Software Requirements Specification
for
ACM Online Judge System

Prepared by:
Muhammet ORAZOV
Mihail PONAMARYOV
Kaan SORAL

Computer Engineering
Middle East Technical University
# Table of Contents

1 Introduction .................................................................................................................. 3  
  1.1 Problem Definition ................................................................................................. 3  
  1.2 Purpose .................................................................................................................. 4  
  1.3 Scope ....................................................................................................................... 4  
  1.4 Literature Survey and User ..................................................................................... 5  
    1.4.1 Market Search .................................................................................................... 5  
      1.4.1.1 Topcoder ...................................................................................................... 6  
      1.4.1.2 Sphere Online Judge (SPOJ) ....................................................................... 6  
    1.4.2 Literature Survey ............................................................................................... 7  
      1.4.2.1 Security ........................................................................................................ 7  
      1.4.2.2 Rating .......................................................................................................... 8  
      1.4.2.3 Educational Aspect ...................................................................................... 8  
    1.4.3 User Characteristics ............................................................................................ 8  
      1.4.3.1 Users/Contestants ....................................................................................... 8  
      1.4.3.2 Judges ........................................................................................................... 9  
      1.4.3.3 Administrators ............................................................................................ 9  
  1.5 Definitions and Abbreviations ............................................................................... 9  
  1.6 References ............................................................................................................. 9  
  1.7 Overview ................................................................................................................ 9  
2 Overall Description ..................................................................................................... 10  
  2.1 Product Perspective ............................................................................................... 11  
  2.2 Product Functions .................................................................................................. 11  
3 Specific Requirements .................................................................................................. 16  
  3.1 Functional Requirements ....................................................................................... 16  
    3.1.1 System functions available for all users ......................................................... 16  
    3.1.2 System functions available for a contestant ............................................... 20  
    3.1.3 System functions available for Judge ......................................................... 25  
    3.1.4 System functions available for administrators ......................................... 27  
  3.2 Non-Functional Requirements ............................................................................ 27  
    3.2.1 Design Constraints ......................................................................................... 27  
      3.2.2.1 Hardware Constraints ............................................................................... 28  
      3.2.2.2 Software Constraints ............................................................................... 28  
      3.2.2.2 Software Attributes .................................................................................... 30  
4 Data Model and Description ....................................................................................... 32  
  4.1 Data Description .................................................................................................... 32  
    4.1.1 Contestant Data Entity .................................................................................... 32  
    4.1.2 Judge Data Entity ............................................................................................. 32  
    4.1.3 Administrator Data Entity ............................................................................... 33  
    4.1.4 Problem Data Entity ........................................................................................ 33  
    4.1.5 Contest Data Entity ......................................................................................... 33  
    4.1.6 Articles Data Entity ........................................................................................ 34  
  4.2 Data Objects .......................................................................................................... 34  
    4.2.1 User data objects .............................................................................................. 35  
    4.2.2 Problem, Contest and Article data objects .................................................. 36  
  4.3 Data Relationships ................................................................................................ 36  
5 Planning ....................................................................................................................... 37
1 Introduction

This Software Requirements Specification (SRS) analysis report is prepared for the senior design course, CEng491 2010-2011 fall semester of Computer Engineering Department (CEng) of Middle East Technical University (METU). It mainly aims to present and describe the project ACM Online Judge in detail which is to be designed and implemented by Algorithmists team.

1.1 Problem Definition

Online Judge is a online system used to test correctness and efficiency of programs written in various programming languages. They are used on programming competitions held online as well as on site. Most importantly they are used to practice for a programming contests. These systems are used by lots of people, particularly students, to learn issue related to programming languages, algorithms, data-structures and to improve programming skills through training and competing in several programming contests. Furthermore, they have been used by recruiters of the well known companies for job applications (i.e, www.codility.com).

There are a lot of online judge systems. However, none of them have been able to provide an environment that will boost the knowledge sharing, expert help and social interaction.

In this project we are planning to build an environment that will help users to interact, practice and improve their programming skills via social interaction, expert guidance and by organizing live online competitions.

Another motivation that lead to build this project is the lack of these systems in
Turkey. ACM Online Judge will play a crucial role in organizing competition among university students and will provide a social environment that will improve programming skills.

1.2 Purpose

The purpose of this document is to describe the software requirements for the ACM Online Judge System. It is important that these requirements are reached in order to answer all the expectations. This document uses written descriptions as well as modeling diagrams to illustrate the high level structure of the software. Furthermore, this documents aims to give necessary view of requirements for each stakeholders considering their responsibilities.

