**Teach Yourself**

**Documentation**

**Content**

1. Introduction

1. Task force

2. Project Aim and Purposes

1. Project Description

1. Technologies

2. System Requirements Specification

1. Planning
2. Architecture
3. Management
4. Realization
5. Frontend
6. Backend
7. Contact and Helpdesk
8. Summary
9. **Introduction**

Nowadays the world is ruled by technology, because everyone is using it. We are under its power. We are doing our shopping most of the time online, we learn online, we chat with our family, and friends online. As TeachYourself team, we wanted to create an application that helps students in organized learning. The TeachYourself application is a platform that gives the opportunity to the user to organize their information. If you try our product, you have the advantage of store your questions/problems and their answers as well.

1. **Project Aim and Purposes**

Our project’s goal is to make learning easier by creating a smarter tester that would help students learn by noticing questions, and then to find a perfectly suitable answer to it. You can group your questions by thematic. This means that if you create a class with sport thematic, in this class will be questions about sport (athletes, championships, evolution of sports). This method will help the user to meet with same question more often and like this memorize much better the answers.

1. **Task Force**

The task force consist of Szabó Huba and Fábián Ervin under the supervisorship of Szántó Zoltán.

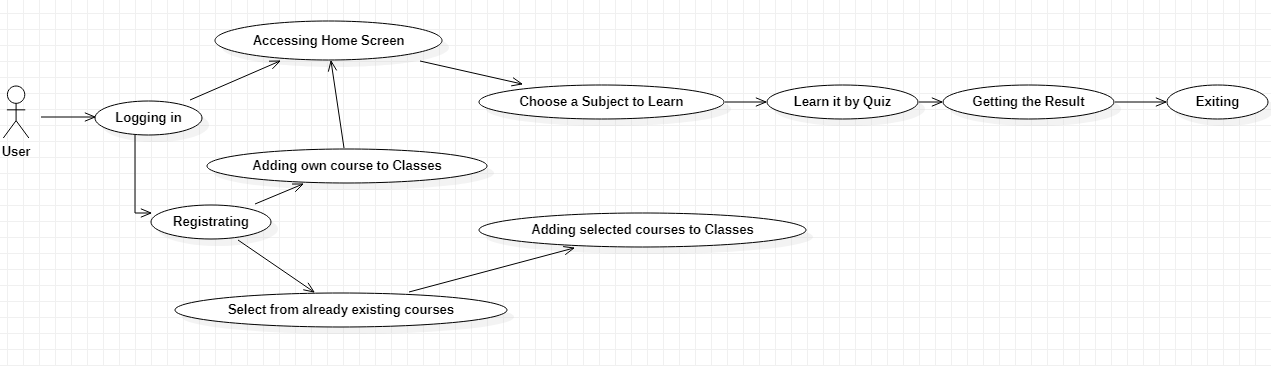
1. **Project Description**

At the planning of the project one of our most important point of view was to implement the whole project in Python programming language for developing and deepening our knowledge in that aspect.

1. **Technologies**

We worked on the project on Windows 10 operating system using Python programming language in PyCharm environment. The Graphical User Interface was made with Kivy Framework a Python plugin. For database we used SQLite3.

1. **Requirements**
   1. User Requirements



The image above is the Use-case diagram of the software which shows functionalities which are available for the user.

* 1. System Requirements

1. Functional Requirements

Here we list the most important requirements:

* Registration/Logging in:

The user must register to be able to find and use the classes created by him/her. For registration we need a username, an email address and a password.

If the user is successfully registered then could his/her account by logging in.

* Selecting a subject you would like to learn:

The user after logging in or registering have access to the Home menu, where he is able to select a subject which he/she would like to learn.

* Learning the Subject:

If the user has selected the desired subject, then he/she can iterate through the 10 questions belonging to each subject, and must answer all of them. At the end of the quiz the user get the result of the quiz and can get back to the home menu to try the test again.

* Getting the Result:

After successfully finishing a test (which is the way of learning) the user gets result of the test and if it is not perfect (10 points) the user can get back to the home menu and try again.

* Exiting when the test are done:

At the end of the test the user can close the software.

ii. Non-Functional Requirements

* The product fulfill the obvious non-functional requirements like

Effectiveness and Reliability. The software was tested in many aspects but we cannot close out the opportunity of a crash.

* The product is runnable on Windows Operating Systems.
* GitHub version control.
* The product do not reveal any personal information to outsider items.

