

**ISTANBUL TECHNICAL UNIVERSITY  
FACULTY OF COMPUTER AND  
INFORMATICS**

**AUTOMATED CODE AND CONTEST  
EVALUATION SERVICE (ACCES)**

**Graduation Project Interim Report**

**Kadir Emre Oto  
150140032**

**Muhammed Burak Buğrul  
150140015**

**Süheyl Emre Karabela  
150140109**

**Department: Computer Engineering  
Division: Computer Engineering**

**Advisor: Asst. Prof. Ayşe Tosun**

December 2019


## Statement of Authenticity

We hereby declare that in this study

1. all the content influenced from external references are cited clearly and in detail,
2. and all the remaining sections, especially the theoretical studies and implemented software/hardware that constitute the fundamental essence of this study is originated by our individual authenticity.

03.10.2019

Kadir Emre Oto



03.10.2019

Muhammed Burak Buğrul



03.10.2019

Süheyl Emre Karabela



# AUTOMATED CODE AND CONTEST EVALUATION SERVICE (SUMMARY)

Coding assignments are very popular not only in computer engineering departments but also in almost every engineering department. Classic process for the lecturers/T.A.s:

- Give the homework statement and a couple of examples.
- Collect students codes by the deadline.
- Evaluate them with a test set and analyze them one by one (most time consuming part).
- Announce the grades.

The 3rd step can take months for a crowded course. But it is generally a deterministic step. All the codes submitted by students should be tested with the same test cases. And every code should be analyzed in same aspects like memory usage, run time etc. An automated system can do this within seconds instead of months. In addition with a fast system, students can have a feedback mechanism. This situation leads students to learn and practice more efficient and T.A.s to save their times.

ACCES targets these outcomes within a scalable, secure and robust system. In addition, it will be a Software as a Service solution for every school to use it easily.

In point of view of a student; when a code is submitted for a homework, process afterwards is unknown. Which type of test cases failed, which type of test cases passed, how was the performance of the code etc. Even he/she can not know whether the code run or crashed in the T.A.s' computer. But in ACCES, run environment of the projects are known and fair for every student. A student can run his/her code before submitting the homework to analyze its' behaviour. After submission, he/she can see which test cases passed and failed. In addition he/she can see the data of test cases if T.A. or lecturer allows. At the end, he/she can see an expected grade of the submitted code. Of course it may not be the final grade.

In point of T.A.s, they can be assigned to courses and manage students and homework of assigned courses. They can give homework and determine test data for the homework. They can split the test data sample(visible to students) and private(only the pass/fail status visible to students) sets. They can add or delete test cases whenever they want. All they need is a working solution for the homework, add/delete parts can be done via ACCES website. When they prepare everything for a homework, they can evaluate submitted codes with one click in minutes instead of months. In classical evaluation processes, changing the test data leads T.A. to reevaluate all of the submitted codes. In acces re-evaluation is a one click operation as well.

In point of lecturers, they can do everything T.A.s can do, in addition they can manage T.A.s.

Besides the benefits from the view point of users, ACCES ensures some technical specifications:

- Security: Every user communicates with the system by using time based authentication tokens and every code run operation handled by the judge is in a separate sandbox.
- Performance: We use the back-end in a functional 'do it and disappear' method in order to shorten the data travel path between users and the system. In addition, with this methodology, every endpoint can be scaled in different machines. This scale option provides handling high request/seconds operations easily.
- Availability: System saves the data in more than one place. And our databases are different services. Other than the database records, we log everything in a separate service.

In conclusion, our motivation to encourage usage of ACCES in schools is providing almost equal opportunities to students, making the code evaluation process as transparent as it can be, making T.A.s job easier and establish a good feedback mechanism.