This document is intended for:

Developers: in order to be sure they are developing the right software that fulfills requirements provided in this document.

Contestants, Users: to get familiar with the idea of the software and suggest other features that would make it even more functional.

Judges: to get familiar with procedures involved in assigning contest problems and grading them and suggest more practical features.

Administrators: in order to know exactly what they have to expect from the system, right inputs and outputs and response in error situations.

Documentation writers: to know the features and in what way they have to explain. What security technologies are required, how the system will response in each user's action.

1.3 Scope

ACM Online Judge aims to build an environment for the people, specifically students, around the world to interact, practice and improve their programming skills. It is designed to organize live online programming contests in fair and secure way. Moreover, it provides an environment for training and practicing algorithms, data-structures and programming languages.
To achieve the main goal of the project, two main parts should be connected:

1. **The interface that users can use.**

   The user interface will be essentially the web interface. There will be several user interface modules where each module is designed for a specific user.

2. **The backend that will validate the submitted solutions for the users or contestants.**

   All the grading of problems will be done at the backend of the system.

The main features of this project are:

1. **Training/Practice:**

   Solving the problems of the site. The coders can send their solutions of problems at any time, each problem has its own discussion forum, statistics, etc.

2. **Managing live programming contests:**

   Judges/Experts, determined by the ratings of users, can organize live programming contests. They can propose new problems, or can choose from the problem set of the system. The system will receive the submissions only for a determined period of time, only within contest duration. The format of the contest and the rating of the problems will be set by the judges.

3. **Tutorials/Articles:**

   Judges can write tutorials related to algorithms, data structures and programming languages.

   Moreover, students can use this environment on programming courses related to learning programming, algorithms and data structures at schools, universities, etc.

### 1.4 Literature Survey and User

#### 1.4.1 Market Search

There are several online judge systems; however, two of them are similar to *ACM Online Judge*. In order to make the reader familiar with the project it is a good idea to
outline some brief information about these projects.

1.4.1.1 Topcoder

TopCoder is a company which administers contests in computer programming. TopCoder hosts online algorithm competitions as well as weekly competitions in design and development.

TopCoder has been hosting algorithm competitions since 2001. Contests consist of four phases:

- **Coding phase (75 minutes):** Coders write programs to solve three short problems using a limited selection of languages (currently C++, Java, C# and Visual Basic). Each problem set consists of an easy, medium, and hard question, the difficulty of which is reflected in each problem's point value.

- **Intermission (5 minutes):** After the coding phase ends, there is a short break before the challenge phase begins. This time can be used to think of challenge cases.

- **Challenge phase (15 minutes):** Coders can challenge the submitted solutions of other participants in their room by constructing test cases in an attempt to generate erroneous output. The challenger receives 50 points for a successful challenge (the challenged coder loses all his points for that problem), and loses 25 points for an unsuccessful challenge. Each successful challenge is added to the set of tests to be run during the system testing phase.

- **System-testing phase:** Each problem that survives through the challenge phase is run on many test cases. If a coder's solution fails the system tests, that coder receives no points for the problem.

At the end of the contest, ratings are updated to incorporate each participating coder's performance.
1.4.1.2 Sphere Online Judge (SPOJ)

SPOJ (Sphere Online Judge) is an online judge system with over 75,000 registered users and over 6,000 problems. The solution to problems can be submitted over 40 languages including C, C++, Python, C#, Go, Haskell, Ocaml, F#. SPOJ has a rapidly growing problem-set/tasks available for practice 24 hours/day, including many original tasks prepared by the community of expert problem setters associated with the project.

SPOJ allows advanced users to organize contests under their own rules and problems.

1.4.2 Literature Survey

Literature Survey is important in order to build ACM Online Judge. There are several academic papers that should be read before building any online judge.

1.4.2.1 Security

Security is the one of the important attributes in all online judge systems so that online competitions are organized in a fair way.


This paper analysis currently available online judge system. It suggests an approach to design an online judge system as a secure IT system using the known methods from the area of computer security. Moreover, it gives a detailed analysis of new, specific threats that arise in the context of online judges and provides methods to handle all the threats mentioned.


This paper presents methods to construct a restricted environment in order to prevent any buggy or malicious submitted code from damaging the grading system,
bypassing resource constraints or stealing information in order to obtain a better score using Linux Security Module.

1.4.2.2 Rating

Rating is crucial in any competitive environment.