1. **Planning**

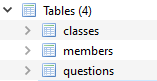
The architecture of the software is approximately simple, it consists of the running software which runs on the PC and the database.

The main goal is to create a software which already contains courses which can be learned, but give an opportunity for the user to create his own classes with specific questions in that topic.

The software then should be able to distinguish which questions mean a challenge for the user and ask them more frequent.

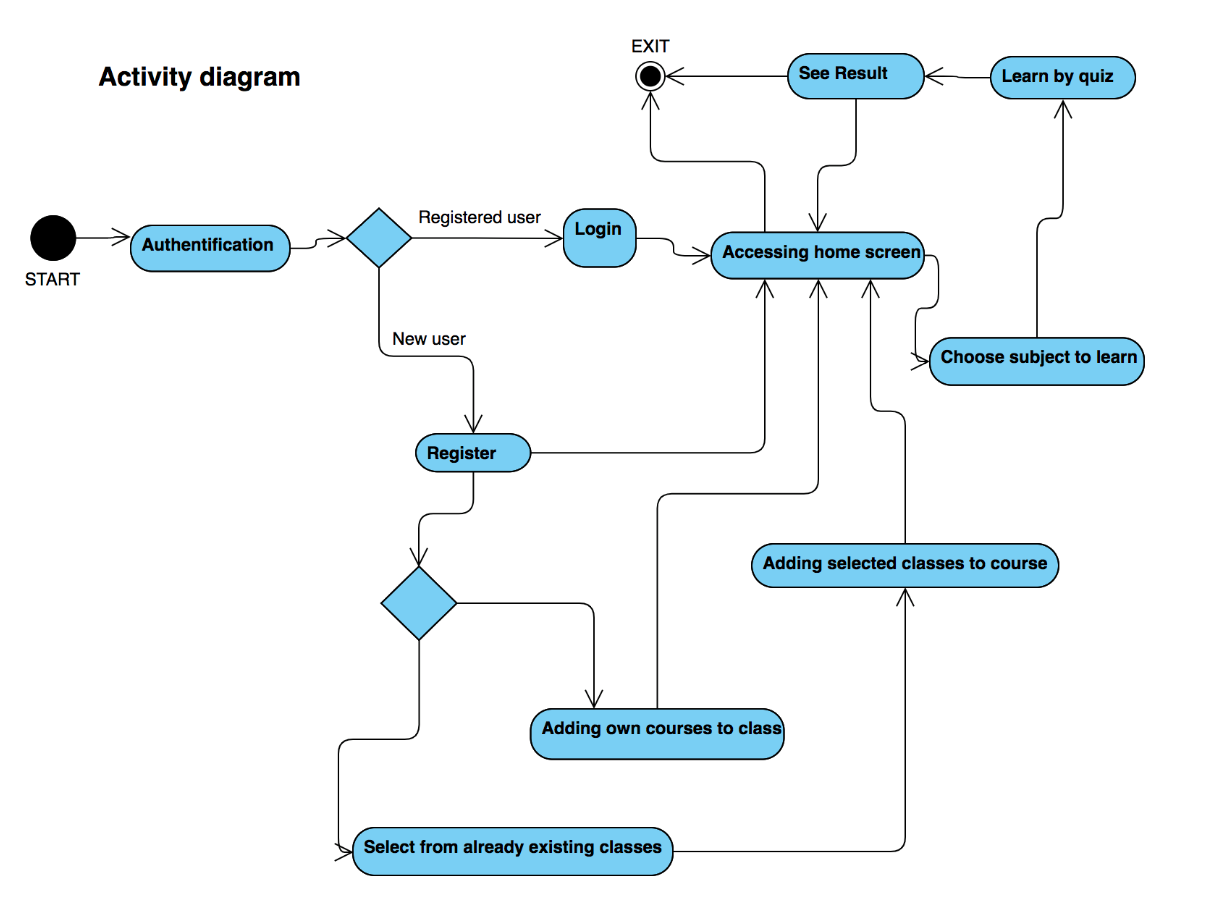


1. **Architecture**
   1. The Database



The Database schema used by us consists of three element:

* + - Members table: contains the already registered users. Its fields are a username, a password and an email address.
    - Classes table: contains the available classes for the user. Its fields are a class\_name and a username.
    - Questions table: contains the questions of each class.
  1. Activity Diagram



1. **Management**

The managing of the project was done in the GitHub-Kanban Board environment, controlled with three threads:

* what we need to do in the future
* what is in progress
* what is already done

Because we worked as a team we must have used version control. The version controlling was done with Git and GitHub.

