Smart Exam Administration System

Final Report for

ENGR 497 Global Design Project I

Ali Emre ÖZ

Fatih Çağatay GÜLMEZ

Muhammed Yasin YILDIRIM

Ali ÇAKMAK, Assistant Professor

College of Engineering

Istanbul Şehir University

January 26, 2018

Abstract

Smart Exam Administration System, also known as SEAS, develops a computer-based examination system for avoiding paper usage, providing fast and reliable evaluation processes, preventing cheating issues and creating student-friendly environments during exams.

Table of Contents

Introduction

SEAS is a 2-side application which consists of a Server-Side and a Client-Side. In Client-Side, there are two different GUIs for both students and instructors. Within the GUI for students, users are able to see questions, type answers and test them whenever they need. At the same time, instructors have the opportunity to monitor the information gathered during the exam. Server-Side is responsible for compiling or testing the answers and analyzing collected data to reveal various problems such as cheating.

The objectives that the project team consider are providing reliable programming-based exams due to writing and testing codes on a paper leads some problems at the end of the day, more comfortable exam environments for students since the exams are supposed to be a test for their knowledge rather than their attention etc, making evaluation processes easier and less time consuming as well as giving a chance to improve both instructors and students themselves by using analyzed statistics provided by SEAS.

For reaching that objectives, the team’s motivation is mainly focused on providing a better evaluation of knowledge at first. Thereby, lectures’ efficiency might be increased through more accurate statistics. Besides that, a modern approach to examination might lead to improve brand values of educational institutions and decrease their paper usage which makes archiving become much easier than ever before.

