**Introduction**

Sports can relax people’s body and make ourselves strong and healthy. It is one of the most popular ways to relax and exercise our body ever since. However, people are often troubled by many special situations. They always cannot have enough people to do sports together. The place of doing sports will also be a big problem. Additionally, when some people are late, they do not have a proper way to solve this problem. In this case, it is crucial to use a project to make people go together and do team sports easily, and encourage people to go out and exercise more. We are required to build an easily used system to make people do sports together more efficiently.

**Background**

To achieve the aim of this project, we divided the project into two parts: 1. Do research to know what users really want and design our system 2. Use better language and framework to finish the program in the first part, we ask many people of what they want when they want to order place and do sports together. Importantly, the research that we do give us a way to design our program. Then we can change the functions that users want to use. Finally, we get the proper design decision to achieve the final aim. In the second part, we search for the better language and framework to program in a proper way. By the recommendation of supervisor and searching on the internet, we finally decide the language and framework to finish our work in programming. After these two parts, the total plan and schedule of this project are decided.

By the finish of these two parts, the program gives the following functions: it lets users to register when users want to organize the events. Then users can organize their events when the give the detailed information. If users want to cancel it, all users who attend in this event will receive a message about it. For those who are often late or absent, the program will give punishment to them. Then we attempt to adjust our program and apply it to the project which are often seen in the application of sports website. We also consider to add new functions in future when users need.

**Research**

For the research part which can decide some key decisions, we divided into five parts: programming languages, operating systems, schedule design, computer or mobiles and software/framework. To decide what kind of research to do in each part, the first we should decide will be the type of this application. At the beginning of it, we consider to design both on website and application.

When the website and application are connected, it is convenient to control all the project in a proper way. Then we use the right language to make the website connect to our first version of the program. When the first version is completed, the best platform is founded to design our project. Django and Agile Development are two suitable methods to adjust the schedule and project in the later progress.

**Requirement Analysis**

For the requirement that need to be designed, in the research part there are mainly five parts: programming languages, operating systems, schedule design, computer or mobiles and software/framework.

For the programming languages, the initial types are PHP and html. Then python will then be used in the designing and code removing. The languages about WeChat designing will also be used in the next stage. In other parts of new type designing, new languages may be used when they are needed.

For operating systems, the mobiles will be used to design and change the code of this program, then these will return to mobile phones which can be used easily. In all program, both mobile phones and laptops will be used to change or design the program in different ways.

For schedule design, it can be divided into two parts: the first part is the schedule of key points, such as the code finishing, the UI design and the website design. The second part is small schedule, such as the work of group members and the meeting record. As a result, in the whole project, we use Agile Development. Agile Development is a kind of efficient and flexible method when the project is designing. In later programming and requirements, it can adapt to possible different situations. For the whole schedule, it can be adjusted when it is needed.

For computer and mobiles, the program will finally be used on mobiles, the computer will only receive some useful informal that can help to control. On mobiles, it will not show as application. Instead of this, it will be show like a mini program on WeChat. For most of people today, they always want more efficient and convenient service to make their life do not have too many troubles. Instead of applications which will make them to download and install them and also have problems while using them, mini application will be more suitable for people who just would like to attend a sport event or organize a sport event. People who always use WeChat can easily open a website or a mini program to achieve this function. This is the reason why the whole project will design a mini program but not a kind of application.

For software/frame work to be used, the supervisor gives some important idea about how to choose the best one for the program design. In the future, the best software/frame will best choose in a proper way.

After finishing the research of these parts, we found several familiar apps and in this special part we choose 2 of them (TeamSnap and Teamer) to analysis market and location of our application so that they can make us use some functions to achieve the aim and design what users really want.

# Software Design

**Database Design:**

For the database design, we made a few updates.

1. First, we turned to use built-in table for users in Django to model users in our database. The benefit of this is obvious: we can use the built-in attributes instead of designing on our own. Besides, many built-in functions and parameters fetch data directly from the built-in user table, which provides us much convenience when implementing our functions.
2. Second, we added two attributes to table ‘Event’, ‘date\_added’ and owner, which shows when and who the event was created. In the previous version, we only focused on the details of an event and didn’t have these two attributes. Because at that time we hadn’t started to implement interaction part and events are separated from other objects. But in the latest version, users need to interact with events. So it’s necessary to allow users to know who and when the event was created as this can help them decide whether to attend the event.
3. Third, we changed the previous “join\_state” table to a new table “Participant”. Previously, we kept the data of each pair (event\_id, user\_id) and stored their join\_states (i.e. Yes/No). If there exists m events and n users in our system, then there should be m\*n entries in the previous table. This is very space-consuming, so we modified the database design. We only store the pair (event\_id, user\_id) in which the user has joined the event. Since we only store “joined”, there’s no need to keep the ‘join\_state’ attribute. Instead, we added a ‘date\_added’ attribute which records when the user joined the event.
4. Fourth, we added a new table “Profiles” to our database, which stores personal information of users. This is natural in all software related to user account. It allows users to present personal details that they are willing to show to others.

**Architecture Design:**

Since our software aims to help users organize/join team sports, we divided it into two parts: Users and Events, which adheres to Object Oriented design principles. Although we didn’t use Object Oriented programming language for implementation, we did use OO principles to design the software architecture. With the guidance of OO principles, Users part is further divided into two parts: account and profiles. We made this design because account and profiles are two common attributes of users, it seems rational and natural to include them. In addition, Events part is extended to: add new events, view events, join in an event, exit from an event, edit events. This is because these are common methods related to an event.

