



BEIJING-DUBLIN INTERNATIONAL COLLEGE

---

# COMP3017J SOFTWARE METHODOLOGY PROJECT REPORT

---

## OuterView: An Online Technical Interview Platform

### Authors

Jiehongxu Wu (21372323)

Sichen Li (21372309)

Te Qi (21372325)

Tongyu Wu (21372320)

Ziqin Ma (21372329)

December 16, 2023

# Contents

<b>1</b>	<b>Preface</b>	<b>3</b>
<b>2</b>	<b>System Specification</b>	<b>4</b>
2.1	Customer Problem Statement . . . . .	4
2.2	Glossary of Terms . . . . .	5
2.3	User Requirements Definition . . . . .	5
2.4	System Requirements Specification . . . . .	7
<b>3</b>	<b>System Design</b>	<b>14</b>
3.1	Sequence Diagrams . . . . .	14
3.2	Class Diagram . . . . .	15
3.3	System Architecture and System Design . . . . .	16
3.4	Design of Tests . . . . .	19
3.5	User Interface Design . . . . .	20
<b>4</b>	<b>Project Management</b>	<b>23</b>
4.1	Team Agreement . . . . .	23
4.2	Division of Work . . . . .	24

# 1 Preface

The intended readers of this document are primarily technical interviewing platform developers, product managers, and other stakeholders involved in the design, development, and implementation of collaborative technical interviewing software systems. The purpose of this document is to provide a comprehensive overview of the system requirements and customer problem statement, clearly defining the requirements and capabilities of the platform to guide the development team in creating solutions that meet the needs of interviewers and candidates.

This document has gone through several versions, each improving and expanding on the previous version. The rationale for creating a new version is to incorporate stakeholder feedback, update the document with the latest requirements, and improve overall clarity and organization.

A summary of the changes made in each release is as follows:

- Version 1.0
  1. Initial version of the document. Provided an overview of the system requirements of the technical interview platform.
  2. Identified customer problem statements and software system recommendations to solve problems faced by customers.
- Version 1.1
  1. The description of the problem statement was enhanced to include more examples and details, and stakeholder feedback was incorporated to further refine the content and requirements to illustrate the customer challenges.
  2. Drew use case diagrams and completed use case descriptions to improve the accuracy of project understanding.
  3. Completed the user requirements definition part to make the project requirements clearer.
- Version 1.2
  1. A preface has been added to identify the intended readership and provide context for the document.
  2. A glossary of terms was completed to clarify concepts related to professional terms used in the project.
  3. Completed the UI design and production of the front-end main page, reservation meeting page, and meeting information page.
  4. Implemented the backend logic related to sign up, login and reservation meeting and completed the corresponding database design.
  5. Build the project foundation.
- Version 1.3
  1. Completed system requirements definition.
  2. Updated document structure to improve organization and readability.
  3. The front-end and back-end implementation of login registration function.
  4. Further improvements in the backend to meet the need of frontend.

5. Completed the rendering and basic logic of the frontend main page, reservation meeting page, and meeting information page.
- Version 1.4
    1. Completed an interaction diagram primarily based on sequence diagrams and a class diagram.
    2. Completed interface specification related content.
    3. Created partial content on system architecture and system design.
    4. Completed the layout and style of the meeting room page, and integrated IDE and RTC.
    5. Completed the front-end and partial back-end implementation of the IDE page.
    6. Completed partial implementation of video calls.
  - Version 1.5
    1. Completed all aspects of system architecture and design.
    2. Understood and completed the content of algorithms and data structures.
    3. Completed the code implementation for video calls.
    4. Completed all code implementations for the IDE page.
  - Version 1.6
    1. Completed the overall requirements of the project.
    2. Completed all parts of the test cases and synchronized the refinement of the terminology to avoid potential ambiguity issues.
    3. A comprehensive review and revision of the project report was conducted.
    4. Each version of this document has been carefully prepared to provide a comprehensive and up-to-date overview of the technical interview platform requirements and customer question statements. The current version 1.6 is the most detailed and accurate representation of system requirements and objectives.

## 2 System Specification

### 2.1 Customer Problem Statement

As a customer, I am facing the problem of conducting technical interviews in an inefficient and time-consuming manner. Traditional interview methods often involve face-to-face meetings, which can be challenging due to time constraints, geographical limitations, and the need for specialized equipment. For instance, scheduling interviews with candidates from different time zones or remote locations can be difficult, and setting up a suitable environment for coding tasks can be expensive and time-consuming. Additionally, it can be difficult to accurately evaluate a candidate's skills and knowledge without a proper platform for live collaboration and code execution.

To address these challenges, I suggest a software system that provides a live, collaborative environment for both interviewers and candidates to write, execute, and debug code together. This system should help candidates easily share their skills and support multiple programming languages,

such as Python, JavaScript, and Java. It should also have video and audio interviewing functions, scheduling and management features for interviewers, a virtual whiteboard, and the ability to evaluate a candidate's code with automated test cases. Additionally, the system should allow me to replay interviews for future reference, helping me to better assess candidate performance and make more informed decisions.

By implementing this software system, I believe it will significantly improve the efficiency and effectiveness of technical interviews, enabling me to identify the best candidates more quickly and accurately. This will not only save time and resources but also enhance the overall quality of the interview process, leading to better matches between candidates and employers.

## 2.2 Glossary of Terms

- **Technical Interview Platform:** A website designed to facilitate live, collaborative technical interviews between interviewers and candidates.
- **Browser-based IDE:** A web-based integrated development environment that allows candidates to share their coding skills during interviews.
- **Programming Language:** A set of rules and conventions used to create computer programs.
- **Video and Audio Interviewing:** Functionality that allows interviewers and candidates to communicate via video and audio during the interview process.
- **Scheduling and Management:** Features that enable interviewers to schedule and manage interviews efficiently.
- **Virtual Whiteboard:** A digital whiteboard that allows interviewers and candidates to visualize their technical conversations during interviews.
- **Automated Test Cases:** Pre-written tests that evaluate a candidate's code for correctness and efficiency.
- **Playback Function:** A feature that allows interviewers to replay the entire interview after it has ended.
- **Not Start:** The meeting has not yet started, and the interviewee is waiting for the meeting.
- **In progress:** The meeting is currently in progress, and the interviewees are in the meeting status.
- **Completed:** The meeting has ended, and the interview is now in its current state.