For better synchronization we weekly kept meetings a day before the online class.

1. **Realization**

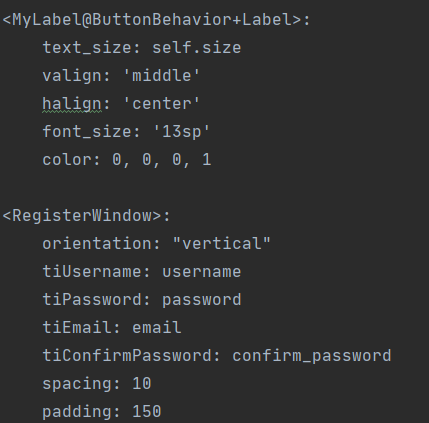
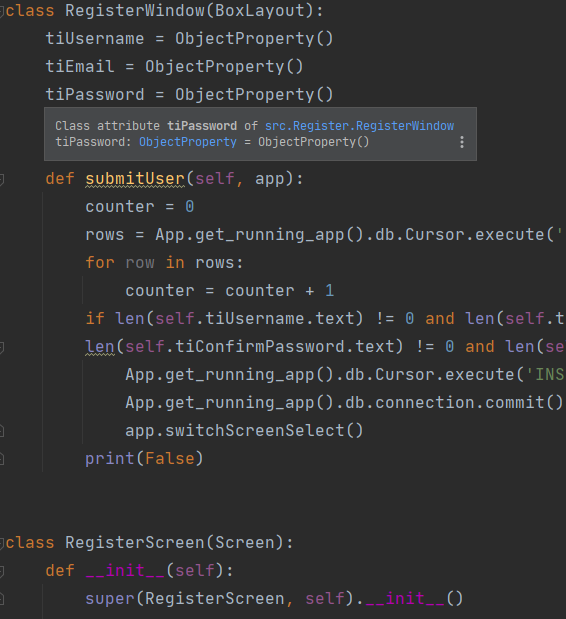
The whole project was written in Python programming language in PyCharm environment.

Our main goal was to develop our knowledge and understanding in this kind of technology.

The part of the User Interface and the backend logic is really blended, so its approximately hard to distinguish them. We used object oriented way of programming.

The functionality of creating and adding new courses is not yet implemented because we had a lot of impediments at the beginning of the workflow due to the our very low level of knowledge in this technology. So we do not reach to the level on which the software could ask the questions which cause hardness more frequently.

1. **User Interface**

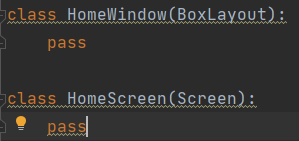
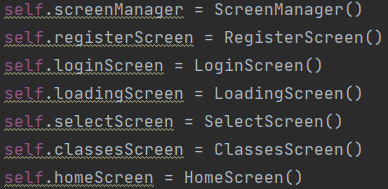
 

The User Interface was created with the help a Cross Platform Phyton Framework called Kivy. The images above shows that how some elements of the User Interface were created.

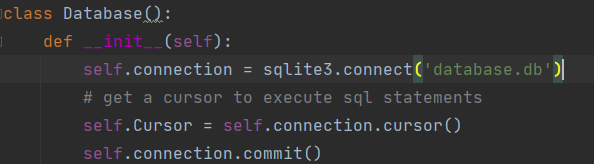
The widgets are placed on a BoxLayout so the whole User Interface is responsive.

1. **Business Logic**

The Business Logic of the project was created in the Object Oriented way of programming.

The visible parts of the User Interface are the screens which are managed by the ScreenManager, it changes the current screens. The child of a Screen is a Layout class, on which the used widgets are placed.



The connection to database.

1. **Contact and Helpdesk**

For any additional information please contact us on the following email addresses:

[ervinfab@gmail.com](mailto:ervinfab@gmail.com)

[huba.sz23@gmail.com](mailto:huba.sz23@gmail.com)

or on the following mobile numbers:

0740838765 Fábián Ervin

0728871525 Szabó Huba

1. **Summary**

While we were working on the project we have experienced both the ups and downs of working

in a team. We have successfully created a software which is able to help its user to learn subjects, unfortunately in a very basic way.

We are sure aware of the fact that the implemented functionalities are very primitive but there are stable base functionalities and lot of personal experience on which we can build up more complex functionalities.

Our goal is to implement the functionalities which are mentioned in the Requirements part, but are not yet realized.