Interactive Language Exercises Software

Client: Cecilia Trevino, Modern Language Centre, King's College London.

Development Team - The Truth Seekers:

Chun (Leon) Chan

Chung (Wesley) Tsang

Denis Saidov

Kwo (Frankie) Lee

Marco Pacheco Costa

Martina Naydenova

Yin (Evans) Chan

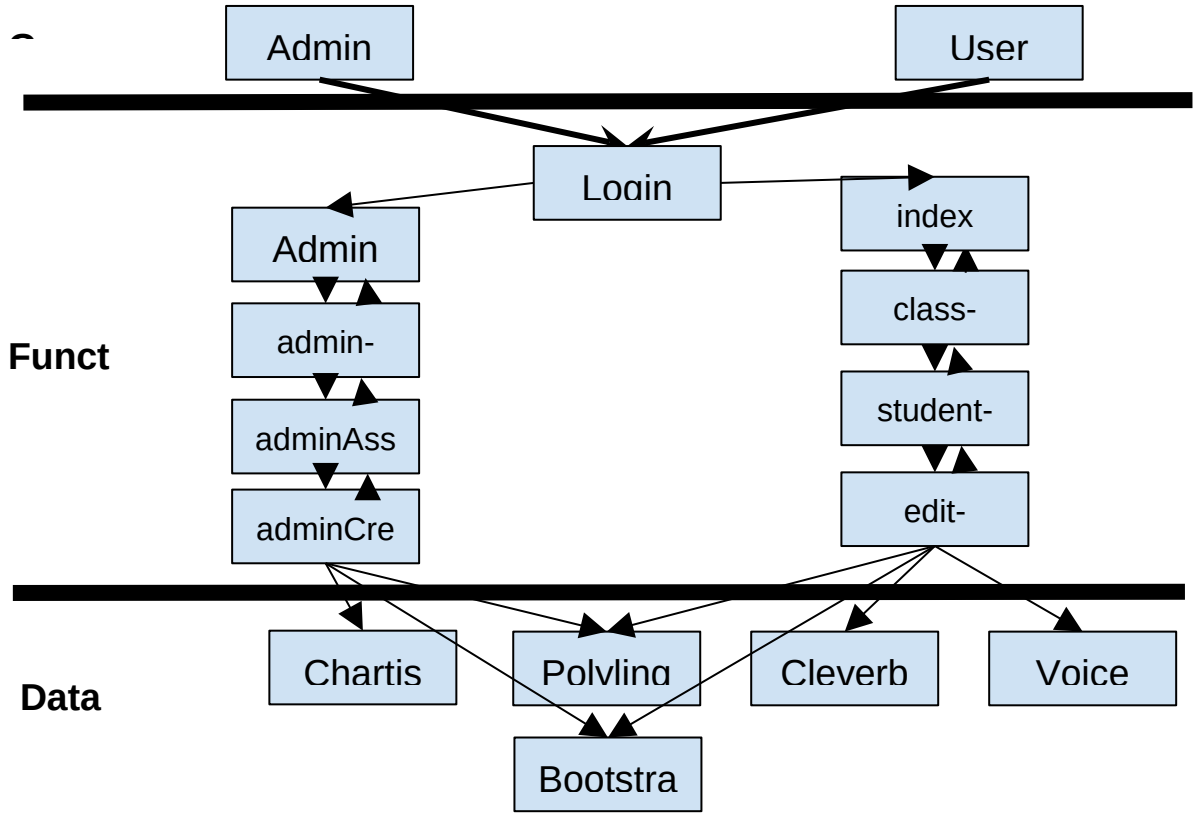
Introduction

This project was closely developed with our client Cecilia Trevino and we always took consideration the key factors of the location where it could be deployed (Modern Language Center).
The main objective of this project is to create a web application allowing language teacher(users) to create custom interactive exercises that the student(users) could then solve by themselves.
Before having our first meeting with our client we have created a Briefing with possible requirements for the project based on the current information we had.
During the meeting we showed this briefing to our client and based on what we discussed we managed to get a full list of the requirements.
The main requirements wanted by the client are presented in the following order:

- Allow the teacher(users) to create different types of interactive exercises (Example: Multiple Choice, Drag n Drop, etc.);
- Allow the student(users) to solve these exercises;
- Have a login system to distinguish both types of users;
- Have a simple and modern user interface;
- Some types of exercises based on the “Hot potatoes” application;
- Have some gamification included in the application.
- Allow it to run on a web environment;
- Have a stable and reliable software.

Outcome

The end product of this project consists on a web application named POLYLINGUAL.
The name of the application comes with the meaning “Many Languages”, (Poly from Greek Polus = Many and Lingual = Language), since this was to be developed for the Modern Language Center.
The web application was build using a combination of HTML, CSS, Javascript, jQuery, AJAX, JSON, PHP and MYSQL along with some APIs that enabled us to have some advanced features.
The main system diagram with the main features and pages is presented below:



In general the front-end is modern and minimalistic based on a “flat design” approach. The color scheme selected is a mixture of both white and green. White is for simplicity and green has the meaning of life, harmony and energy. The color combination is to enforce the idea that Polylingual is a friendly and engaging language learning platform.