# Contents

<b>Introduction and Problem Definition</b>	<b>5</b>
<b>Literature Survey</b>	<b>6</b>
<b>Novel Aspects and Technological Contributions</b>	<b>8</b>
<b>System Requirements</b>	<b>9</b>
<b>Definitions</b>	<b>9</b>
<b>Functional Requirements</b>	<b>10</b>
<b>Non-Functional Requirements</b>	<b>12</b>
<b>High Level Design</b>	<b>13</b>
<b>Project Plan</b>	<b>14</b>
<b>Project Resources</b>	<b>14</b>
<b>Work Breakdown and Work Assignment</b>	<b>14</b>
<b>Goals and Evaluation Criteria</b>	<b>16</b>
<b>References</b>	<b>17</b>

# 1. Introduction and Problem Definition

Most universities, especially in Turkey, do not have a fully automated homework or exam evaluation system. An automated system can reduce the evaluation time hundreds of times while increasing accuracy for code assignments in computer engineering and similar departments. For this reason, instructors have to evaluate the student submissions manually (sometimes this process takes 2 or 3 months), and students are not able to get any feedback about their submissions until the grades are announced months later.

In this project we are offering a system that can evaluate the student's submissions within minutes and give the necessary feedback to students about their submission. In this way, we are planning to use this system in coding exams and other similar cases that take lots of time to evaluate. For that matter:

- Students can see their grades and code errors in a short while.
- Students will have the opportunity to correct their codes by receiving quick feedback (at most 5 minutes instead of 3 months).
- Instructors won't waste their precious time doing these chores.
- The probability of human error will reduce to almost zero.

## 2. Literature Survey

In this section, both competitive programming platforms and educational platforms will be addressed.

### **HackerRank:**

HackerRank is a platform that enables companies to determine the skills of developers by providing programming challenges and online environments [1]. Other than business solutions, users can also use the site to solve programming challenges provided by HackerRank in many domains to improve their coding skills and attend to the contests that are organized by them [2]. HackerRank uses a cloud-based code execution and evaluation system which has a similar function and operating method to our code evaluation system. It uses the system to execute the user's codes submitted to the challenges with the predetermined inputs to check if they give the correct outputs based on custom grading functions or exact match to the predetermined outputs. The memory and time constraints for executions can be configured. Our judge system will have similar functionality, it will support the same input and custom grading-based evaluation methods and configurable time and memory constraints. The main difference between the HackerRank and ACCES is that ACCES provides automated homework evaluation system for academic institutions while HackerRank helps companies hire skilled developers.

### **Codeforces:**

Codeforces is an online competitive programming and social network platform [3]. It has a large number of challenges in various difficulties. It holds many rated and unrated contests which the rated ones affect the rank of the participant. It is more contest oriented than HackerRank and do not provide hiring products to businesses. It has a similar online judging system to HackerRank and ACCES.

### **CSAcademy:**

It is a competitive programming website [4]. It contains some problems that require a one file batch solution. In addition, it holds some online contests for competitive programming enthusiasts. The most important aspect of CSAcademy that differs it from other competitive programming platforms is its applications. There are 3 applications in the platform:

- Graph Editor: It allows one to create directed/undirected, weighted/unweighted graphs and change places of nodes. Through this editor, the test inputs can be visualized dynamically.
- Geometry Widget: It allows one to create and replace simple geometric shapes in a 2-dimensional plane.
- Diff Tool: It accepts two texts as input and returns a detailed difference output of these two input texts. Details are like in the vimdiff tool in unix systems with colors. This tool can also be used to highlight the differences in code pieces.

**Calico:**

"Calico is a utility for checking command-line programs in terms of their input and output". It checks whether a program creates the right yield when given a few sources of info. It was created to assess straightforward programming assignments in an introductory programming course. [5]

Instructors of BLG102E Int to Scientific & Eng.Computing (C) at ITU are currently using this python module. But it only provides the exact input-output checking via command line.

**GradeScope:**

It is an automated exam evaluation service [6]. It works on predetermined, partial point milestones and gives notes to students according to them. It can detect handwritings, and it can be used for an online objection system.



### 3. Novel Aspects and Technological Contributions

There are many online code evaluation tools for both commercial and individual use available on the market. They are primarily used as assessment tools aimed at companies looking for hiring talented developers. Many of them are not compatible with homework grading in a course domain. ACCES aims to aid schools in evaluating and managing course assignments in a user friendly environment. It differs from its competitors by being a Software as a Service product.