## 2.3 User Requirements Definition

Requirements List		
Priority	Type	Description
10	Functional Requirements	Both interviewers and interviewees can see the meeting arrangement in chronological order downwards at the bottom of the homepage.

10	Functional Requirements	Interviewers can initiate audio and video calls or pure audio calls with other designated users.
10	Functional Requirements	Interviewers can cancel a meeting
5	Functional Requirements	Both interviewers and interviewees can set a reminder for their meetings.
10	Functional Requirements	Both interviewers and interviewees can end the meeting session.
8	Functional Requirements	Both interviewers and interviewees can re-enter the meeting session they leave as long as the other users in the meeting remain in the session.
5	Functional Requirements	Both interviewers and interviewees can choose which microphone and camera they use for meetings.
5	Functional Requirements	Both interviewers and interviewees can choose different programming languages for coding in the meeting.
8	Functional Requirements	Both interviewers and interviewees can run the code in the editor and debug them..
5	Functional Requirements	Both interviewers and interviewees can comment and change the code in the editor.
8	Functional Requirements	Interviewers can set the questions and send them to the interviewees..
5	Functional Requirements	Interviewers can send answers to the interviewees.
3	Non-Functional Requirements	Interviewers can set a timer for the session and each question.
10	User Interface Requirements	Both interviewers and interviewees have buttons for booking meetings and joining meetings on the homepage.
8	User Interface Requirements	Interviewers have a button for canceling or changing a meeting in the meeting list on the homepage.
8	User Interface Requirements	Both interviewers and interviewees can see tags containing the name, start time, organizer for each booked meeting in the list on the homepage.
3	User Interface Requirements	Both interviewers and interviewees can see the reminder they set for the meeting in the list on the homepage.
3	User Interface Requirements	Interviewers have buttons for canceling and changing the meeting in the meeting session.
10	User Interface Requirements	Both interviewers and interviewees have a button for exiting from the current page in the meeting session.
10	User Interface Requirements	Both interviewers and interviewees can see the ID and title of the meeting and the username and position of all the users in the meeting session.

8	User Interface Requirements	Both interviewers and interviewees can see the viewing frame of their camera and the other sides' and change the size of all these frames.
---	-----------------------------	--

## 2.4 System Requirements Specification

### Stakeholders

The stakeholders of the system are the interviewers and interviewees who need to conduct the programming interview. In our case, the interviewer and the interviewee can set up a meeting, so that the interviewer can see the interviewee's programming situation in real time, and make comments and code compilation in real time. The interviewee will also be able to see the interviewer's comments in real time and will be able to communicate via video.

### Actors and Goals

Actors and Goals List	
Actors	Goals
Interviewers	Registration and login
Interviewers	Schedule or cancel a meeting
Interviewers	Join or exit a meetingn
Interviewers	Ability to compile and run code
Interviewers	Ability to write comments
Interviewers	Turn on/off microphone and camera
Interviewees	Registration and login
Interviewees	Join or exit a meeting
Interviewees	Ability to compile and run code
Interviewees	Ability to write comments
Interviewees	Turn on/off microphone and camera
Interviewees	Ability to choose programming language