In this thesis author shows the disadvantages of the currently used rating system in online programming contests, namely Bayesian reasoning about the skills of the contestants. It provides a novel approach to ratings that is based on Item Response Theory (IRT). Using this testing theory author suggests a new type of rating system that considers the task difficulty.

1.4.2.3 Educational Aspect

Most of the online judge systems are used in undergraduate and graduate courses of universities worldwide such as algorithms, data-structures and programming languages.

- Adrian Kosowski, Michal Malafiejski, Tomasz Noinski: *Application of an Online Judge & Contester System in Academic Tuition*. ICWL 2007: 343-354

The paper describes how SPOJ online judge system has been used for learning programming, which has been successfully applied in the tuition of students at the Gdańsk University of Technology. Moreover, authors present their experiences connected with the use of such systems in academic courses at an undergraduate and graduate level.

1.4.3 User Characteristics

Intended user characteristics are mentioned below.

1.4.3.1 Users/Contestants

Normal users or Contestants are primary users of the *ACM Online Judge*. We
expecting most of the users to be university students. Most of the users will have medium level knowledge on programming. Moreover, most of them, about %25, will have an programming competition experience.

1.4.3.2 Judges

Judges will be essentially contestants. They are users with high level rating. Most of the judges will be researcher on the field of programming, algorithms and data-structures

1.4.3.3 Administrators

Administrators are primarily responsible for maintenance of the system. They contribute minimally to the problems and contests themselves, but spend more time modifying the system configuration and making updates.

1.5 Definitions and Abbreviations

The abbreviations used throughout the document and their definitions are as follows:

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACM</td>
<td>Association for Computing Machinery</td>
</tr>
<tr>
<td>METU</td>
<td>Middle East Technical University</td>
</tr>
<tr>
<td>CENG</td>
<td>Computer Engineering</td>
</tr>
</tbody>
</table>

1.6 References


1.7 Overview

The remainder of this document is two chapters. The next chapter of this document provides a full description of the ACM Online Judge software project. It introduces the interfaces involved and provides the summary of the functions that the software will
perform.

The third chapter describes the ACM Online Judge functions in full detail so that it would be sufficient for the designers to design the software. This chapter provides use-case scenarios in order to describe functional requirements. Non-functional requirements are also described in this chapter. It specifies the required hardware and software; moreover, it describes the necessary attributes so that their success are verified.

This specification is organized as follows

1) **Introduction:** which introduces the specifications for the ACM Online Judge to its readers.

2) **Overall Description:** which provides a brief, high level description of the project.

3) **Specific Requirements:** which specifies the functional and non-functional requirements of ACM Online Judge.

4) **Data Model and Description:** gives detailed analysis of data stored on the databases

5) **Planning:** describes our team structure, process model and gives plan estimation

6) **Conclusion**

## 2 Overall Description

ACM Online Judge system is composed of several modules designed according to user profiles.

### 2.1 Product Perspective

ACM Online Judge consists of a database server which contains data for users and the problems added by judges. Each user’s data are divided into groups and subgroups so that they are organized in the form that serves right the user. Groups
contain entries with usernames, passwords, ratings, problems solved, problems added, contests attended etc. Problems database contains all the information about the problems such as categories it belongs, statistics, etc. All the data model is described in the section 4.

In the diagram below there are main components of the system, subsystem interconnections and external interfaces to help the reader understand the main idea of ACM Online Judge. All of them are analyzed with more detail in this document.

\[Illustration 1: Components of ACM Online Judge\]

### 2.2 Product Functions

**ACM Online Judge** provides the users with the following functions. Detailed use-cases are given in the section 3.1 Functional requirements

- **Register**

  A login screen is required for the system. In order not to get bored with lots of registration forms, an e-mail address of the user and their password will be taken and
a user profile will be created if there isn’t one with that mail address already.

- **Login**
  
  Registered users can login to the system after providing their usernames and passwords.

- **Update profile**
  
  A profile editing page is required to be more social in this environment. A profile picture and a general profile info creation must be included. A profile should only be edited by its owner.

- **View a problem**
  
  Users that are logged into system can view a problem descriptions and statistics.

- **Submit a solution**
  
  Users that are logged into the system can submit a solution of the problem written in one of the programming languages.

- **View results**
  
  Users can view the results of their submitted solution to any problem.

- **Register to a contest**
  
  Users can register to a contest which is organized by a judge.

- **Read an article**
  
  Users can read articles or tutorials related to programming written by the judges of the system.