## 4. System Requirements

### 4.1. Definitions

**System:** A system that automates code evaluation process for computer science related courses.

**User:** An authenticated person:

- **Student:** Can view enrolled course materials and can submit homework and see result of homework via ACCES.
- **Teaching Assistant:** Can organize materials of related courses according to given permission by course owner(Lecturer)
- **Lecturer:** Can create courses, add enrolled students and assign T.A.s to his/her courses. Also can give permissions on course materials to assigned T.A.s.
- **School Admin:** Can organize lecturer/T.A./Student registrations to the system.
- **System Admin:** Can create and assign new systems to new schools. Can monitor system situation and interfere into the system if required.

**Profile Page:** Shows enrolled courses, announcements, user info etc. One can navigate to lectures and related announcement pages from profile page.

**Profile Settings:** A user can change his/her password etc.

**School Selection Page:** A user can select one of his/her current schools.

**Lecture Page:** Shows course info (Lecturer, time, grading etc.) and homework. One can navigate to related pages from lecture page. Also provides course adding features.

**T.A. Page:** Shows lecture info (Lecturer, time, grading etc.) and homework. One can navigate to related pages from lecture page.

**Homework Page:** Shows homework info(statement, deadline, language constraints). Includes an online editor. One can write and submit his/her homework code via this editor. Afterwards he/she can see submission results.

**Online Editor:** Organizes a file system for homework. Students can change permitted fields on this editor. Also this editor provides code colouring.

**Course Message Panel:** Shows message threads created by students, T.A.s and lecturers.

**Course Message Thread:** One single topic page, shows all messages sent to related topic in chronological order. Also, one can submit his/her own message to related thread.

**Course Settings:** Lecturer can change permissions of T.A.s and add/update/delete homework.

**Course Adding:** One with the permission can add course with materials.

**Homework Adding:** One with the permission can add homework with materials (statement, sample inputs/outputs, language constraints, judge type etc.).

**Homework Settings:** One with the permission can change settings.

## 4.2. Functional Requirements

### 1. User Requirements

#### a. Signup and Login

- i. School admins can register students single or bulk with school mails, IDs and random passwords.
- ii. Users can log in to the system via their school mails and passwords

#### b. User activities

##### i. Profile Activities

1. A user can edit his/her profile entities (password etc.)

##### ii. School Activity

1. A user can change his/her current school

##### iii. Student Activities

##### 1. Course activities

- a. Can see his/her enrolled courses
- b. Can see course materials
- c. Can send and see messages on message threads from an enrolled course
- d. Can see his/her grades

##### 2. Homework Activities

- a. Can submit homework documents to a given homework.
- b. Can see his/her homework results

##### iv. T.A. Activities

##### 1. Course activities

- a. Can see his/her assigned courses
- b. Can see/edit permitted course materials

- c. Can send and see messages on message threads from an assigned course
    - 2. Homework Activities
      - a. Can edit homework documents to an assigned homework.
      - b. Can add/edit/delete permitted homework grades.
      - c. Can add homeworks if permitted
      - d. Can rejudge
  - v. Lecturer Activities
    - 1. Course activities
      - a. Can create courses
      - b. Can see/edit course materials
      - c. Can send and see messages on message threads from a course
    - 2. Homework Activities
      - a. Can add/edit/delete homework
      - b. Can add/edit/delete homework grades.
      - c. Can rejudge
  - vi. School Admin
    - 1. User Activities
      - a. Can register new users (Lecturer, T.A., Student) to a school with their emails and IDs.
      - b. Can change a users' type in his/her own school (Lecturer, T.A., Student)
  - vii. System Admin
    - 1. User Activities
      - a. Can add schools and school admins
    - 2. System Activities
      - a. Can monitor current schools status(number of students, number of courses, judge queues etc.)
2. System Requirements
- a. Judge Requirements
    - i. Can evaluate submissions
      - 1. Submissions should be evaluated in an isolated environment
      - 2. Submissions should be evaluated in chronological order
      - 3. The judge system will be distributed
      - 4. Submissions can be re-evaluated based on user request and new test suite
  - b. Data storage Requirements
    - i. Course material
    - ii. Homework material