### User Cases

### User Case Diagram:

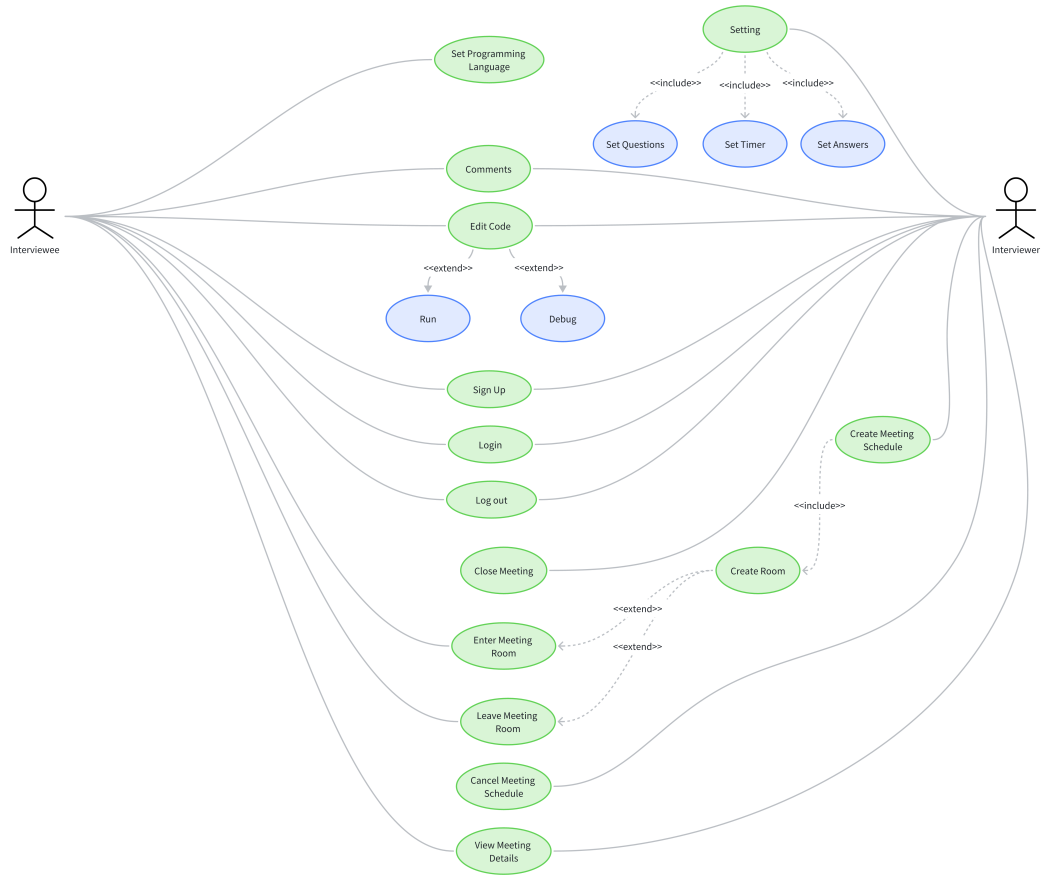


Figure 1: Use Case Diagram



**Casual Description:**

- **UC1:**  
**Sign Up:** Users can create new accounts.
- **UC2:**  
**Login:** Users can login using the registered account.
- **UC3:**  
**Log out:** Users can exit the current account.
- **UC4:**  
**Enter Meeting Room:** Users can enter the meeting.
- **UC5:**  
**Leave Meeting Room:** Users can leave the meeting.
- **UC6:**  
**Create Meeting Schedule:** Interviewers can create new meeting schedule.
- **UC7:**  
**Cancel Meeting Schedule:** Interviewers can cancel a meeting schedule.
- **UC8:**  
**View Meeting Details:** Users can see the details of a scheduled meeting.
- **UC9:**  
**Create Room:** Interviewers can create new meeting rooms.
- **UC10:**  
**Close Meeting:** Interviewers can end the meeting rooms.
- **UC11:**  
**Set Programming Language:** Interviewees can choose a programming language.
- **UC12:**  
**Setting:** Interviewers are able to set interview questions, answers, answer times, etc.
- **UC13:**  
**Edit Code:** Interviewees can edit code, run and debug code.
- **UC14:**  
**Comments:** Users can write comments.
- **UC15:**  
**Set Questions:** Interviewers can set questions.
- **UC16:**  
**Set Timer:** Interviewers can set a timer for the interviewees in a timed question.

- **UC17:**

**Set Answers:** Interviewers can set answers for the questions they set.

**Detailed User Case Descriptions:**

<b>Use Case UC1: Sign Up</b>
<b>Summary:</b> Describes how the user successfully goes to the home page.
<b>Precondition:</b> The Sign Up page must be open in a browser meeting the system requirements.
<b>Trigger:</b> mouse click
<b>Event Process:</b> <ol style="list-style-type: none"> <li>1. The user will be prompted to enter an user name, user id, email and password.</li> <li>2. The user inputs the information.</li> <li>3. The user clicks "Sign Up Now" button to register an account.</li> </ol>
<b>Use Case UC4: Enter Meeting Room</b>
<b>Summary:</b> The user (interviewer and/or interviewee) can join the meeting room, start the interview, choose to turn on/off the camera and microphone, open the shared IDE, and exit the meeting room
<b>Precondition:</b> The user is added to the meeting room by the interviewer. The meeting room is available on the user's home page.
<b>Trigger:</b> mouse click
<b>Event Process:</b> <ol style="list-style-type: none"> <li>1. The user clicks the "Enter Meeting Room" button next to the meeting room to connect to the meeting room.</li> <li>2. Users enable the camera and microphone (off by default).</li> <li>3. The screen displays the screen status, microphone status, name and other information of other users in the meeting room.</li> <li>4. Start the interview.</li> <li>5. Click Share IDE to share the IDE with others in the meeting room.</li> </ol>

<b>User Case UC6: Create Meeting Schedule:</b>
<b>Summary:</b> Users can create meetings.
<b>Precondition:</b> The main window must be open in a browser meeting the system requirements.
<b>Trigger:</b> mouse click
<b>Event Process:</b> <ol style="list-style-type: none"><li>1. If the user wants to create a meeting, click "Create Meeting Schedule".</li><li>2. Users need to fill in more information for this meeting in the display section. Information to fill in may includes:<ul style="list-style-type: none"><li>• Meeting name (default name is username)</li><li>• Start time and end time</li><li>• Duration of the meeting (30min, 45min and 60min for option)</li><li>• Participants of the meeting</li></ul></li><li>3. Users invited to the meeting. This field requires an ID.</li><li>4. The user clicks "Complete appointment" below the display section to finish filling the information.</li><li>5. After completion, the user will return to the main page and see the newly scheduled meeting in the right scheduling table.</li></ol>

<b>User Case UC8: View Meeting Details:</b>
<b>Summary:</b> The user can see the details about a meeting.
<b>Precondition:</b> The user has the meeting he or she wants to attend scheduled.
<b>Trigger:</b> mouse click
<b>Event Process:</b> <ol style="list-style-type: none"> <li>1. In the configuration table on the right, click the meeting for which the user wants to view specific information.</li> <li>2. The page will provide a pop-up window to show the details of the meeting. The information displayed may include: <ul style="list-style-type: none"> <li>• Meeting name (default name is username)</li> <li>• Start time and end time</li> <li>• Duration of the meeting (30min, 45min and 60min for option)</li> <li>• Participants of the meeting</li> </ul> </li> <li>3. The user clicks "Enter Meeting" in the pop-up window to complete the request; or click the "Back" button in the upper left corner to close the popup.</li> <li>4. The "Cancel Meeting" and "Edit Meeting" buttons at the bottom of the pop-up window are only eligible for the people who create the meeting schedule to click; the invitees cannot click either button. After clicking "Cancel Meeting", the meeting will be deleted; after clicking "Edit Meeting", the page will jump to the information filling page in the scheduled meeting where the user can modify it.</li> </ol>
<b>User Case UC13: Edit Code</b>
<b>Summary:</b> Participants can edit code, run and debug code.
<b>Precondition:</b> The participant enters the meeting and the interviewee selects the language to enter the corresponding IDE
<b>Trigger:</b> mouse click
<b>Event Process:</b> <ol style="list-style-type: none"> <li>1. The interviewee chooses the programming language.</li> <li>2. The interviewee enters the code</li> <li>3. The interviewer comments in real time</li> <li>4. The interviewee runs or debugs the code</li> <li>5. The interviewer see the results and communicate by camera or comments.</li> </ol>