- **Search for a user**
  
  User can search his/her friends in the user database (name, mail address, education etc.). The main purpose is to make contact with their friends all over the world.

- **Send a friend request**
  
  Users can request to be friends via mail or message. User who receives the request confirms or ignores.

- **Start a conversation with a friend**
Users can start a conversation (instant messaging) with an online friend via instant messaging services provided by the system.

- **Search a expert**

  Users can search for the experts who are able to help them in certain topics. This property is maintained after a search for a certain topic.

- **Send dialog request with a expert**

  After finding an expert who the user thinks that he/she can be helpful for a topic, user sends them a request for a conversation. Expert who receives the request confirms or ignores.

- **Add a problem**

  Users with judge privileges can add problems to currently active contests, either their own problems or problems from the site problem collection.

- **Set a contest**

  Users with judge privileges can set up new contests.

- **Write an article**

  Users with judge privileges can post their articles.

- **View submissions**

  Users with administrator privileges can view codes submitted by any contestant for any problem.

Below use-case diagrams show the summary of the functions mentioned above.
Illustration 2: Functions for Contestant/Normal User
Illustration 3: Functions for Judge
3 Specific Requirements

3.1 Functional Requirements

System functions are given according to their availability to different types of users. They are organized by use cases so that the main functions of the system will be understandable.

3.1.1 System functions available for all users
**System Authorization**

**Use Case 1 : Register**

*Primary Actor :* User  
*Precondition :* Internet Connection Available, *ACM Online Judge* is open on any browser  
*Stakeholders and Interests :*  
Users wants to register to the system.

**Main Scenario :**

1. User presses ‘Register’ button  
2. User fills in fields with personal information. Fields like user name, password, and email address are required to be filled in.  
3. User presses ‘Done’ button.  
4. User is prompted about “Successful Registration”  
5. User is taken to the Login page

**Alternate Scenario :**

2(a). User is already registered  
2(a).1 Prompt user that he/she is registered  
2(a).2 Take back to Login page  
2(b). User is has entered a password shorter than 8 characters  
2(b).1 Ask a user to enter longer password  

**Use Case 2 : Login / Logout**

*Primary Actor :* User  
*Precondition :* Internet Connection Available  
*Stakeholders and Interests :*  
User wants to enter into the system

**Main Scenario :**

1. User opens a browser and enters to the acm.metu.edu.tr  
2. User is prompted for login and password.
3. User enters his/her username and password.
4. System does authentication
5. Main page is displayed. User can see all permitted pages.

Alternate Scenario:
4(a). Authorization fails
   4(a).1 Prompt user that he/she typed wrong password
   4(a).2 Allow him/her to re-enter the password. Give 3 chances

<table>
<thead>
<tr>
<th>Use Case 3 : Update Profile</th>
</tr>
</thead>
</table>
Primary Actor : User
Precondition : User is logged in
Stakeholders and Interests :
Users want to update their profiles

**Main Scenario** :
1. User presses “Update Profile” button
2. Users are required to specify the necessary fields
3. User presses “Save” button
4. User is taken to his/her personal page

Alternate Scenario :
2(a). User did not obey the criteria of required fields
   2(a).1 Prompt user that to reenter the information
   2(a).2 Take back to profile editing page
2(b). User presses “Cancel” button
   2(b).1 User taken to his/her personal page

<table>
<thead>
<tr>
<th>Use Case 4 : Search a friend</th>
</tr>
</thead>
</table>
Primary Actor : User
Precondition : User is logged in
Stakeholders and Interests:
User wants to find his/her friend in the system

Main Scenario:

1. User presses “Search” button
2. User chooses to search an contestant
3. User enters parameters like username, problems added or contests set to conduct search on
4. User presses “Ok” button
5. Links to personal pages of all users that satisfy search criteria is shown
6. User selects one user from search result and goes to his/her personal page

Alternate Scenario:

6(a) User presses “Cancel” button and goes to his/her personal page

Use Case 5: Send a friend request

Primary Actor: User

Precondition: User should be logged in, user is on the personal page of an user

Stakeholders and Interests:
User wants to be a friend with a user

Main Scenario:

1. User presses “Send Friend Request” button
2. User presses “Back” button and goes to his/her personal page
### Use Case 6: Start conversation with a friend

**Primary Actor**: User  
**Precondition**: User should be logged in  
**Stakeholders and Interests**:  
User wants to talk with his/her friends  