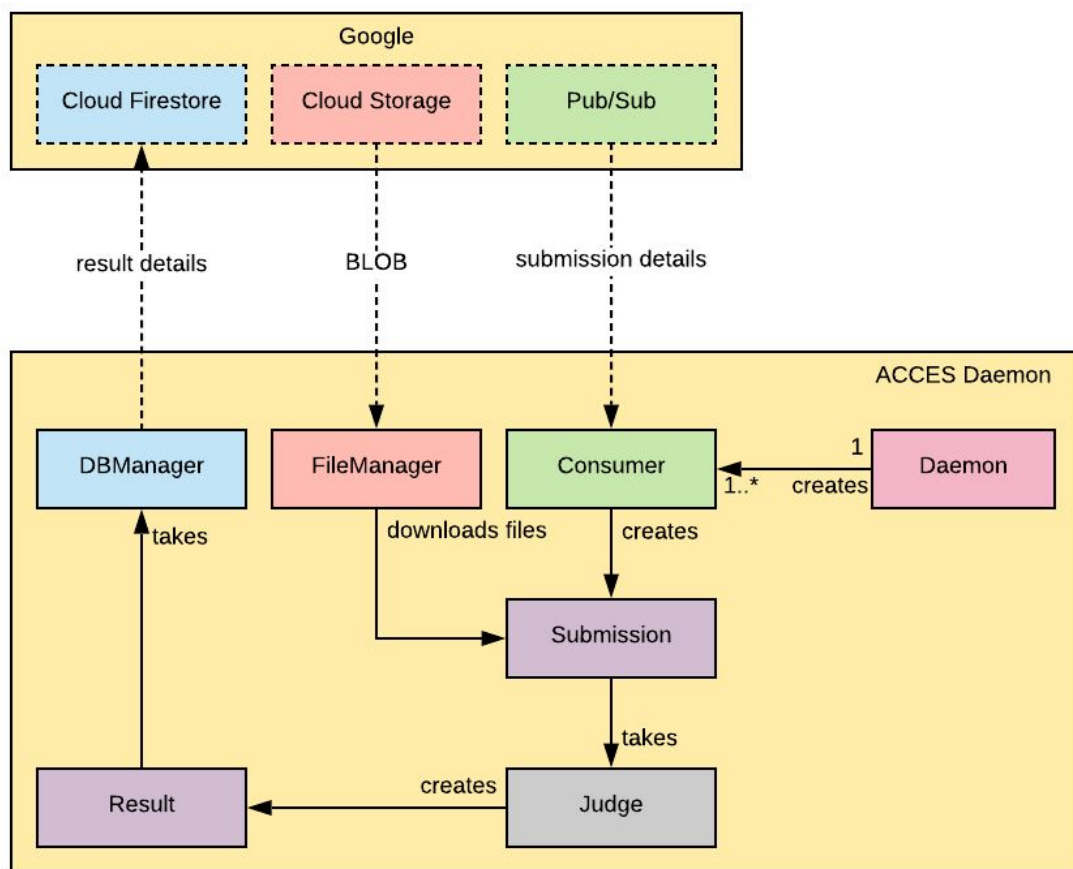
- iii. Homework test cases
- iv. Student submission details (codes etc.)
  - 1. All source codes should be stored
- v. Evaluation results
  - 1. All past results should be stored
- c. Grading Requirements
  - i. Based on judge results and user grading criteria, final grades will be calculated.
  - ii. Grading criteria should support partial and binary scoring
- d. Reporting Requirements
  - i. Grades sent from grading service can be exported based on the export options below
    - 1. PDF
    - 2. csv
    - 3. json
  - ii. Grades can also be shown to corresponding users
  - iii. Grade and submission statistics can also be computed and presented to the related user
    - 1. Time series showing the number of submissions
    - 2. Judge usage (performance, response time etc.) statistics for system admins
    - 3. Test case execution statistics (accepted ratio)
    - 4. Grade distribution for a particular homework or course