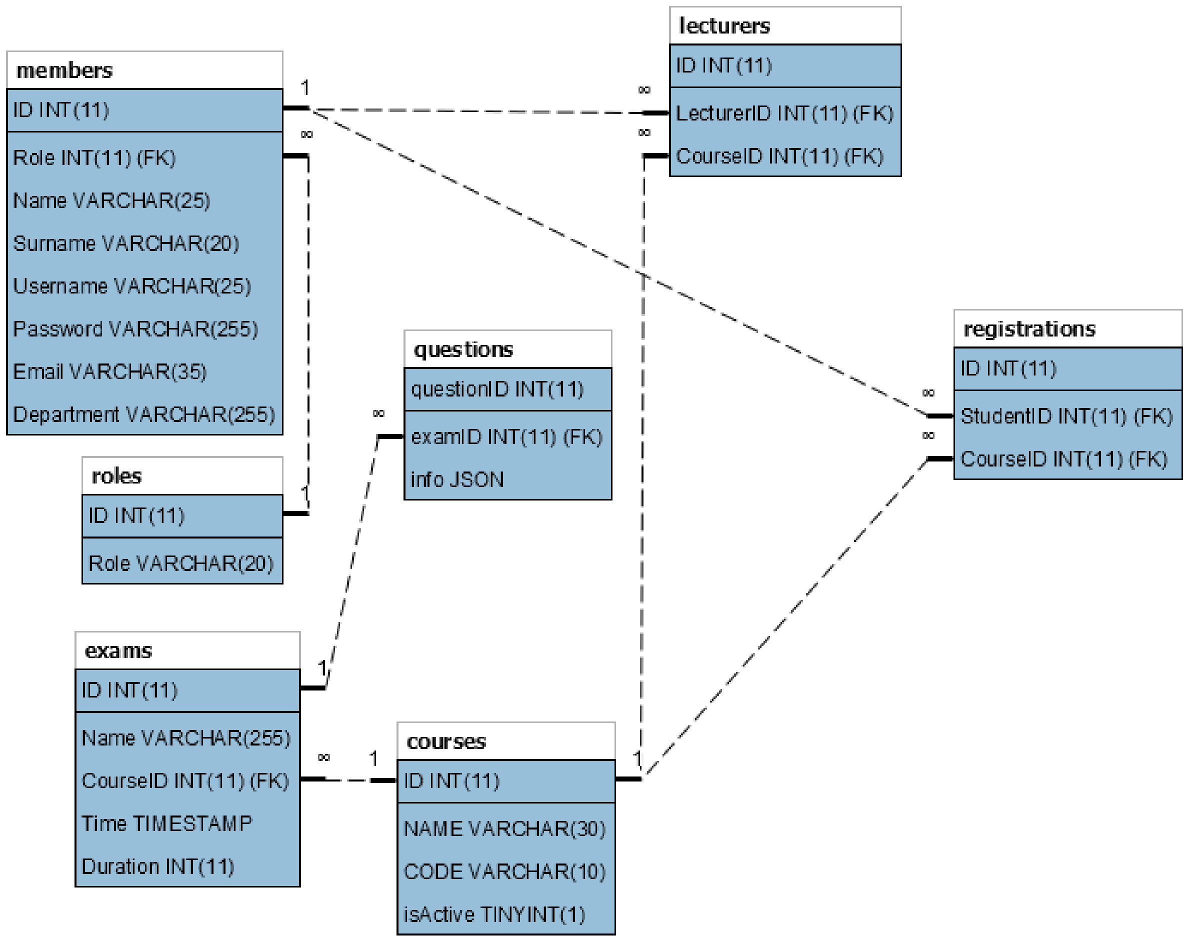
Background

The main idea of the project is the same with one of the previous Global Design Projects that Jareth Conley Moyo, Ibrahim Hamadjoda and Asad Boolani tried to develop. Design of the project is simple as SEAS, a Server-Side and Client-Side app offering various services for both students and instructors. However, this project couldn’t get completed and implemented as a working system. Yet it might be a guide for SEAS when machine-learning-related works get in, it contains some algorithms to prevent cheating as planned for SEAS.

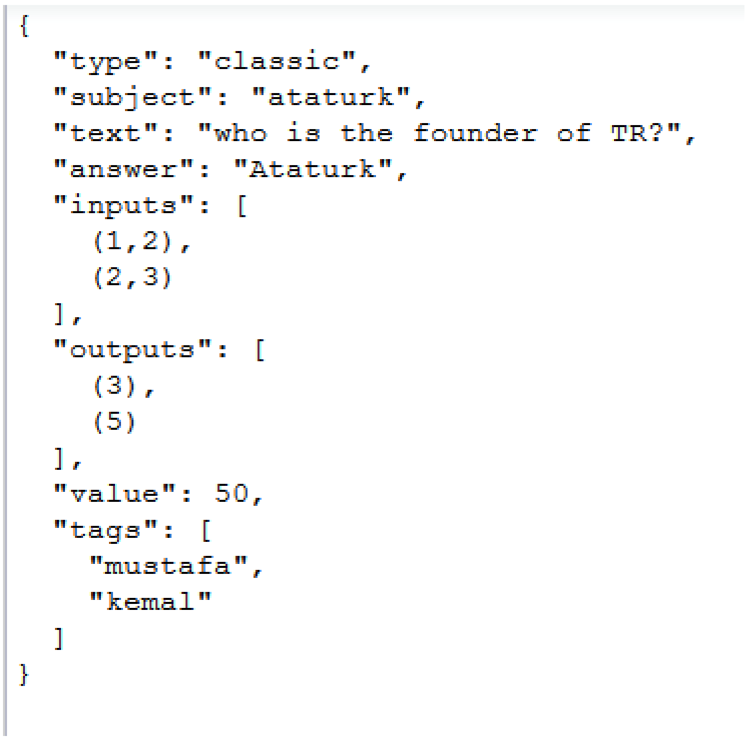
Another similar system is Codio which provides a learning experience, that is why it authorizes teachers and students with its technology and learning resources. Codio believes that students and teachers are individuals, teaching and learning styles are different. It offers to enable teachers and faculty to support and master the skills and achievements of students at their own pace by spending 20% more time teaching and 20% less time administering using the provided grading tools. It's integrated with some LMS platforms such as Canvas, Blackboard or Moodle. Besides, Codio claims that it fits to any programming language, any version, any database, any software component to use on its online IDE.

Lastly, there is that Inginious which is also similar to SEAS in some terms. It basically provides a coding ground for testing and evaluating code pieces. However, it is not designed for real-time exams. Thereby, it is not compatible for academic usage. The lack of security and statistics is also an important issue which SEAS overcomes. Inginious, like Codio, is a web-based application and contains Moodle & edX integrations.