**Functionality Design:**

**1.** As mentioned in Architecture Design section, Users should have two attributes: account and profiles.

**(1)**” Account” is represented as username and password in our database. Accordingly, we designed **Log in/Log out** function. This is to ensure only valid users have access to the system. According to our stakeholders, our system will be used to organize events among their friends. This means strangers should not have access to the events list and valid users’ personal information. Therefore, it’s necessary to add the Log in/Log out function.

**(2)** “Profiles” is represented by a separate table in our database and a foreign key is added to connect profiles and users. The function related to it is **My Profiles**, in which users can view and edit his/her personal details. These details are open to other valid users of the system. Because users may need to check personal information of other participants of an event to decide whether to join in this event.

**2.** Again as mentioned in Architectural Design section, Events should contain the following methods: add new events, view events, join in an event, exit from an event, edit events. Among these functions,

**(1)**. Add new events function is designed for organizer. An organizer is asked to fill in and submit an event form, then the new event will be stored in database and be shown in the event list. This function aims to help organizer publish an event so that other users can join in it. It supports organizing a team event online without the need of meeting and conversation.

**(2)**. Besides, user should be able to **view current published events**. Actually, an event list is designed to be shown on the home page of our website because it’s related to the core function of our website. Users should be able to check details of their interested events to decide whether to join in. But there should not be too much information on the home page. Therefore, according to our design, only abstract of events is shown in the events list on the home page. To view details, they can click on an event to go to the detail page of it, in which all information of this event is presented including details of the event and its participants.

**(3)**. Then, if users find an appropriate event, they should be able to join in it and they will appear in the participants list of this event. Similarly, if they no longer want to attend this event, they should be able to exit from it and their names should be deleted from the participants list. Thus, two more functions **join in an event** and **exit from an event** are designed.

**(4)**. Finally, considering that an organizer may need to change the ‘settings’ of an event, we added an ‘**edit events**’ function to deal with it. Such condition may occur when the organizer wants to change the time of an event due to emergency, or the organizer needs to change the player limit of the event due to the limitation of facilities. In these cases, it’s necessary to edit the settings of an event. However, this function should be only open to the organizer otherwise it will lead to mess because everyone can attempt to change the event.

**Software Frame-Work**

**Why use PHP?**

Last semester, our team have confirmed the system requirements, we make a crucial formal meeting about development language, development framework, establishment of database, user management and some other aspect. Because our production must have characteristic of time-effectiveness and convenient for user, the original design of our system is that develop a Web application to satisfy the requirements. But according to the widespread usage of mobile device, our system must have the transferability to mobile device like iPhone. So our team choose PHP as the original language to develop the system. First of all, PHP has a very good cross-platform nature. It supports almost all operating system platforms and database systems. This characteristic benefits our team that makes creating, using and maintaining database easily and when our team members develop software on different operating system, it is much easy to work parallel and merge our progress. Secondly, PHP as a strongly scripting language, it grammar combines C, Java, Perl and PHP new grammar, have the ability to execute web pages extremely quickly. Moreover, the ability of expand of PHP satisfies the requirement of the system which need augment-ability and portability. Besides, PHP has recognized great security performance. This ensure a mature, stable and safe system. Then for the whole system, our team use HTML as an interactive page, use MySQL: a traditional, easy-to-use, simple configuration tool, to create the database. This is the original prototype of our system.

**Why not use PHP?**

When our team begin to build the system, improve function of it, we find some problems. First, as the number of HTML files increase, we have to write or copy some pieces of code in each file. This means that our team actually do lots of duplicate work and waste time. Secondly, we always spend 80-percent of time to find some pieces of code rather modify the code, this is also due to the code disorder. Our team have to suspend development and discuss about our original system plan. We found that we do not consider a clear logic and architecture to implement our system, and this problem not only affect our development progress but affect the later maintenance of the system. In other word, the original plan is bad. We need a modular, well-organized framework to help our team make a clear framework structure. Finally, we change to use Python.

**Why use Python?**

After the completion of the first version of our application, we have noticed several problems occurred in it. The PHP version between different operating system appears different. Also, due to the OS difference, we have developed the first version of our program using different hierarchy and patterns, which causes the first version of our application a big mass. This problem makes the further development and maintenance of this application obscure and complicated. The code made this application might be changed tens of thousands of time to fit the requirement of our customers and the situation we faced at that time cannot meet our needs. So, we turn into the python solution and use a framework named Django to solve these problems. Compare to PHP, Python is more flexible, more adaptable, more portable and more readable. PHP has a lot of linguistic laws and only after plenty of time to study can we implement the application properly, where Python has less linguistic restrictions and easy to make the application lightweight. Python is more readable, for it uses indent as the signal to divide different hierarchy of code and more similar to natural language. Python has good expandability and embed-ability, which means we can use other excellent languages like JavaScript to implement some concrete parts of our application to avoid some shortcoming of Python itself like low speed. Since we have used the Agile development pattern, we need a more flexible language to rewrite our application and python is well satisfies our requirement because of the abundant development kit and its ample development library.

**Why using Django?**

As is mentioned in the PHP part, we have faced one crucial obstacle in developing our application in a standard and clear way. The architecture of the application needs to be settled down and clarified by using the framework. However, there exist hundreds of frameworks on the internet for developing a web application using Python. After research, we choose four main frameworks which are Django, Pylons, Tornado and Bottle. Bottle is lightweight but we need to find other third-party libraries to implement our application. Tornado is fast and simple but is it hard to encapsulate functionality modules. Pylons is flexible and powerful, but it takes a lot of time to study and it’s hard for newcomers. While Django is not as excellent in some features like other frameworks, but it is balanced and easy in use. The full MVC pattern and modular hierarchy divide provided by Django makes the architecture clear and stable. The luxuriant library contained in Django provides enough support in developing the whole application. Django is flexible enough and easy to use, it takes less time between study and finally implement codes using Django. It also provides good support to different platforms which means that we don’t need more work in adopting different operating systems. Although it is not that lightweight and powerful compared other Python web application frameworks, it is most suitable for us newcomers to develop the GoTogether. So we choose Django as the framework of our web application.

