yoga	2	Yoga Video2 level 2	added

Recharge interface:


Recharge
RECHARGE
Voucher Center
Please select the amount you want to recharge:
New users get an extra 50%! ! !
¥100
¥200
¥500


Purchase Membership
purchase a membership
Membership purchase
Please select your purchase days:
The more days you buy, the more money you save! ! !
¥ 500/ Month
¥ 1200/ Quarter
¥3000/Year