**Main Scenario**:
1. User presses “Friends” button  
2. System show all online friends of the user  
3. User chooses a friend and presses “Talk” button  
4. If friend confirms a messaging windows will be opened on both sites  

**Alternate Scenario**:
4(a) If friend did not confirm the request  
4(a).1 User is prompted with a message

### 3.1.2 System functions available for a contestant

### Use Case 7: View Problem

**Primary Actor**: User  
**Precondition**: User logged in  
**Stakeholders and Interests**:  
User or contestants wants to read or solve a problem  

**Main Scenario #1: Practice Mode**  
1. User goes to the page of practice problems  
2. User chooses a problem from the list  
3. User presses “View” button and (s)he is taken to problem descriptions
4. User presses “Back” button and (s)he is taken to her/his personal page

*Alternate Scenario:*

**Main Scenario #2: Contest Mode**

1. User chooses a contest from the list of available and (s)he is taken to page containing all the problems added to the contest
2. User selects a problem from the list and presses “View” button
3. User is taken to the page with statement of the problem
4. User presses “Back” button and (s)he is taken to her/his personal page

*Alternate Scenario:*

1(a): User is not registered to a contest
   1(a). 1: Prompt user that (s)he is not registered to see problems
   1(a). 2: Take user to contest registration page

---

**Use Case 8: Submit Solution**

*Primary Actor:* User

*Precondition:* User logged in

*Stakeholders and Interests:*

User or contestants wants to submit a solution to a problem

**Main Scenario #1: Practice Mode**

1. User goes to the page of practice problems
2. User chooses a problem from the list
3. User presses “Upload Solution” button, chooses a file, specifies the programming language and presses upload
4. User presses “Back” button and (s)he is taken to her/his personal page
Alternate Scenario :

3(a): User has not specified any programming language
   3(a). 1: Prompt user that (s)he has not chosen any programming language
   3(a). 2: Take user to solution submission page

Main Scenario #2: Contest Mode

1. User chooses a contest from the list of available and (s)he is taken to page containing all the problems added to the contest
2. User chooses a problem from the list
3. User presses “Upload Solution” button, chooses a file, specifies the programming language and presses upload
4. User is taken to the page with statement of the problem
5. User presses “Back” button and (s)he is taken to her/his personal page

Alternate Scenario:

3(a): User has not specified any programming language
   3(a). 1: Prompt user that (s)he has not chosen any programming language
   3(a). 2: Take user to solution submission page

Use Case 9: Register to a contest

Primary Actor : User
Precondition : User is logged in
Stakeholders and Interests :
Users wants to register to the available contests
Main Scenario :
   1. User selects a contest from an available list
   2. User presses a “Register” button
3. Success message appears and user is taken to page with contest problems

Alternate Scenario:

2(a). User is already registered
   2(a).1 Prompt user that he/she is registered
   2(a).2 Take to the page with problems

---

**Use Case 10: Read an Article**

**Primary Actor**: User  
**Precondition**: User is logged in  
**Stakeholders and Interests**:  
Users wants to read an article written by judges  

**Main Scenario**:

1. User presses “Articles” button and (s)he is taken to articles page currently available  
2. User selects an article from the list and presses “Read” button and (s)he is taken to the page with full-text of the article  
3. User presses “Back” button and (s)he is taken to the articles page

---

**Use Case 11: Search for an Expert**

**Primary Actor**: User  
**Precondition**: User is logged in  
**Stakeholders and Interests**:  
User wants to search an expert on a specific topic  

**Main Scenario**:

1. User presses “Search” button  
2. User chooses to search an expert
3. User enters parameters like username, problems added or contests set to conduct search on

4. User presses “Ok” button

5. Links to personal pages of all expert (judges) that satisfy search criteria is shown

6. User selects one expert from search result and goes to expert's personal page

Alternate Scenario :

6(a) User presses “Cancel” button and goes to his/her personal page

<table>
<thead>
<tr>
<th>Use Case 12: Send Dialog Request to an Expert</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Primary Actor</strong> : User</td>
</tr>
<tr>
<td><strong>Precondition</strong> : User should be logged in, user is on the personal page of an expert</td>
</tr>
<tr>
<td><strong>Stakeholders and Interests</strong> :</td>
</tr>
<tr>
<td>User wants to take an expert advice</td>
</tr>
<tr>
<td><strong>Main Scenario</strong> :</td>
</tr>
<tr>
<td>1. User presses “Send Conversation Request” button</td>
</tr>
<tr>
<td>2. Expert confirms the request</td>
</tr>
<tr>
<td>3. Messaging window appears on both ends</td>
</tr>
<tr>
<td><strong>Alternate Scenario</strong> :</td>
</tr>
<tr>
<td>2(a) If an expert is not online or did not confirm the request</td>
</tr>
<tr>
<td>2(a).1 User is prompted with a message</td>
</tr>
</tbody>
</table>
3.1.3 System functions available for Judge