7.3 Distributed Version Control System -Git

In fact, after our team started to write code, we had a lot of trouble when our respective responsibilities were to be integrated. However, this has improved a lot since we started using GIT and remote repository.

7.3.1 Not adopted at the beginning

There are several main reasons why we didn't use GIT and remote repository at first:

1. **Learning time is short**. We only officially started learning git in the last course. Maybe you just used it to clone code.
2. **This is the first time to start a team project.** We all think that as long as we take charge of our own part, the project will be successful.
3. **The secrecy of the remote repository is poor**. The Remote repository is public, while the private remote repository is not cheap.

Therefore, under these factors, we did not use GIT and remote repository at the beginning.

7.3.2 Problem encountered

As I mentioned at the beginning, we encountered some problems in the integration process, which made it inefficient to write code at the beginning. For example:

1. **One person is needed to integrate everyone's work**. In fact, because 4-5 people have different code styles, especially when code conflicts occur, the difficulty of integration is greatly increased. The team member who responsible for integration often spend around the same as coding time to complete the work. At the same time, other members were unable to continue their work.
2. **When making modifications on the basis of others**, it is often necessary to understand others’ code beforehand. Increased personal working hours, delayed the integration point.

7.3.3 reason for using Git

Our inefficiency has prevented us from achieving some of the goals we set for the week on several occasions. The team began to turn its attention to GIT and remote repository.

The reasons for our switch to GIT and remote repository are as follows:

1. **It is more convenient to get the working source code**. Each version can be uploaded to a remote repository and retrieved. Instead of making a compressed file for each new version, and transmitted through WeChat.
2. **Conflicts can be easily resolved** between any two developers. Instead of leaving them to the member who responsible for integration to deal with all conflicts. And when conflicts are resolved locally, project integration is not a problem.
3. **We learn very fast**. Since the GIT operations needed to complete our work are not complicated.
4. **There is no need for high confidentiality** because of not much commercialization involved.

After starting to use git, though we may not be familiar with it at first, and our efficiency has not improved significantly, the original integration problems begin to decrease.

8.UI design

8.1 Design Idea

Because it is an app for information services, we want to use simple steps and guidance to enable users to complete a complete experience. The main design ideas are:

* Simple and clear layout
* Relaxing content, such as background and tone, etc.



Figure 8. 1 Montserrat Font

8.2 Overall layout

8.2.1 Basic elements

|  |  |  |
| --- | --- | --- |
| **Section** | **Choice** | **Reason** |
| Tone | blue and light blue | 1. majority of people like blue 2. Makes the product seem trustworthy 3. Friendly to colorblindness[[1]](#footnote-1) |
| Background | Blue background image |
| Font | Montserrat | The default font was abandoned because Montserrat belongs to Sans serif. The corners are smoother and more delicate. It's more eye-catching and comfortable in tables and lists.[[2]](#footnote-2) |

8.2.2 Pages

Because we **divide** each **page** into **base and** **content** sections. The **Base** is the same for each page. It reflects the overall layout of the page.

The Benefit of doing so is:

* Reduces code redundancy. Because each page extends the base page. We don't have to rewrite the same head and tail. Instead, you just need to care about the content part.

**8.2.2.1 Base Page**

In our project, there ‘are two part in the base section:

1. a **top-fixed** **collapse** navigation bar. It contains the basic operations that all users encounter. For example, registration, login, login, etc.
2. **global message alert box**

The **purpose** of set navigation bar:

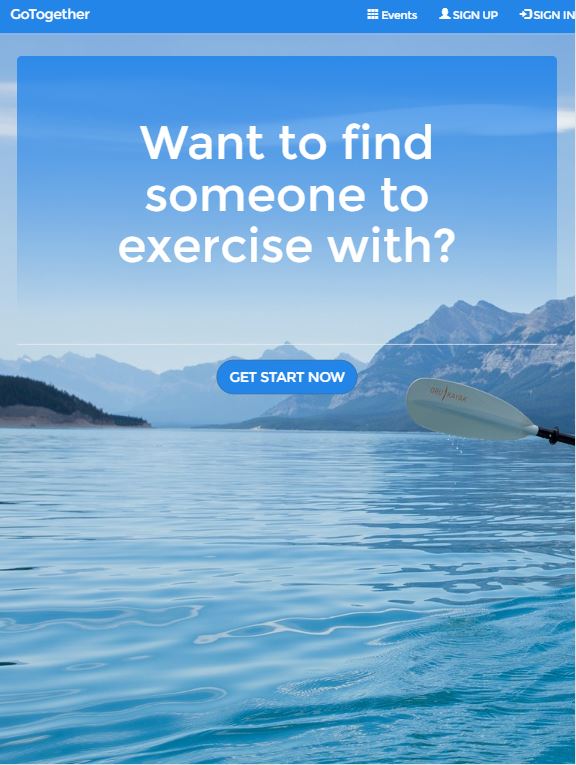


Figure 8. 3 Navbar on iPad

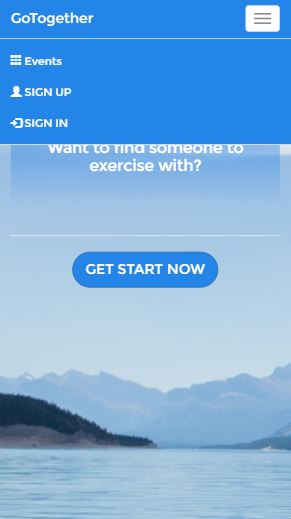


Figure 8. 2 Navbar on iPhone 8

|  |  |
| --- | --- |
| top-fixed | To make it **easy** for users to do some basic operations at any time while browsing. For example, return to the home page. |
| Set navigation bar  Collapsed | Designed to **respond** to the small screen size of mobile devices. On a tablet, it does not collapse. |

The global message reminder box is also designed to **reduce code redundancy**. It is set to hide under the navigation bar. When there is a message reminder (success, error, etc.), it will appear.