**Acceptance Test Cases:**

<b>Use Case: Edit Code</b>
<b>Precondition:</b> The participant enters the meeting and the interviewee selects the language to enter the corresponding IDE
<b>Expected Result:</b> Attendees can edit code, run and debug code in real time.
<b>Steps:</b> <ol style="list-style-type: none"> <li>1. The participant enters the meeting and the interviewee selects the language to enter the corresponding IDE</li> <li>2. Participants enter the code</li> <li>3. Participants click Run or Debug</li> </ol>
<b>Actual Result:</b> Attendees can edit code, run and debug code in real time.
<b>Use Case: Create Room</b>
<b>Precondition:</b> The meeting window must be open in a browser meeting the system requirements.
<b>Expected Result:</b> The interviewer can invite the interviewee and create the meeting, and the interviewee can see the invited meeting and accept or decline the invitation.
<b>Steps:</b> <ol style="list-style-type: none"> <li>1. The interviewer creates the meeting</li> <li>2. The interviewer chooses the invitee</li> <li>3. The interviewer sets the time and content of the meeting</li> <li>4. Interviewers release meetings</li> <li>5. Interviewers release meetings</li> </ol>
<b>Actual Result:</b> The interviewer can invite the interviewee and create the meeting, and the interviewee can see the invited meeting and accept or decline the invitation.
<b>Use Case: Login and Sign Up</b>
<b>Precondition:</b> The user enters the login and registration page
<b>Expected Result:</b> The user successfully registered, the information was saved in the database, and the user was able to log in
<b>Steps:</b> <ol style="list-style-type: none"> <li>1. Open the registration or login page</li> <li>2. Enter the username and password</li> <li>3. Click the Sign Up or sign in button</li> </ol>
<b>Actual Result:</b> The user successfully registered, the information was saved in the database, and the user was able to log in

### 3 System Design

#### 3.1 Sequence Diagrams

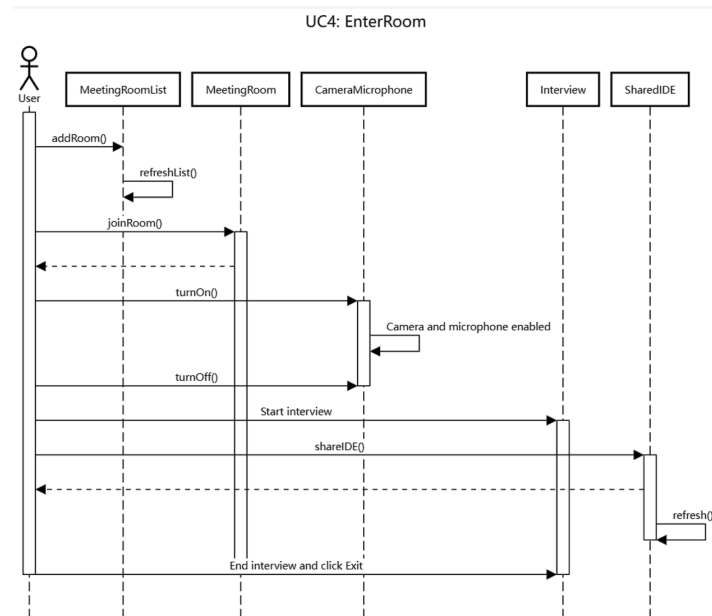


Figure 2: Use observer mode to check whether the user is in the meeting.

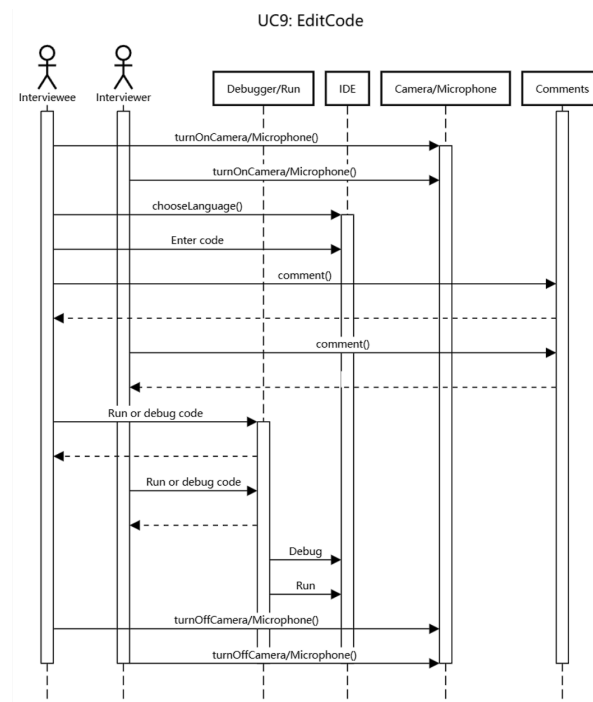


Figure 3: Use the chain of responsibility pattern to let the interviewer and the interviewee work with the same compiler and pass code.

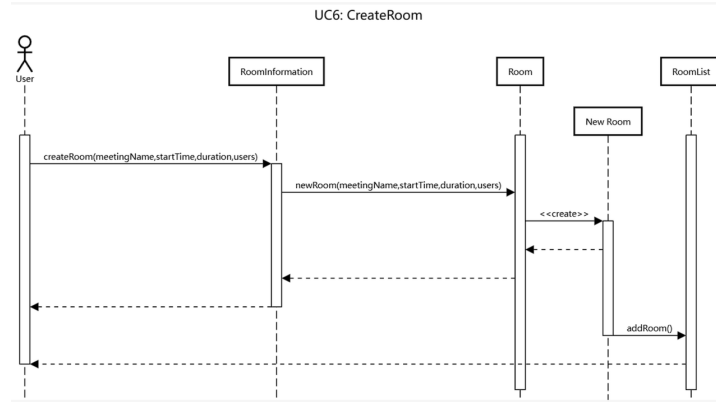


Figure 4: The meeting room is created using Builder mode.