The admin side (Teacher User) has the following functionalities and features:
The admin can create classrooms to organise group of students based on a specific language better. Each created classroom has an unique generated invitation code that the lecturer can send to the student so they can join their respective classroom. Inside each classroom the admin will have a personal dashboard that contains an analytic graph to show classroom activities for past 7 days and a table with all the students enrolled.
From this page, the teacher can then move to the assignment creation page where he/she can create assignments and set them with a custom overall score for gamification purposes.

When the admin enters the assignment, he is presented with a builder where he can create different type of custom interactive exercises. The types of exercises that can be created are single choice questions, multiple choice questions, drag and drop, sentence reordering and fill-in questions. These type of exercises were made taken into account the clients needs of being able to build them for different level types of students (Basic to Advanced).

The standard user side (Student User) has the following functionalities and features:
Students can join one or more classrooms using the invite code they received from their lecturer.
After this the students will be presented with a dashboard containing the assignments for the respective classroom. Inside this page they can choose which incomplete assignments to take or review ones which has already completed . After entering an incompleted assignment, the students will be presented with the exercises made on the admin side with some of the questions randomised or rearranged for a better learning experience. After submitting the answers, the students can view the result of the completed exercises to check for correct and incorrect answers and will get a calculated score based on his performance.
The students can also edit their own profile information or change details regarding their account.

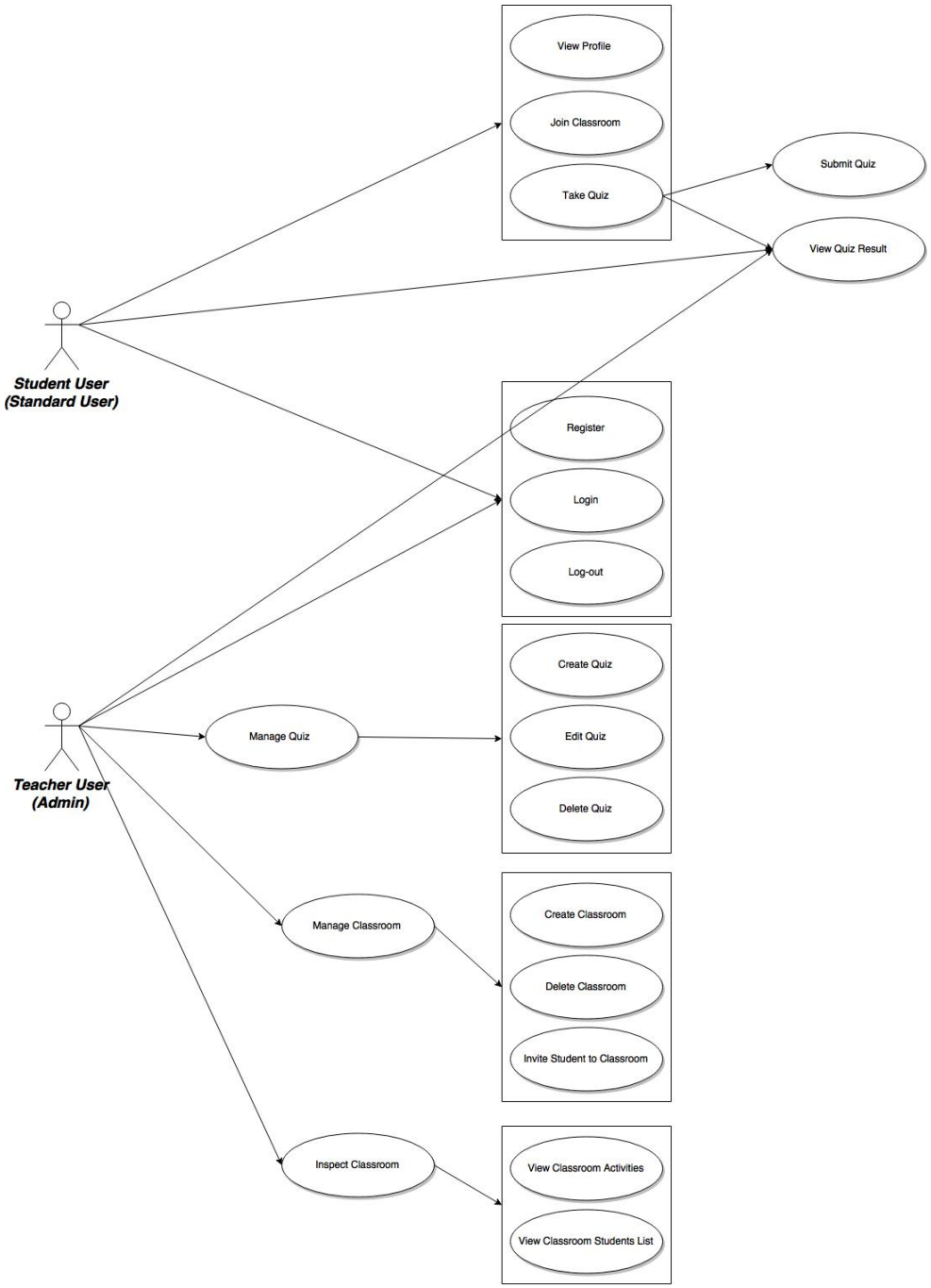
APIs, external services and existing code used:
Cleverbot API - An AI Chat bot that can respond in multiple languages based on what the user inputs. We believed this was a really nice extra feature to enhance the student user experience and this was confirmed by our client as well when we showed it.
The chatbot can be a side tool for students to practice conversation in different languages.