**8.2.2.2 Index page**

As You can see the index page from figure 8.3. It looks very simple, contains only a big slogan and a button to guide users to start using it. Perhaps the current slogan is not perfect, we followed the simple and clear point in design idea.

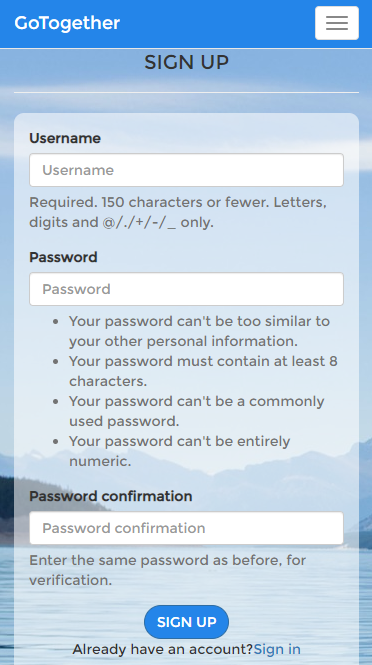


Figure 8. 4 Sign Up

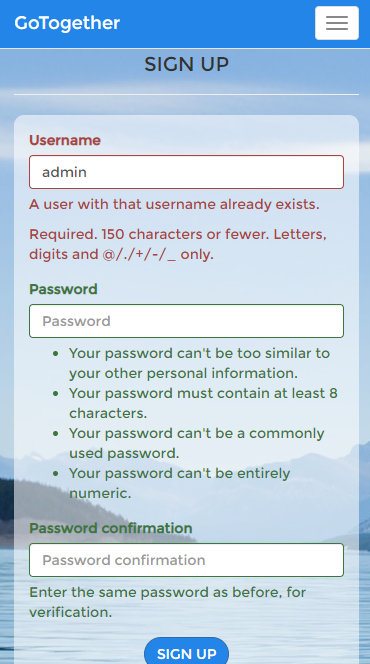


Figure 8. 5 Username existed

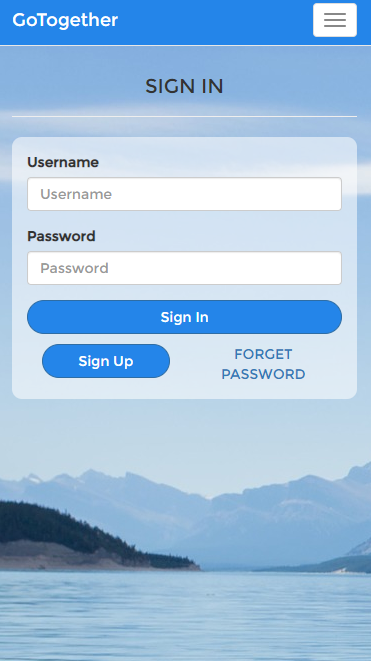


Figure 8. 6 Sign In

**8.2.2.3 Sign Up page**

Sign Up consists of two parts:

A **registration form** and a **hyperlink to the sign in page** (figure 8.4).

|  |  |
| --- | --- |
| **Part** | **Purpose** |
| Registration Form | provides detailed registration **requirements** for username and password. **Form Validation** is also implemented. For example, when a user name has been registered (Figure 8.5). |
| Hyperlink | designed to **prevent** registered users from entering the registration page by mistake. |

**8.2.2.4 Sign In Page**

Sign in page consists of three parts:

A login form, a button to redirect to the signup page, and a hyperlink to recover the password (Figure 8.6).

|  |  |
| --- | --- |
| **Part** | **Purpose** |
| login form | Implementing password verification |
| Redirection button | Provide convenience for unregistered users to enter the login page by mistake |
| Hyperlink | Necessary functionality for users who forget their passwords |

**8.2.2.5 Events page**

The Events Page is used to show user events that are currently available (Figure 8.7).

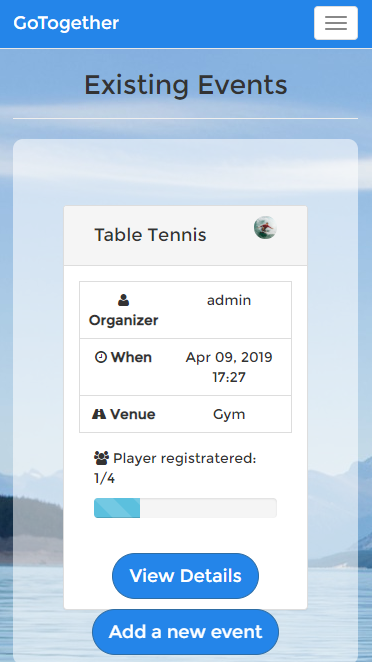


Figure 8. 7 Events

It consists of two main parts, Brief Event presentation, and Event Add Button.

|  |  |  |
| --- | --- | --- |
| **Section** | **Components used** | **Purpose** |
| Brief Event List | Panel | To better distinguish event name from event details |
| Table | To introduce events concisely and systematically |
| Progress Bar | In order to visualize the number of people and make it more intuitive. |
| Add Event Button | Fixed-bottom navbar | In order to make it easier for users to add new events whenever they want while browsing without having to go back to the top of the page. |

**Function Implementation**

Based on the database design, the functionalities can be divided into four main parts: Register, Create Events, Join Events and Modify the Event. In implementing the Register functionality, we used the built-in user library inside the Django itself. Since the **Django has contained the user part and the attached user operate functions meet our needs well,** it is no need to create new model named user to implement user part. Using the build-in part, we have allocated the authority of adding and modifying events to users to assist users interact with events.