Body of Report



Server-Side corresponds to 2 different parts connected to each other, MySQL Database and Flask RESTFul API. The database runs on localhost of the server meaning that database is not connected to the internet directly. The server connects database to its clients in terms of some rules and provided APIs. Database elements are organized as schemas and tables. Each organization has its own schema and this schema is a container for tables of that organization. The tables are members, courses, lecturers, registrations, exams, roles, questions. The members table holds all individuals' information; the columns are respectively ID, Role, Name, Surname, Username, Password, Email, and Department. The Role columns here is a foreign key placed into ID column of Roles table which consists of ID column and Role column. The courses table holds all information related to courses that uses SEAS; the columns are ID, Name, isActive. ID columns is course ID generated by Database automatically and will be used as foreign key in further tables. The registrations table holds the registration information for each student; the columns are StudentID and CourseID. StudentID and courseID is foreign keys referenced to members.ID and courses.ID columns respectively. The lecturers table holds the lecturer-course information; the columns are LecturerID and CourseID which are foreign keys referenced to members.ID and courses.ID respectively. The exams table has ID, Name, CourseID, Time, Duration fields. The ID column is auto generated by database and a source for exam foreign keys. CourseID is a foreign key coming from courses.ID column. Time column of the exam table holds TIMESTAMP type, which is a universal time variable, for everywhere the exam start time will be at the same moment. The questions table is pool for all questions of organization and is connected to exams table with a foreign key column called examID sourced to exams.ID column. The other columns are questionID and info. QuestionID is auto generated by database. The info column holds JSON datatype and this field is for all information about the questions. The JSON fields are type, subject, text, answer, inputs, outputs, value and tags. Type is questions type states if the question is a test question or a classic one or a programming question or etc. Subject fields is for questions subject and tags are for question classification by user, these would be necessary when we analyze exam statistics at further steps. Inputs and outputs are for programming questions, for other types they will be a none string. Text is questions text part including options of test and answer is for represents correct solution. Value is point value of the question. Below an example can be seen. Input/output fields are useless and only for being an example.



```
{
  "type": "classic",
  "subject": "ataturk",
  "text": "who is the founder of TR?",
  "answer": "Ataturk",
  "inputs": [
    (1,2),
    (2,3)
  ],
  "outputs": [
    (3),
    (5)
  ],
  "value": 50,
  "tags": [
    "mustafa",
    "kemal"
  ]
}
```

In API level, SEAS uses Flask Microservice for Python 2.7. This API provides some key functions to users. API is basically able to handle Organization Sign Up, User Sign Up for a registered organization, User Sign In / Out and so on. In addition, adding a course to an organization, getting a course's information, registering students to a course, getting course student list, getting lecturer's all courses, changing password, deleting a student from a course, creating an exam, getting exams of a course, getting exam information including questions APIs are provided. All API's returns JSON's and works with special URLs and RESTFul requests.

The most important topic in the security part is that, students can use their computers during exams. So, SEAS must consider what they are able to do on their computer and corresponding enhancements to limit them. Within this regard, more than one method used in the security part and improvements made to prevent cheating at the highest possible level.