### 4.3. Non-Functional Requirements

- 1. Compatibility
  - a. The system web apps should be accessible via using at least LTS versions of Google Chrome, Mozilla Firefox, Microsoft Edge and Safari web browsers
- 2. Security
  - a. The system should not leak personal information of its users
  - b. User codes should be executed in an isolated environment in order not to crash the system
- 3. Availability
  - a. In case of a system failure, the system should be recovered in 30 seconds
- 4. Reliability
  - a. The system should store back-ups weekly
  - b. Static code check analysis will be used to ensure code quality
  - c. All unit tests before release should be executed successfully
  - d. There should not be unresolved critical defect prior to a release
- 5. Performance

- a. The system should support more than 10.000 concurrent users
  - b. The system should have a transparent queueing mechanism for homework evaluations so that the performance bottlenecks can be monitored
  - c. The system should be as much as asynchronous, so that it can have a flawless web app experience for end users
6. Maintainability
- a. The system should have understandable and well structured documentation
  - b. System should be modular, so that new developers can easily add features
  - c. Static code check analysis will be used to ensure code quality

#### 4.4. High Level Design



**Figure 1:** High Level Architecture of Judge Microservice

## 5. Project Plan

### 5.1. Project Resources

- Software
  - PyCharm
  - VSCode
  - Termius
  - Adobe XD
  - Postman
  - Docker / Kubernetes
  - Frameworks
    - IOI Isolate
    - Facebook React
    - Google Firebase
    - Google Pub/Sub
    - Palantir Blueprint
- Hardware
  - 3 servers for judging (Digital Ocean)
  - 3 laptops for developing

### 5.2. Work Breakdown and Work Assignment

To achieve a more efficient architecture, we use direct connections between front-end and some Firebase Services (read only operations on database, upload to static storage operations etc.). Proper implementation of this architecture requires a great effort on front-end compared to a classic monolithic web app. It is the reason why we split works in front-end.

Work	Responsible	Deadline
System Architecture	Everybody	Done
Firebase Integrations	Everybody	15 December
Judge Service	Kadir Emre Oto	30 October
Back-end Service	Muhammed Burak Buğrul	13 November
Back-end and Judge Integration	Muhammed Burak Buğrul - Kadir Emre Oto	20 November
Front-end Skeleton	Süheyl Emre Karabela	Done
Front-end Side Online Code Editor	Süheyl Emre Karabela	30 October
Front-end Student Pages	Kadir Emre Oto	30 November

Front-end Lecturer and T.A. Pages	Muhammed Burak Buğrul	30 November
Front-end and Back-end Integration	Süheyl Emre Karabela - Muhammed Burak Buğrul	5 December
Static Storage Management	Süheyl Emre Karabela - Kadir Emre Oto	13 November
Tests	Everybody	30 December



## 6. Goals and Evaluation Criteria

As mentioned in the requirements section, the system should be highly available, secure and robust. These are the numerical goals:

- The system should support more than 10.000 concurrent users.
- The system should not leak personal information of its users.
- User codes should be executed in an isolated environment in order not to crash the system.
- In case of a system failure, the system should be recovered in 30 seconds.
- The system web apps should be accessible via using at least LTS versions of Google Chrome, Mozilla Firefox, Microsoft Edge and Safari web browsers
- All submissions should be judged in 1 minute.
- A T.A. can add students to a course.
- A T.A. can add homework to a course.
- A T.A. can modify test data of a homework.
- A T.A. can evaluate homework submissions.

## 7. References

1. "About Us" 2019 Online. <https://www.hackerrank.com/about-us>
2. "Contests" 2019 Online. <https://www.hackerrank.com/contests>
3. "Frequently Asked Questions" 2010 Online. <http://codeforces.com/help>
4. "CS Academy" Online. <https://csacademy.com/about/>
5. "calico 1.1.2" Online. <https://pypi.org/project/calico/>
6. "Gradescope" 2019 Online. <https://www.gradescope.com>