One important role in our application is **Event**. In implementing the Create, Join and Modify Events part, according to the object-oriented programming, we set the functionalities focus on the event object. Since this application is based on the event, we want to build one event with all operations inside it, and by allocate different authorities to different kind of users to operate events in different stages. Another thing we want to achieve is to let our application easy to maintain and further developing, we use the MVC pattern to design the event part as well as possible. So, in the final implementation, we divided the form, view and model to make our application clean, we use the templates to assist view functions and make the hierarchy more clearly. Considered the individuation, besides the time, venue, and requirements as the basic information to attend one event, we have added one attribute named description for the event creators to describe their events more freely. And by using the template to manage our different form, the efficiency in create the event will increase, for the activity that inform others all the details and gathering them have been transformed into filling the forms and click the join button.

Another important role in our application is the **user**. The main problems we want to solve in our application is focus on the interaction between user and events and the relationship between user and user. So, we encapsulate related user functionalities inside the user application. Since we want more user play with each other, we don’t use the most common “friend” functionality nowadays, rather then, we use the event to connect users. Also based on this thought, users will have its own profiles to show their basic information to others who attend the same event. Thus, in implementing the user part, we added profile functionality to user and attached the “users attend event” in event detail information page to show other users in the same event. We have delete the e-mail and other detailed information attributes that had been built default inside the user table when initializing the database, so users in this application will be totally anonymous which will make them braver to get in touch with others. And by adding the user sculpture, users will be more identical.

**13. Testing**

**13.1 Testing principle**

According to Quadri and Sheikh[[3]](#footnote-3), software testing should follow several principles. Some of them are:

1. **Design Effective Test cases**. It means making good choices. Designing too many similar ranges and similar types will only increase the consumption of testing but will not bring too much benefit.
2. **Test for valid as well as invalid Conditions**. This means that testing invalid values is as important as valid values. We need to make sure that we try to cover all situations that software may encounter, no matter good or bad.
3. **Testing must be done by different persons at different levels**. Testing by multiple people can increase the diversity of test designs and cover a wider range of cases (which others have not considered). Adding different levels of testing is to ensure the correctness of individual parts of the software and when combined.

**13.2 Testing Strategies**

A series of good strategies will contribute a lot to the success of software testing. In our testing, we adopt several testing strategies which are commonly used in software engineering:

1. Unit testing
2. Integration testing
3. System testing
4. Security testing

**13.2.1 Unit testing**

Unit testing is a level of software testing where individual units/ components of the software are tested[[4]](#footnote-4). Through unit testing, we can confirm whether each part (such as functions, methods) runs as they designed.

In our project, we tested several important forms of different pages. They include:

**Sign Up** Registration Form, **Sign In** Form. Specific test design and results are attached to the **Appendix**.

**13.2.2 Integration testing**

Integration testing is a level of software testing where individual units are combined and tested as a group[[5]](#footnote-5). Its purpose is to ensure that different parts of the system interact well with each other. Our project involves the interaction of different components. For example, the page skipping, or submits the form and redirected to other pages. Because that process involves the transfer of variables and the acquisition of the database. It's a particularly important part of testing. The results are attached to the **Appendix**.

**13.2.3 System testing**

System testing is a level of software testing where a complete and integrated software is tested. The purpose of this test is to evaluate the system’s compliance with the specified requirements[[6]](#footnote-6).

We adopted the black-box test method[[7]](#footnote-7), That is, without knowing the code and system composition structure, perform as many possible operations as possible on the system. The results are attached in the **Appendix**.

**13.2.4 Security testing**

Security testing attempts to verify that protection mechanisms built into a system will protect it from improper penetration[[8]](#footnote-8).Based on our project, we have done some simple security operations:

When the user not logged in:

* Operations other than login, registration and browsing events list are prohibited

When Login:

* Disallow access to non-existent events
* Prohibit join non-existent events
* **Hide** the common **http://url/admin** address using High Strength Password Entry, prevent the admin system being cracked.

Our security testing mainly tests some abnormal access based on those ranges. The test results are attached to the Appendix.

# Reflection

After two semesters work, we encountered many difficulties, including problems in the development process, problems between team members. Some problems have been solved, others have not. Although these problems are troubling us, we have also gained a lot of valuable experience.

**Problem in implement**

First problem of implementation is the **database migration**. Considered the target group of our application will not conclude lots of people, we used the **sqlite3** as the implement tool of our database. However, since we have to change our database very frequently especially in building the model of the whole program, there exist one error when we want to migrate the new version of models to our database. It occurs “**You are trying to add a non-nullable field 'name' to contact without a default**” error. Based on research of this problem on internet, we have found that the problem actually derived from two problems, the first is that some attributes of our database model **doesn’t exist the default value**, the second is that the migration files occurs **conflict with other migration files**. For the attributes that are able to define a default value, we set the default value in building the model. For the attributes that cannot define a default value and the situation that conflicts occurs between different migration files, we have checked that this problem will not occur usually if the model is valid. And we have checked that delete the database will not affect the following operations and developments. Based on these two reason, we use the Ostrich algorithm thoughts, which means we delete the whole database when error occurs in migration.