### 3.2 Class Diagram

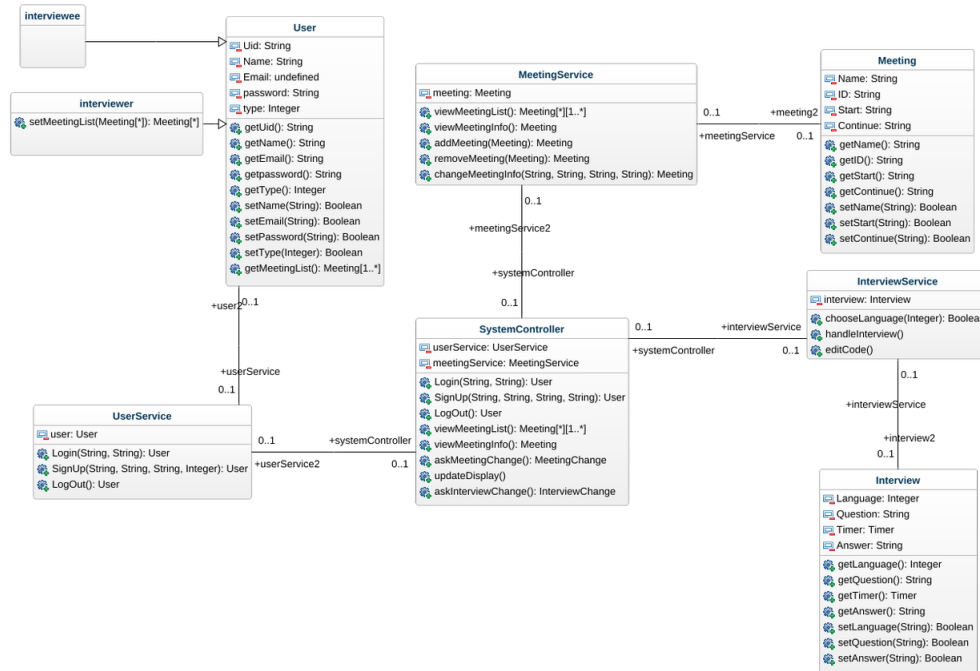


Figure 5: Class Diagram

#### Description of Responsibilities

- **Meeting:** The Meeting class is a basic entity class. Its attributes include name, id, start time, and duration continue. The Meeting class contains the following methods: geter() method for each attribute and setter() method for each attribute excluding ID.
- **MeetingService:** The MeetingService class is a basic entity class. Its properties are instance objects of the Meeting class. The MeetingService class includes the following methods:

methods for browsing meeting lists, methods for browsing meeting information, methods for modifying meeting information, methods for adding and removing meetings.

- **User:** The User class is a basic entity class, which is the parent class of the Interviewee class and the Interviewer class. Its properties include uid, name, email, password and type filled in during registration, where type is used to distinguish whether the user is an internee or an interviewee. The User class contains the following methods: `getter()` and `setter()` methods for each attribute, and `getMeetingList()` methods for obtaining the meeting list.
- **Interviewee:** The Interviewee class is a basic entity class that is a subclass of the User class.
- **Interviewer:** The Interviewer class is a basic entity class that is a subclass of the User class. In addition to reusing all methods of the parent class, the Interviewer class also has the `setMeetingList (Meeting [*])` method for creating a meeting list.
- **UserService:** The UserService class is a basic entity class. Its property is an instance object of the User class. The UserService class includes the following methods: registration and login for users when entering the application, as well as logout methods when exiting the application.
- **Interview:** The View class is a basic entity class. Its properties include programming language, question, answer duration timer, and answer. The Interview class contains the following methods: `getter()` methods for each property and `setter()` methods for each property that removes timers.
- **InterviewService:** The InterviewService class is a basic entity class. Its property is an instance object of the Interview class. The InterviewService class includes the following methods: the language selection method `chooseLanguage (Integer)`, and methods for managing interviews and editing code.
- **SystemController:** The SystemController class is a basic entity class. Its properties are instance objects of the UserService class and the MeetingService class. The SystemController class includes several methods: registration and login for users when entering the application, as well as logout methods when exiting the application. The browsing methods for meeting lists and meeting information. For the inquiry method modified during the meeting, for the inquiry method modified during the interview, and for updating the display method.

### 3.3 System Architecture and System Design

#### Architectural Styles

- **Frontend:** The front-end uses the React framework to generate SPA (Single Page Application), which requests the back-end to return JSON data and then renders it by the front-end framework. The front end uses `tsx` based on `typescript` to generate html templates, `typescript` is used to execute the script language and control the front-end logic, and `moduleCSS+tailwindCSS` is used to generate the style sheet. Use the `webRTC` interface that comes with the browser to read the audio and video streams. After entering the room, the front and back ends will use `socket.io` to establish a full-duplex P2P connection.
- **Backend:** The backend of the platform makes use of Flask as its framework to support all the functionalities, from creating an account to making a meeting appointment. It uses SQL Alchemy to make it easier and clearer to generate database models.



## UML Package Diagram of Subsystems

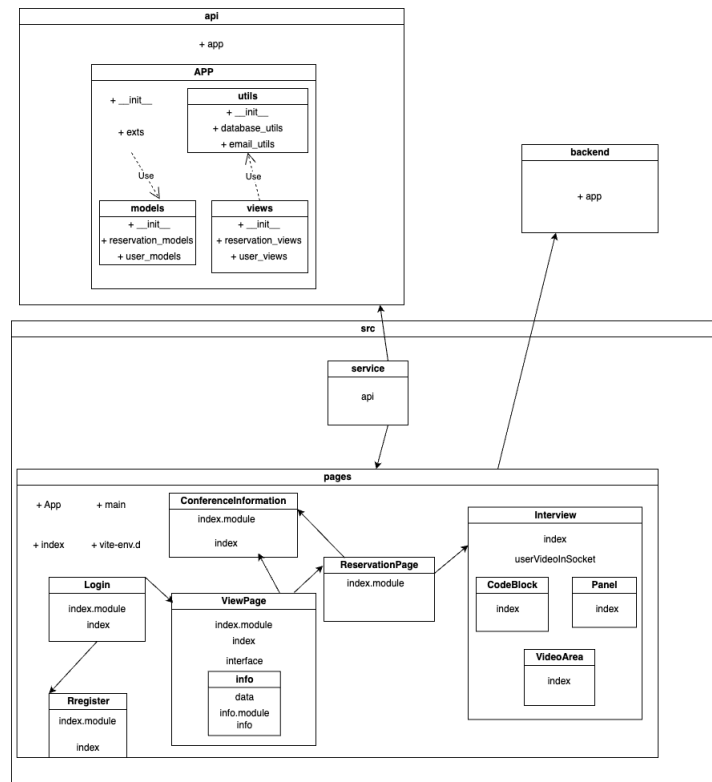


Figure 6: Package Diagram

## Database Schema of the System

There are three tables in the database: **user**, **reservation** and **reservation\\_participant**. The **user** table is for storing data related to the user account of this platform (user ID, password, name, type of user, etc.), with **user\\_id** as its primary key, identifying each user. The **reservation** table is for storing data related to the information of meetings. Its attributes involve the name, start time, end time, state, details and date of a meeting. The table uses an auto-increment ID serving as the primary key to identify each meeting. The **reservation\\_participant** table is a helper table for remembering the relationship of users and meetings, that is, the number of users involved in a meeting. It has an auto-increment ID as its primary key.