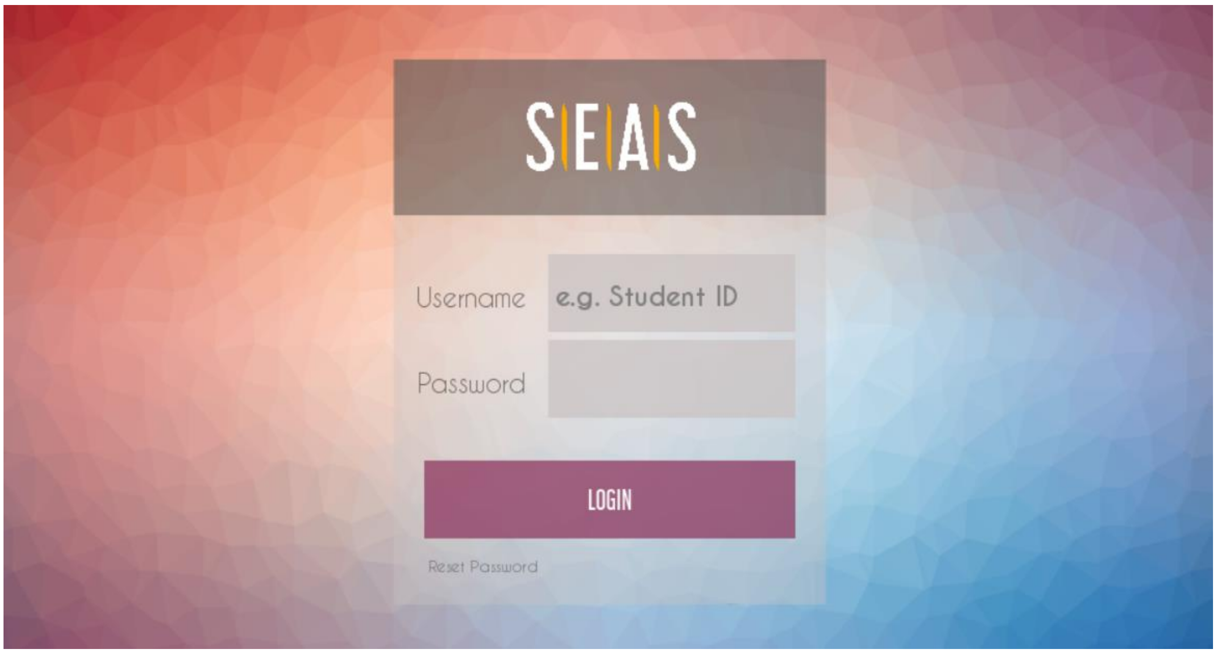
As the team, we tried to block the students' access to internet by using very basic networking methods in order to prevent them surf and cheat during exams. However, as the result of our research, we realized that it is even easier to overcome this issue. By keeping the software always on the top while exams go on, SEAS prevents the use of any external program other than itself. Apart from this approach, SEAS is also able to maximize the security by blocking some basic operating system shortcuts (e.g. alt + tab, windows key). In addition to that, we decided to send warning messages first when students try to use these shortcuts or even try to open any external program. Lecturers get informed on such actions taken as well.

SEAS also stores records of students' pressed-key logs for keeping track of what they do during exams. With some enhancements, the software gives accurate feedbacks to lecturers on live-exam-security by using various machine learning approaches analyzing those logs.

Since SEAS aims to run on different operating systems, the libraries that we used for developing is compatible for various systems. Within this regard, GUI is designed to have a dynamic and responsive layout. Although the programming language used in SEAS is Python, Kivy open-source module is the base of GUI.

GUI consists of separate screens such as Login Page, Start Page, Lectures Page and so on. Each of these pages are compatible for all operating systems even including mobile ones such as Android, iOS etc.

Design of the pages are done by 2 files, a ".py" file that contains required functions and a ".seas" file that determines where to place what and how to look. Each page is connected to the main file of GUI, which is called "gui.py" and located under the GUI directory.