Second problem of implementation is the **view part**. When we change the Django setting attribute “DEBUG = True” into false, running “python manage.py runserver”, there exist one error “CommandError: You must set settings. ALLOWED\_HOSTS if DEBUG is False.”. The reason of this problem is that we haven’t set the ALLOWED\_HOSTS attribute correctly. The solution of this problem is to set the ALLOWED\_HOSTS attribute in settings into “ALLOWED\_HOSTS = [‘127.0.0.1’, ‘localhost’]”.

The third problem is that during the period of studying Django, the material of Django 3.6.1 is mixed with Django 2 or lower, thus **some of the code and function might from Python 2 and cannot been used in Python 3 and Django 3**. One problem based on this is that in log in part, when we are trying to import login library from django.contrib.auth.views, when we run Django service, it occurs “ImportError: cannot import name 'login' from 'django.contrib.auth.views'”. After changing the import part into “from django.contrib.auth import login”, Django service runs normally, but another error occurs when clicking login link. It occurs “TypeError at /users/login/

login() got an unexpected keyword argument 'template\_name'

Request Method: GET

Request URL: http://127.0.0.1:8000/users/login/

Django Version: 2.1.4

Exception Type: TypeError

Exception Value:

login() got an unexpected keyword argument 'template\_name'”

After searching lots of sources, the problem is that in Django 3 we have to **denote the app name** inside views.py. And the **pattern of url function has changed**. So, the problem have been solved after we added one line to denote the app name and change the url function “url(r'^login/$', login, {'template\_name': 'users/login.html'}, name = 'login'),” into “url(r'^login/$', LoginView.as\_view(template\_name='users/login.html'), name="login"),”.

Fourth problem is that **the logic of event might conflict with users**. In designing the application, we have divided the whole application into two role, the event and the user. However, some functionality of them might have overlap or even conflict with each other. For example, the **edit event** function inside the event part will conflict with the **authentication** function inside the user part. The reason of this problem is that the distribution of actual functionalities has the logic confliction. So, the solution of this problem is to change the delete the authentication functionality, and by adding the decoration @LoginRequired to achieve the authentication.

# Problem in Team

After working together in a whole semester, we have a deep understanding of each of our team members. Everyone is responsible for their work, and everyone has a great contribution for the project. This is a very valuable experience that very helpful for our future. But because this is the first time that we do a team cooperation, lack of work experience, so we found many problems in team.

Preparation and Design

In the Software Engineering module, at the beginning of project, it is necessary to confirm all the requirement of the project, so our team must meet the supervisor frequently. But we do not have a methodical plan and great preparation before we have a meeting. We just decide the meeting time and attend meeting on time. Our team do not have ideas about the project. For example, what do we ask about the requirement, what do we want to know about the project. And this led to most of our meetings being about requirements and wasted lots of time. The reason why our team lack of preparation is that we do not have a correct understanding of the meaning of 'team project', we treat it as a coursework, ask supervisor for advice and get feedback, then do it repeatedly. After the bad beginning, our team realized that we need have our own thinking of the project, requirement is not the most important, the essential of project is that we must design by ourselves, form a prototype in the mind then we can achieve it purposefully and clearly.

Communication in Team Work

In the first semester, our team don't have a unified time and place to work together. Everyone has his own working hours and place. But there is a drawback that we cannot exchange work achievement and information between each other, and this leads to a lot of complicated and redundant work. The solution is that we decide to stipulate a time to work together and that help us have a better communication and work effectively.

Self-Management

No one in the team has a good self-management ability. Except GRP, we also have other courses and many other important coursework. In addition, being late for class and absent from meetings are frequent occurrences. We often delayed GRP team work under the pretext of too much coursework, this will greatly affect other team members and reduce the efficiency of team work. The reason for all this is that each of us has no self-management ability, which is called procrastination, and it is a very bad habit. It influences not only team work, but also our living habits. Leading to life without planning and organization, seriously, it may even affect our health. So self-management is very crucial for our future work experience.

Time-Management

Last semester, our team worked out a timetable for tasks to be accomplished over a period of time. But the actual progress of our team is always behind the schedule. The reason why our progress always behind is that we do not have the ability about time management, not enough time was allotted to project work. Start work before deadline, lead the quality of the work very poor, and spend lots of time to modify. Such a vicious circle result in slower progress and worse quality. Therefore, it is very important to work with sufficient time, which can not only ensure the efficiency and quality of work, but also have enough time to check and optimize project.

Be Punctual

The concept of time is very important both in school and in future work. But that's what our team did poorly. For example, the meeting begins at one o'clock, but some team members are always a few minutes late. Maybe a few minutes are short, but it reflects a person's attitude towards work. Additionally, attend meeting on time is a respect to others. Since our team members are all friends, we didn't care much about the time at first, until there was a formal meeting with our supervisor, our supervisor discussed the problem with us very seriously. We realize that punctuality is essential. Our solution is that the person which late arrivals buy drinks for others. Although this method can't solve the problem fundamentally, but it gives us a sense of time, and it must be helpful for our future experience.

**Summary & Conclusion**

In the whole project, although all the group members meet a great number of questions, we finally solve them. The progress of the project is totally in schedule in the end. In the research part, the research that we get helps us to develop the project in a proper way.

At the beginning of first semester, we mainly tried to do the research and found proper software to do the development. After about one month to do these works, we decided to begin our programming part in the next three months. on the winter holidays, we met some problems on the development part, so we decided to change some strategies in the next semester. At the beginning of spring semester, we spent about two months to finish the software and website part in first version. We get some ideas from web and supervisor so that we can do the development part rapidly.