The design of the database meets the requirements of the Boyce-Codd normal form. Each table has a primary key, and all attributes in each table are functionally dependent on their respective primary keys, which are also superkeys. This eliminates redundancy and updates anomalies by ensuring every non-trivial functional dependency is on a superkey.

## Global Control Flow of the System

The user registers an account, and the system backend creates a new account and stores it in the database. When the user logs in, the backend verifies that the password is correct. If the password is incorrect, an error message is returned. If the password is correct, the user enters the main page.

On the main page, the system displays the user's nickname and identity, as well as a button to return to the login interface, displays user-related meeting information, and the interviewer additionally displays a button to schedule a meeting.

When the interviewer clicks the reservation button, the system will jump to the reservation page. The system will display the meeting-related information fill-in column. After the interviewer fills in the relevant information, click the reservation button to create the meeting. The system will navigate to the main meeting page and re-render the meeting information.

When the user clicks the meeting information icon, the system will jump to the meeting information page, where the system displays meeting-related information.

On the meeting information page, the interviewer clicks the Cancel Meeting button to cancel the meeting, and the system navigates to the main page and re-renders the meeting information.

On the meeting information page, the interviewer clicks the Edit meeting button to edit the meeting, and the system navigates to the reservation page. After the interviewer refills the information, the system navigates to the main page and re-renders the meeting information.

In the meeting information page, the user clicks the Enter meeting button, and the system navigates to the meeting page.

After clicking to join the interview, the backend will verify whether the user is in this interview. If not in the interview, the user is allowed to enter the interview room page.

After entering the interview room page, the interviewer/interviewee needs to click a button to turn on his or her camera and microphone.

If the network is normal, the connection will be established and the interviewer/interviewee can see the other party's ID, screen and share the IDE.

## Hardware Requirements

- **Minimal configuration:**

1. The device can run the chrome browser.
2. The device can access the Internet (for P2P connection establishment)

- **Optimal configuration:** Available camera/microphone to enable video and audio streaming, called via Chrome browser.

## Algorithms and Data Structures in the System: HashTable

```

1 // Save all connections
2 const clients = {};
3 setInterval(() => {
4   for (const client in clients) {
5     clients[client].ws.emit('updateAllClients', {
6       data: Object.keys(clients),
7     })
8   }
9 }, 2000)

```

User (client) related information, including socket instance object, room id and user id are stored in the clients object.

The key is the user's id, and the value is user-related information.

```
1 switch (message.type) {  
2   case "connect":  
3     // Save the connected clients  
4     clientID = message.data.myID;  
5     roomID = message.data.roomID  
6     console.log(`${clientID} has connected in ${roomID}`)  
7     clients[clientID] = {  
8       clientID,  
9       roomID,  
10      ws,  
11    };  
12    ...  
13 }
```

When a user initiates a video stream push to another user, the socket instance object of the other user is obtained through the `clientID`, and the `emit` method is called to trigger the event.

Using `hashTable` to store information makes the structure clear and improves access efficiency.

### 3.4 Design of Tests

#### 1. Unit Testing:

- Test Case 1: Browser-based IDE Functionality
  - Description: Verify that the browser-based IDE allows candidates to write, execute, and debug code.
  - Steps:
    - (a) Open the IDE
    - (b) Write code in different programming languages.
    - (c) Execute and debug code within the IDE.
    - (d) Ensure proper error handling.
- Test Case 2: Multi-language Support
  - Description: Confirm that the platform supports multiple programming languages.
  - Steps:
    - (a) Write and execute code in different programming languages.
    - (b) Verify language-specific features (e.g., syntax highlighting).
    - (c) Ensure smooth transitions between languages.
- Test Case 3: Video and Audio Interviewing
  - Description: Test the functionality of video and audio interviewing.
  - Steps:
    - (a) Initiate a video interview session.
    - (b) Confirm video and audio feeds are working.
    - (c) Test mute/unmute and video on/off functionalities.
    - (d) Verify the quality of audio and video.
- Test Case 4: Interview Scheduling and Management

- Description: Ensure interview scheduling and management features work correctly.
- Steps:
  - (a) Schedule an interview for a specific date and time.
  - (b) Confirm notifications are sent to interviewers and candidates.
  - (c) Check rescheduling and cancellation functionalities.

## 2. Test Coverage:

- Aim to cover all critical functions of the browser-based IDE, video/audio interviewing and scheduling.
- Ensure each supported programming language is tested.

## 3. Integration Testing Strategy:

- Integrate components to ensure they work together seamlessly.
- Verify communication between the browser-based IDE, video and audio functions and scheduling features.
- Test the system's ability to handle multiple users in a live interview scenario.

## 4. Integration Testing Plans:

- Steps:
  - (a) Conduct end-to-end testing of a complete interview session.
  - (b) Simulate concurrent users accessing the platform.
  - (c) Test interruptions and resume functionality during an ongoing interview.
  - (d) Check for compatibility with different browsers and devices.

## 3.5 User Interface Design

### User Interface Description

The user interface is divided into four main parts: *Sign Up/Login*, *Joining a Meeting*, *Booking a Meeting*, and a *Meeting Schedule*. The following descriptions provide a brief overview of each feature included and offer preliminary models of some key features.

- **Joining a Meeting:** After clicking to join the meeting, the meeting details will be displayed in a pop-up window.
- **Meeting Schedule:** The meeting schedule contains the meetings that users will participate in in order of meeting start time. After clicking on any of the meetings, the meeting details will appear and the meeting details will be displayed in a pop-up window.
- **Meeting Details:** The content in this pop-up window includes: *meeting name*, *meeting duration*, *start time and end time of the meeting*, *initiator of this meeting*, *canceling the meeting*, *editing the meeting* and *entering the meeting*. The user clicks *Enter Meeting* in the pop-up window to enter the meeting room; or clicks the *Return* button in the upper left corner to close the pop-up window. Only the meeting initiator is allowed to click the *Cancel Meeting* and *Edit Meeting* buttons below the pop-up window; invitees cannot click these two buttons. After clicking *Cancel Meeting*, the meeting will be deleted; after clicking *Edit Meeting*, the page will jump to the information filling page in the *appointed meeting* for modification.