Chartjs - Chartjs is a open source HTML5 Javascript Charts plugin. This was used to show the classroom activity graph in admin's classroom dashboard page. Activities will be recorded and updated on the database, and the Chartjs chart will get the most updated data and display them accordingly to the lecturer.

Bootstrap - Open source CSS framework used for the front-end of our Polylingual web application. This helped our task of making sure the web application worked in different type of devices.

Design

The graphic below is the use case diagram of Polylingual web application. A use case application demonstrates the usage senarios of all users of the web application. This includes the use case of teacher admin and student standard users. The use cases are the oval, users/actors are the stickman and arrow represents what type of user can or will interactive with the use case pointed at. There are common use case where both type of user can interactive with. For example, the login use case is shared because both user need to login inorder to perform other tasks on the website. But as the diagram shows, some use case are only for a paticular type of user. For example, only student user can take a quiz and only admin user can create a quiz.



Software Quality

We acknowledge the importance of ensuring the highest quality standards that we could offer.
Our team has conducted test and inspections in different areas and in various ways. Our approach was to conduct each test two times during the development of the project.
The first time was a basic test right after a new function or feature was implemented. This ensured the basic functionality of this feature. These tests were usually done with different type of inputs/interactions to ensure no error could occur when the end user used our project.
The second time we test them again thoroughly on the whole web application near to the end of the project. We went through different types of student and admin use case scenarios to ensure that the system would be reliable and bug free.
This approach ensured all areas on the web application work correctly together and it reaches the quality standard for deploying to the client as a complete product that could be released for the Modern Language Center with just a few tweaks.

To ensure that no quiz could have its answers by inspecting the web page we made sure that every answer was stored and could only be interacted through PHP. This way no student could check for possible answers making the web application secure and reliable. (Denis Saidov, Marco Pacheco Costa and Yin Chan)

The web browser's inspection mode was a very useful tool for debugging and testing the web application. It was extremely helpful when testing and refining CSS styling and HTML elements on each page. We also used the W3C CSS Validation Service, which is a free online validator for CSS to find any syntax errors or wrong CSS statements so we can debug the CSS file of our web application (Chun Chan, Denis Saidov, Kwo Lee). For testing on PHP and Javascript areas, we used console log and alerts to output the results and error messages for debugging. This allow us to test if the php retrieve and store data to database correctly (by Marco Pacheco Costa, Denis Saidov). For Javascript, we tested on our javascript code that generates quizzes to see whether it can successfully create or display the quizzes correctly (Yin Chan, Chung Tsang). The security aspects including authentication and validation on user accounts was tested too. This was done on the register & login page and database to make sure user passwords are hashed and SQL injection is not possible (Chung Tsang, Denis Saidov, Marco Pacheco Costa and Yin Chan).

Team Organisation

The team's organisation had three stages throughout the project. They were based on the progress status of the project. The team discussed that we should have a basic front-end first, so that we could develop the back-end functionalities over them. Therefore, at the beginning of the project we all started to work on creating the basic structure of the web pages. After the basic front-end structure is mostly ready, we decided to split the team so we have some team members to work on the back-end to get started. The front-end team included Chun (Leon) Chan, Martina Naydenova and Kwo (Frankie) Lee. The back-end team included Chung (Wesley) Tsang, Denis Saidov, Marco Pacheco Costa and Yin (Evan) Chan. After the front-end team was ready, we joined the back-end team to work on the important features and functionalities of the web application. This formation lasted throughout the second half of the project timeline. We experienced this approach gave us more flexibility to adapt the different stages of the project. Also manpower can be utilised more efficiently in this manner.

Slack is our main line of communication between team members. It is very accessible because it works perfectly on web browser, desktop program and mobile apps. This enable the team to be up-to-date and available to respond regarding to the project. It is very easy to share code and screenshots when discussing new ideas or debug problems together. We also have team meetings and coding sessions which allowed us to communicate and share thoughts on the project. Team meeting was aimed to take place at least once a week. Coding sessions took place at least once every week, allowing each other to communicate to work together and help each other.

Other Information

URL: https://www.kcl.dentymedia.com/

Admin Account: polylingual@kcl.ac.uk

Password: //KCLadmin369// (please exclude //)

Student Account: student@kcl.ac.uk

Password: //KCLstudent248// (please exclude //)