In the programming part, the advice from supervisor and searching from internet help us to program in a better way. As for different parts in the project this year, we divided it into stages so that all the group members can learn and have chance to do specified work. On every meeting, we will discuss for the progress of the whole project and get to prepare work for every member. In this way, when we need to meet the supervisor, we will know what to discuss in the formal meeting. Also, when every group member does their works after the meeting, the schedule of the whole project can be controlled to be not to late. For the software part, the initial version for the project will be an easy controlled website which can make users use it in a simple way. For the extra functions what the users need in the future like a mini program on WeChat version, the update version of this project will add them. The new version of this program can be changed when users give advices to us.

Appendix

1. Testing
   1. Unit Testing

In this section, we main focus the form validation of each page of our system because incorrect validation can lead to database errors, which can lead to unpredictable crashes.

**Part 1: SignUp Form**

|  |  |
| --- | --- |
| Field | Requirement |
| Username | Letters, digits and @/./+/-/\_ only. |
| Password | 1. contain at least 8 characters. 2. can't be a commonly used password. 3. can't be entirely numeric. |
| Pass confirm | Need the same password as above |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case | Aim | Input | Expected Output | Actual Output | Pass / Not Pass |
| 1 | Correct input | Username:JASON  Password:12345678w  Pass confirm:12345678w | success | success | P |
| 2 | Wrong username | Username:JASON?  Password:12345678w  Pass confirm:12345678w | invalid username | invalid username | P |
| 3 | Short password | Username:JASON  Password:abcdef  Pass confirm:abcdef | Short password | Short password | P |
| 4 | Numeric password | Username:JASON  Password:12345678  Pass confirm:12345678 | entirely numeric | entirely numeric | P |
| 5 | Short and Numeric | Username:JASON  Password:1234567  Pass confirm:1234567 | entirely numeric.  too short | entirely numeric.  too short | p |
| 6 | Null username | Username:””  Password:12345678 w  Pass confirm:12345678w | Requests missing field | Requests missing field | P |
| 7 | Null password | Username:JASON  Password:””  Pass confirm:”” | Requests missing field | Requests missing field | P |

**Part 2:SignIn Form**

|  |  |
| --- | --- |
| Field | Requirement |
| Username | Valid username A existed in DB |
| Password | Correct password of A |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case | Aim | Input | Expected Output | Actual Output | Pass / Not Pass |
| 1 | Correct input | Username:JASON  Password:1234567w | Log in | Log in | P |
| 2 | Non-exist username | Username:JASONW  Password:1234567w | Error reminding | Error reminding | P |
| 3 | Lowercase  username | Username:Jason  Password:1234567w | Error reminding | Error reminding | P |
| 4 | Wrong password | Username:JASON  Password:1234567m | Error reminding | Error reminding | P |
| 5 | Null username | Username:  Password:1234567w | Requests missing field | Error reminding | P |
| 6 | Null password | Username:JASON  Password: | Requests missing field | Error reminding | P |

* 1. Integration Testing

Test page jumps and redirections between different components.

**Note**: Navigation Bar Page Jump Tests Only Once(because base and content divided)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case | Aim | Input | Expected Output | Actual Output | Pass / Not Pass |
| 1 | Page jump | Status: Not logged in  Index > sign up | success | success | P |
| 2 | Page jump | Status: Not logged in  Index > sign in | success | success | P |
| 3 | Page jump | Status: Not logged in  Index > events list | success | NoReverseMatch | NP |
| 4 | Redirection | Status: Not logged in  Sign up > (sign in)>index | success | success | P |
| 5 | Redirection | Status: Not logged in  Sign in > (sign in) >index | success | success | P |
| 6 | Page jump | Status: logged in  Index > profile | success | success | P |
| 7 | Page jump | Status: logged in  Index > sign out | success | success | P |
| 8 | Page jump | Status: logged in  Index > events list | success | success | P |
| 9 | Page jump | Status: logged in  Event list > event detail | success | success | P |
| 10 | Page jump | Status: logged in  Event list > new event | success | success | P |
| 11 | Redirection | Status: logged in  New event > (add event) > event list | success | success | P |

* 1. System Testing

This part we will perform as many possible operations as possible on the system.

**Note**: Some tests similar to the previous unit and integration testing have been omitted.

And this is a combination of multiple rounds of test, some **Not Pass** results have been repaired.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case | Aim | Input | Expected Output | Actual Output | Pass / Not Pass |
| 1 | Sign up | Valid form | success | success | P |
| 2 | Sign in | Valid form | success | success | P |
| 3 | View events list | Click button | success | success | P |
| 4 | View event detail | Click button | success | success | P |
| 5 | Return events list | Click button | success | success | P |
| 6 | Join Event | Click button | success | success | P |
| 7 | Join another event | Click button | success | success | P |
| 8 | Join same event again | Click button | Error inform | success | NP |
| 9 | Quit from other’s event | Click button | Success inform | Success inform | P |
| 10 | Quit from own’s event | Click button | Should not view this function | success | NP |
| 11 | Edit own event | Click button | success | success | P |
| 12 | Edit other’s event | Click button | Should not view this function | crash | NP |
| 13 | Delete own event | Click button and confirm | success | success | P |
| 14 | Quit from a deleted event | Click button and confirm | Error inform | crash | NP |