- **Booking a Meeting:** After clicking to schedule a meeting, the user needs to fill in the following information for this meeting in a new page: *meeting name* (the default name is "username's scheduled meeting"), *start time*, *duration of the meeting* (options are: 30min, 45min, 60min) and *the invitees*. This column requires an ID. The user clicks *Complete Appointment* below the page to end information filling. After completed, the user will return to the main page and see the newly scheduled meeting in the *Meeting Schedule* on the right side of the page.

## User Interface Prototype

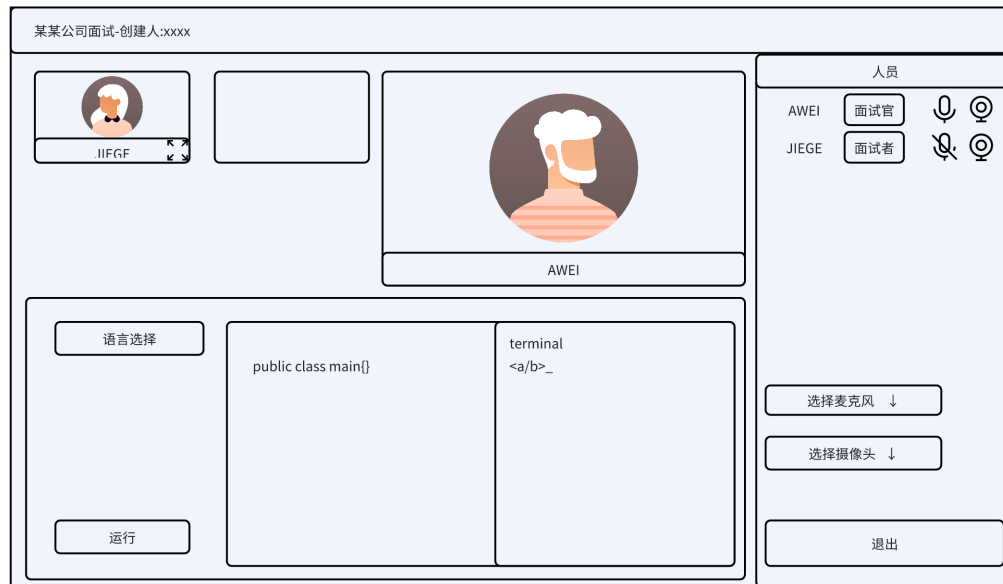


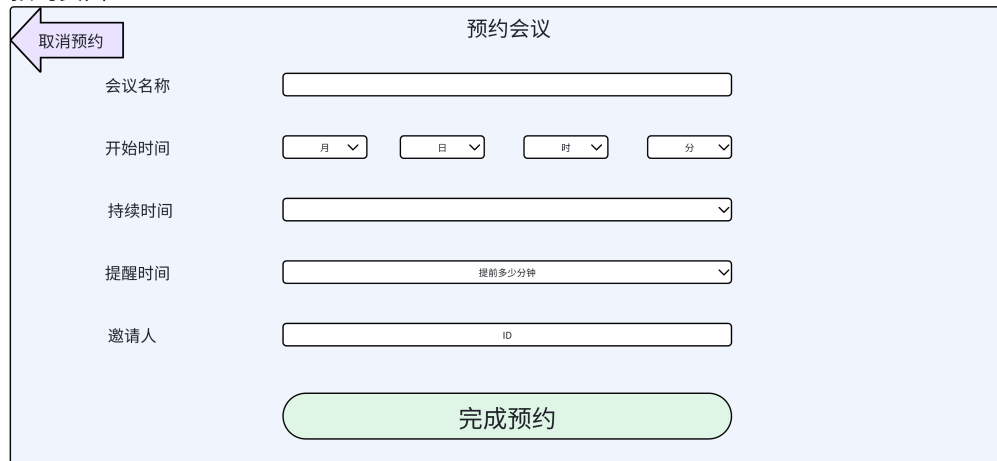
Figure 7: UI Prototype 1

## 主页面

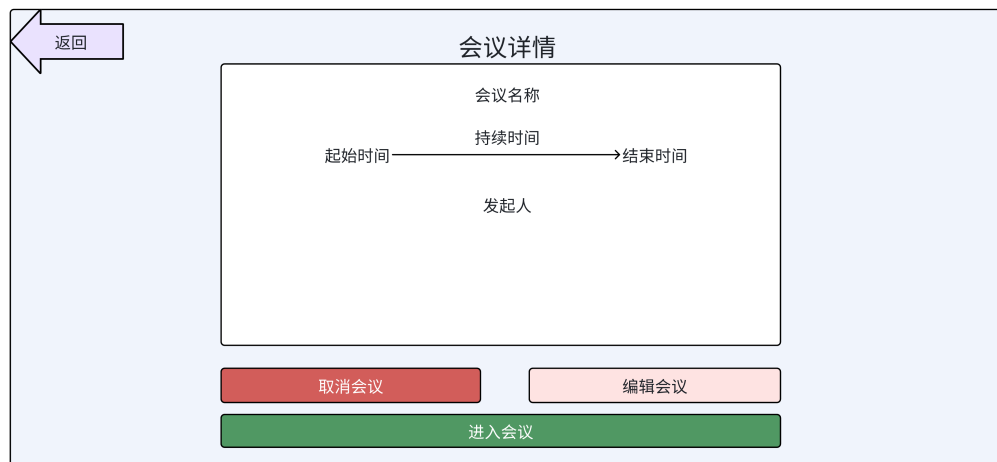


The main page UI features a light blue background. On the left side, there is a vertical navigation bar containing a profile icon with an arrow pointing to '账户设置' (Account Settings), an 'ID' label, a purple '加入会议' (Join Meeting) button, a purple '预约会议' (Book Meeting) button, and a purple arrow button labeled '返回登陆 (退出) 账号' (Return Login (Exit) Account). The main content area on the right is titled '时间' (Time) and contains two identical meeting reservation cards. Each card has a header with '时间' (Time), '会议号' (Meeting ID), and '状态' (Status). The first card shows '会议名称一' (Meeting Name 1) and '预约一' (Book 1), while the second shows '会议名称二' (Meeting Name 2) and '预约二' (Book 2). Both cards have a right-pointing arrow.