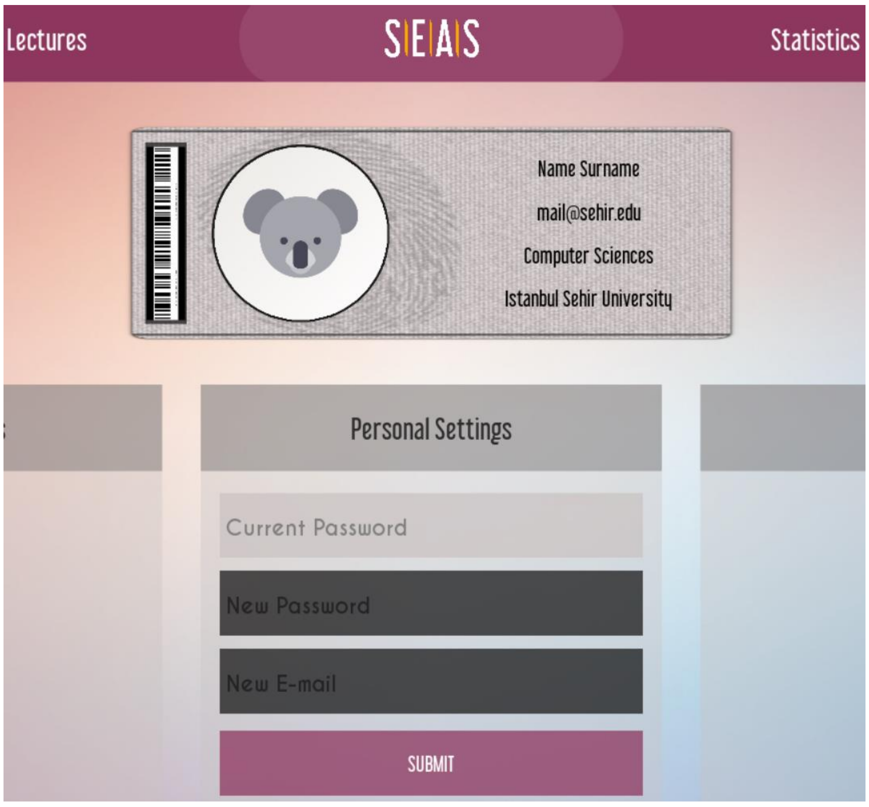


Relation between GUI and security-provider files is established through threading. When a lecturer starts an exam and students start to join, threading gets active and runs security-provider files to prevent specific unwanted actions as stated above. Besides, GUI also contains some security stuff like warning when ESC pressed or logging if the current screen is left somehow.

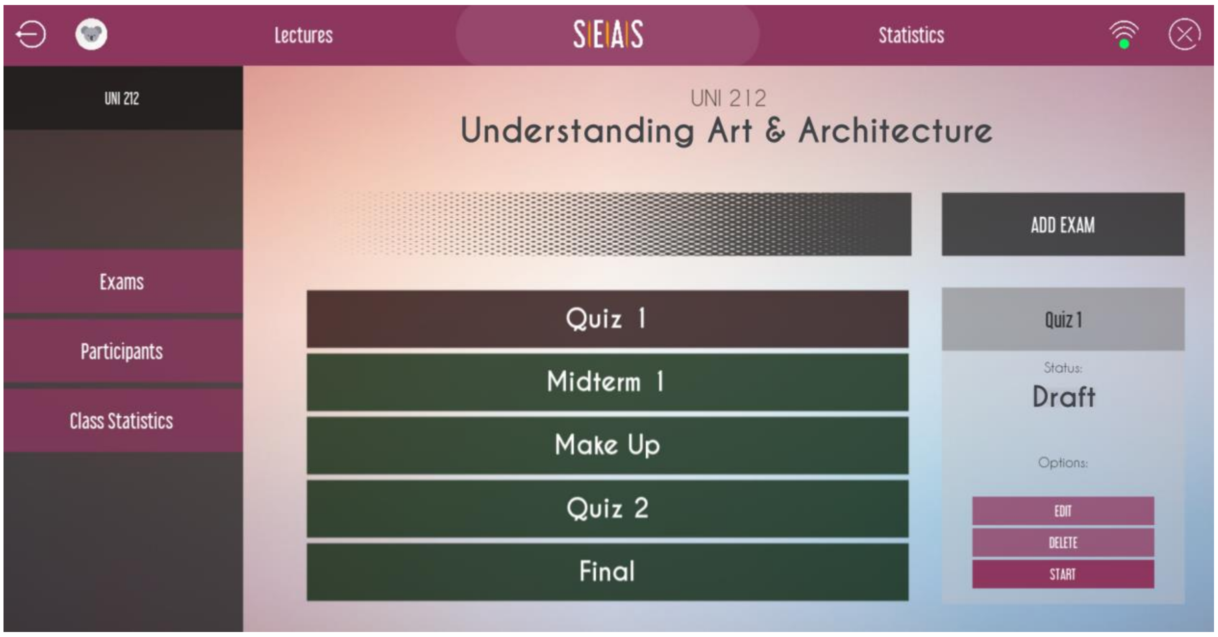
Communication between GUI and Server-Side is established through provided APIs. For instance, when a student wants to sign in, GUI takes the username & password inputs and send them to Server-Side via the API. According to the returned value, student gets in or warned. Those kind of returned values get stored in local ".seas" files temporarily created for each action such as sign in, registered courses etc.

Additionally, GUI enables implementation of different libraries, formats or even languages such as Matplotlib, Json and so on. Within this scope, Matplotlib is used for graph-drawing even though Kivy has its own feature called Garden.