* 1. Security testing

Our security testing mainly tests some abnormal access. (For example, the wrong address entered by the user himself, some pages that are not allowed to be accessed without login, etc.).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case | Aim | Input | Expected Output | Actual Output | Pass / Not Pass |
| 1 | Exceed access authority | Status: Not logged in  events > event detail | Redirect to Sign in | Redirect to Sign in | P |
| 2 | Access deleted event | Status: Not logged in  events > event detail | Redirect to Sign in | Redirect to Sign in | P |
| 3 | Access deleted event | Status: logged in  events > event detail | Redirect to Sign in | Redirect to Sign in | P |
| 4 | Visit fabricated URL | Status: NOT logged in  url: https://grpurl/adadad | 404 | 404 | P |
| 5 | Visit fabricated URL | Status: logged in  url: https://grpurl/adadad | 404 | 404 | P |
| 6 | Attempt to crack Admin interface | Status: NOT logged in  url:’https://grpurl/admin’ | 404 | 404 | P |
| 7 | Attempt to crack Admin interface | Status: logged in  url:’https://grpurl/admin’ | 404 | 404 | P |
| 8 | Access Admin | Status: Log in as user who not admin  url:’https://grpurl/ ZvRoo4fuB5bbVsZb3r63’ | No authority | No authority | P |
| 9 | Access Admin | Status: Log in as user who not admin  url:’https://grpurl/ ZvRoo4fuB5bbVsZb3r63’ | success | success | P |

**Meeting records**

|  |  |  |
| --- | --- | --- |
| Meeting Record | | Main Points |
| NO.1 | 2018.10.18 | Talk about what the software should be, a we-chat Mini Program or an app, the preparation of project website, who will be the stakeholders, the way in which users announce that they want to join in an event, the meaning of 'sports events', how to divide players into teams and the previous ways Matthew and his friends organize team sports. |
| PMB-435 | 1h |

|  |  |  |
| --- | --- | --- |
| Meeting Record | | Main Points |
| NO.2 | 2018.11.2 | We finished personas and several user stories and asked if Matt is appropriate. Is the mini-program in WeChat meet clients' need? Can we use that type to design our application? Since it’s not convenient for user to input lots of personal information, how to fetch the sports level and preference of user in order to automatically divide groups for multiple users? |
| PMB-435 | 1h |

|  |  |  |
| --- | --- | --- |
| Meeting Record | | Main Points |
| NO.3 | 2018..11.9 | Talk about some of the requirements, useful resources of our Group Work, first choice of the form of our software and the software engineering approach to use. |
| PMB-435 | 1h |

|  |  |  |
| --- | --- | --- |
| Meeting Record | | Main Points |
| NO.4 | 2018.11.16 | Talk about if we should give advice on the requirements to improve the software quality and apologize for the slow progress, with the excuse of loads of courseworks. |
| PMB-435 | 1h |

|  |  |  |
| --- | --- | --- |
| Meeting Record | | Main Points |
| NO.5 | 2019.2.26 | Recognize that the project is behind the schedule, we should spend more time to correct it. Make the new meeting record and update rapidly in the future. Get the new version of application and show it next meeting. Begin to design and control the new rate of progress. |
| PMB-435 | 1h |

|  |  |  |
| --- | --- | --- |
| Meeting Record | | Main Points |
| NO.6 | 2019.3.5 | Need to provide Meeting Record and great report to pass the module. To do the following tasks in application: 1. profile 2. log in 3. Inform other event. Focus more on the report, justify the decision in it. Need to diliverable thing in future. Correctly mix the report(Costumer, Supervisor) |
| PMB-435 | 1h |

|  |  |  |
| --- | --- | --- |
| Meeting Record | | Main Points |
| NO.7 | 2019.3.12 | Progress against the plan: 2rd version web view. However, need more usable functions. In meeting: 0. Find way to migrate current app to WeChat. 1. Add more usable functions 2. Supervisor recommend us some good items to deal with Final Report: ⚫ Latex Overleaf ⚫ 1 column Writing ⚫ ACM siggraph(Not sure spell) Format. ⚫ Focus on Final Report. 3. Very Focus on Time. 4. Try to figure out and use right tech methodology |
| PMB-435 | 1h |

|  |  |  |
| --- | --- | --- |
| Meeting Record | | Main Points |
| NO.8 | 2019.3.26 | Discussing about Final Report Draft. |
| PMB-435 | 1h |

1. Nick Babich (2017) The Most Important Color in UI Design. Available at: https://uxplanet.org/the-most-important-color-in-ui-design-d4f23aefffdf (Accessed: 28 Match 2019). [↑](#footnote-ref-1)
2. Carrie Cousins (2018) Serif vs. Sans Serif Fonts: Is One Really Better Than the Other? Available at: https://designshack.net/articles/typography/serif-vs-sans-serif-fonts-is-one-really-better-than-the-other (Accessed: 28 Match 2019). [↑](#footnote-ref-2)
3. S.M.K Quadri and Sheikh Umar Farooq,”Software Testing-Goals,Principles and Limitations,” International Journal of Computer Applications, Volume 6-No.9,September 2010. [↑](#footnote-ref-3)
4. Unit Testing. http://softwaretestingfundamentals.com/unit-testing (no date) (Accessed: 1 April 2019) [↑](#footnote-ref-4)
5. Integration Testing http://softwaretestingfundamentals.com/integration-testing (no date) (Accessed: 1 April 2019) [↑](#footnote-ref-5)
6. System Testing http://softwaretestingfundamentals.com/system-testing(No date) (Accessed: 1 April 2019) [↑](#footnote-ref-6)
7. Jovanovic and Irena,”Software Testing Methods and Techniques,” May 26,2008. [↑](#footnote-ref-7)
8. Sawant A A, Bari P H, Chawan P M. Software testing techniques and strategies[J]. International Journal of Engineering Research and Applications (IJERA), 2012, 2(3): 980-986. [↑](#footnote-ref-8)