## 预约页面



The booking page UI has a light blue background. On the left, there is a purple arrow button labeled '取消预约' (Cancel Booking). The main content area is titled '预约会议' (Book Meeting) and contains a form with the following fields: '会议名称' (Meeting Name) with a text input; '开始时间' (Start Time) with four dropdown menus for '月' (Month), '日' (Day), '时' (Hour), and '分' (Minute); '持续时间' (Duration) with a dropdown menu; '提醒时间' (Reminder Time) with a dropdown menu labeled '提前多少分钟' (How many minutes in advance); and '邀请人' (Inviter) with a text input labeled 'ID'. At the bottom of the form is a green '完成预约' (Complete Booking) button.



The meeting details page UI has a light blue background. On the left, there is a purple arrow button labeled '返回' (Return). The main content area is titled '会议详情' (Meeting Details) and contains a white box with the following information: '会议名称' (Meeting Name), '起始时间' (Start Time) connected by a line to '持续时间' (Duration), which is then connected by an arrow to '结束时间' (End Time), and '发起人' (Initiator) below. At the bottom of the page are three buttons: a red '取消会议' (Cancel Meeting) button, a pink '编辑会议' (Edit Meeting) button, and a green '进入会议' (Enter Meeting) button.

Figure 8: UI Prototype 2

## 4 Project Management

### 4.1 Team Agreement

- **Meeting Time:** Every Monday at 9pm and every Friday at 5pm.
- **Code Management:** This project uses git to manage the source code and other related resources. A private repository on GitHub is used for hosting code and team collaboration. Commit message should be clear and follow the convention like  
`<type>[optional scope]: <description>`. Any changes that may be deemed destructive should be made on a separate branch and then merged to the main branch after stabilized and being reviewed by at least one other team member. All backend code should be tested using Apifox before submitted.

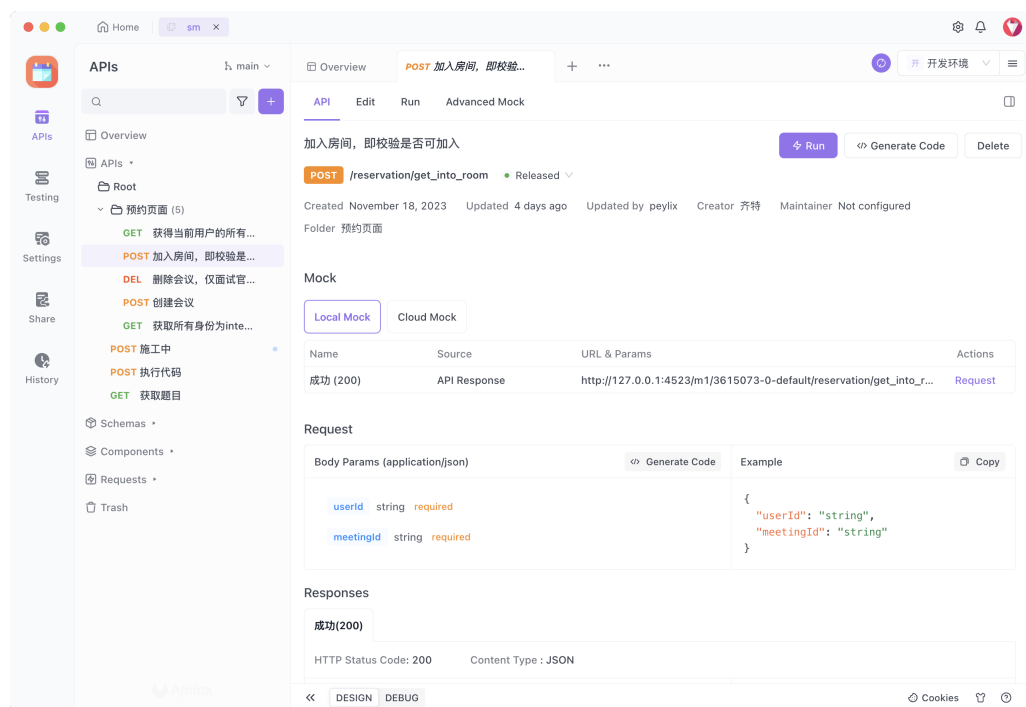


Figure 9: Using Apifox to test APIs

- **Details for Cooperation:** A clear API document with detailed comments should be provided by the team members who is responsible for the backend. The backend developers and frontend developers should work closely together.
- **Overall Management:** In order to better manage tasks and project progress, a Gantt chart was created so that the phased progress of the project and the task allocation of each member can be clearly seen. Not only does this help track overall progress, but it also provides clear direction for team members so they can better understand the overall framework of the project and their own roles.

The Gantt chart can be found in

<https://cs9lscied6.feishu.cn/share/base/view/shrcnWfsEBhYGWS01AVpVeuKQTd>

## 4.2 Division of Work

Frontend:

- Te Qi
- Tongyu Wu
- Jiehongxu Wu

Backend:

- Sichen Li
- Ziqin Ma

The detailed division and contribution of each member is as follows:

Task	Te	Jiehongxu	Ziqin	Tongyu	Sichen	
Preface		100%				0.5
Customer Problem Statement		80%	20%			0.5
Glossary of Terms		70%			30%	0.25
User Requirements Definition	10%	20%	40%	10%	20%	1
System Requirements Specification			90%		10%	2
Interaction Diagrams			100%			0.5
Class Diagram and Interface Specification		50%		50%		1
System Architecture and System Design	20%		30%	20%	30%	1.5
Algorithms and Data Structures (if applicable)	100%					0.5
User Interface Design		100%				0.5
Design of Tests		100%				0.5
Project Management	10%	70%		10%	10%	1
References	50%		50%			0.25
Code	30%	5%	5%	30%	30%	10