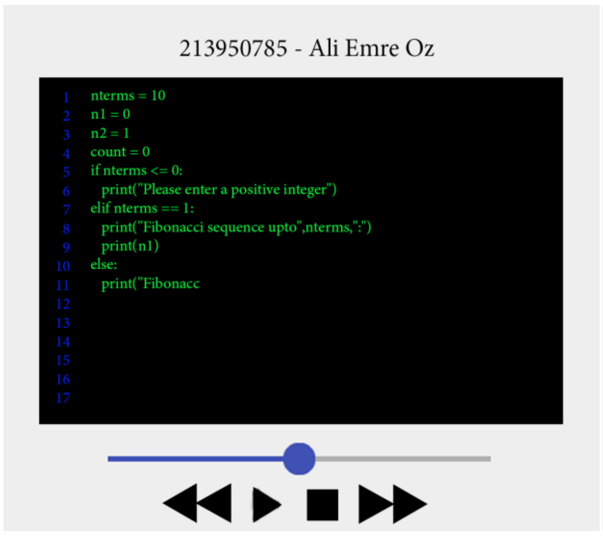
In detail, the welcoming screen of the program which is Login Page consists of some widgets as is shown in the above picture. This page allows users to sign in as mentioned as well as resetting their passwords by their e-mail addresses. After logging in, users have a chance to change both their passwords and e-mail addresses on Profile Page where they are able to track their last logins and latest activities on the program. Database and GUI collaborate in order to provide these services.



When users successfully sign in to the system, Lectures Page appears as the landing page for instructors. Right before entering the page, SEAS communicates with database once again and temporarily stores the courses that the instructor currently gives in an ".seas" file. Hence, the instructor becomes able to select courses on the dropdown menu located at the top-left of the screen. When a lecture is selected, the instructor gets buttons on the left menu enabled, so 3 sub-pages which are Exams, Participants and Class Statistics become available. On Exams, all the exams which belong to the selected lecture are listed along with given options such as Add Exam, Edit, Delete, Grade etc. while on Participants, students registered to the course is listed as well as various options like Import Students List, Add Student, Delete and so on. For students, designs and the options put onto the pages are different from the ones that instructors have. SEAS interface for students is not as complicated as it is for instructors. When students log in, the options provided are mostly focused on exams. Other than that, statistics are also prepared for their service. However, what SEAS offers to students is mainly taking an exam in simple terms.

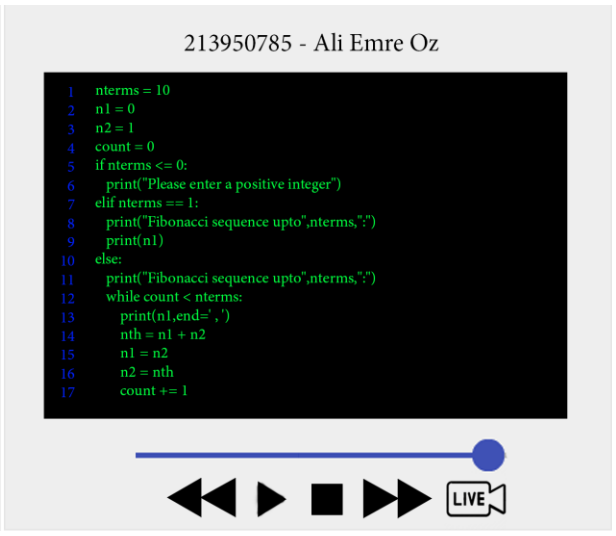


SEAS, by the way, includes multiple features for ensuring safety during the exam. One of these is storing student keystrokes. With this keylogger based approach, suspicious activities can be monitored and warning can be sent to both student and teacher when it is necessary. The pyHook library is used for these operations. This library's OnKeyboardEvent module follows the keyboard operations. A plus aspect of storing these keyboard keystrokes is that, it helps to create new features for teacher. When SEAS stores which key is pressed, it also stores which key is pressed at which time interval. With these two data, it is possible to tell how active the student is during the exam. While SEAS showing this activity to lecturer, line graphs were used. The y-axis of graph shows the activity of the student, while the x-axis shows the time. In addition, with the further development of this application SEAS now has a new feature which is teacher will be able to see answers from a selected student when the exam is going on. With this live monitoring approach, the teacher has the opportunity to intervene instantly to the suspicious student. In addition, the answers of the students who are on monitoring will also be indicated by the rewind feature. The teacher will be able to see what he or she did during the 5th and 10th minutes of exam, in the 30th minute of the exam. In addition to forwarding, rewinding, playback and stopping, teacher can go to the desired minute with a slider. While these features were being developed, the Clock module of the Kivy library was helped.



213950785 - Ali Emre Oz

```
nterms = 10
n1 = 0
n2 = 1
count = 0
if nterms <= 0:
   print("Please enter a positive integer")
elif nterms == 1:
   print("Fibonacci sequence upto",nterms,":")
   print(n1)
else:
   print("Fibonacc
```

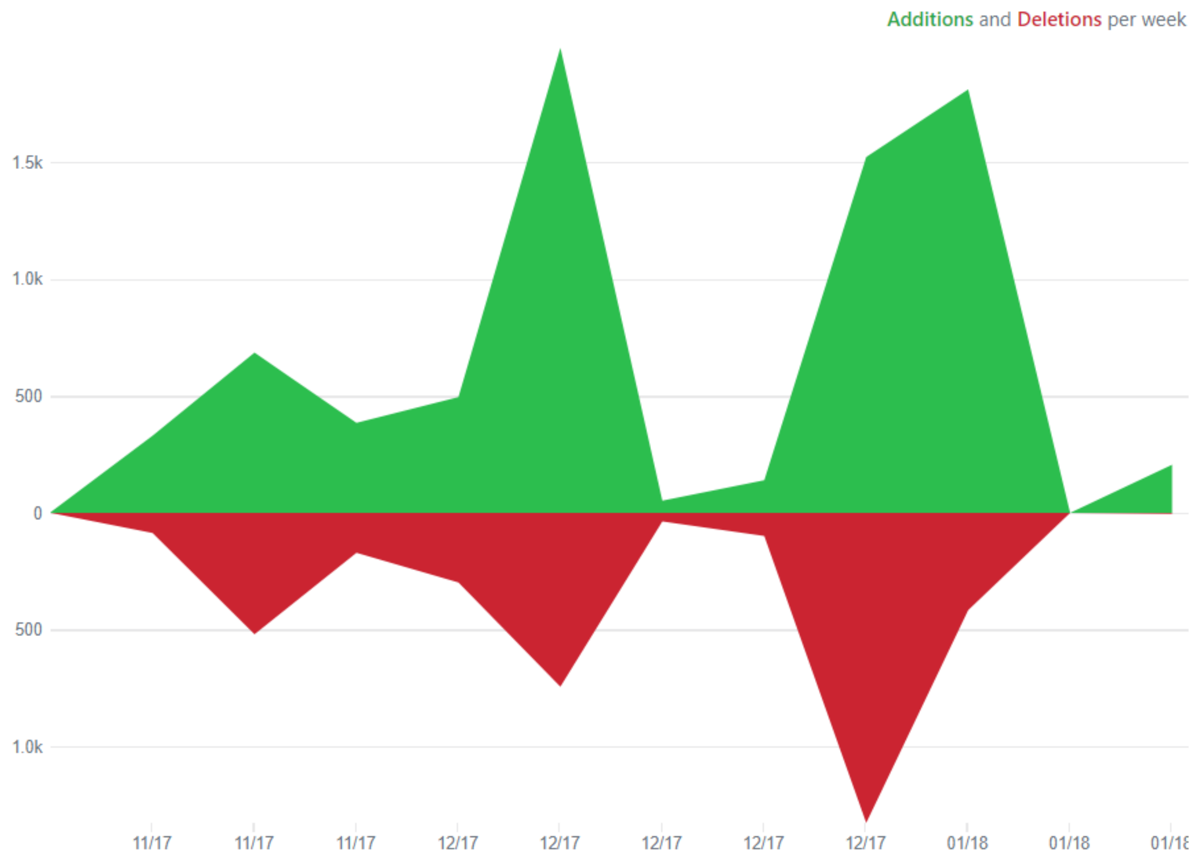


213950785 - Ali Emre Oz

```
nterms = 10
n1 = 0
n2 = 1
count = 0
if nterms <= 0:
   print("Please enter a positive integer")
elif nterms == 1:
   print("Fibonacci sequence upto",nterms,":")
   print(n1)
else:
   print("Fibonacci sequence upto",nterms,":")
   while count < nterms:
       print(n1,end=' , ')
       nth = n1 + n2
       n1 = n2
       n2 = nth
       count += 1
```

Source Code

In order to see what is programmed so far or updated, the source code of SEAS can be visited on this GitHub repository: https://github.com/wivernsoftware/SEAS



References

- https://codio.com/docs/
- http://inginious.readthedocs.io/en/v0.4/
- https://kivy.org/docs/api-kivy.html
- https://dev.mysql.com/doc/