**Use Case 13**: Set a contest

*Primary Actor*: Judge

*Precondition*: User is logged in

*Stakeholders and Interests*:
Judge wants to organize a contest

*Main Scenario*:

1. Judge presses “Set a Contest” button
2. System asks the user to enter the attributes of the contest
3. Judge specifies the required fields
   1. Contest Name
   2. Date
   3. Duration
   4. Number of problems
   5. Judge adds problems
4. Judge presses “Set” button

*Postcondition*: Judge announces the contest on the main page of the ACM Online Judge system.

**Use Case 14**: Add a problem

*Primary Actor*: Judge

*Precondition*: User is logged in

*Stakeholders and Interests*:
Judge wants to add a problems to a contest

*Main Scenario*:
1. Judge presses “Active Contests” button and goes to list of active contest

2. Judge selects a contest and presses “Add own problem” button

3. Judge specifies the required fields
   1. Restricted programming languages
   2. Execution time
   3. Memory limit
   4. Judge upload the problem description
   5. Judge upload the test case and if necessary checker program

4. Judge presses “Done” button

Alternate Scenario:

2(a) : Judge presses "Add problem from site collection"

   2(a).1: Judge selects a problem from problem list of the system

2(b).1: Judge presses “Add” button

---

**Use Case 15**: Post an article

*Primary Actor*: Judge

*Precondition*: User is logged in

*Stakeholders and Interests*:

Judge wants to add an article or tutorial to article's list of the system

*Main Scenario*:

1. Judge presses “Add Article” button
2. System prompts to enter required fields
3. Judge specifies the required fields
   1. Title
   2. Category
3. Short Description

4. Judge upload the full-text to the system

4. Judge presses “Done” button

3.1.4 System functions available for administrators

**Use Case 16**: View Submissions

*Primary Actor*: Administrator

*Precondition*: User is logged in

*Stakeholders and Interests*:
Administrator wants to see the solution submissions to a problem

*Main Scenario*:

1. Administrator presses “Problems” button
2. System provides all available problems
3. Administrator presses “View Submissions” button
4. System provides all solution submissions to a problem
5. Administrator presses “View” button in order to see submitted solution

3.2 Non-Functional Requirements

In this section design constraints and software attributes are mentioned.

3.2.1 Design Constraints

In this section necessary hardware requirements are listed for the system where all grading and judging are done.
3.2.2.1 Hardware Constraints

**Database Servers**

Since Algorithmium does not have enough knowledge to do a reasonable benchmarking among DBMSs, the team plans to consult in a professional for selecting one of the alternatives: MySQL, PostgreSQL, DB2, MsSQL, and Oracle. But presently MySQL is our choice. Because MySQL is free software and according to researches it has most of the abilities of a DBMS should have. Additionally it is easy to host, more powerful and useful.

**Web Server**

We need a fast and reliable server. This machine will show high performance. For web server we will use a dual-core 2.6 Ghz, 2 Gb DDR-2 ram with at least 800 Mhz frequency. Its capacity is not very important. Standart capacity will be sufficient. This server must have a very high bandwith or maybe more servers. Any IBM based computer will be useful, its trade is not important. Any dedicated web server is seeming as mandatory. Moreover it is not mandatory yet desirable if it has very high-speed connection

**Developers Requirements**

Project developers must have at least 1.6 Ghz Intel CPU, 512 Mb DDR2 Ram, with Unix/Linux Operating system. And 20 Gb hard-drive will be sufficient.

3.2.2.2 Software Constraints

**Programming Languages**

Since we will be developing a client-side web interfaces and backend grading system, programming languages such PHP and C/C++ are inevitable.

While developing this project Google App Engine will be used as well.

**Here is a list of software that we will be utilizing while developing this project:**

- For every supported programming language a compiler is needed; preferably one that can generate statically linked stand-alone executable.
- Apache web server with support for PHP >= 5.0.0 and the mysql extension for PHP. We also recommend the posix extension for extra debugging information.
- MySQL >= 4.1.x database and client software
- PHP >= 5.0.0 command line interface and the mysql extension.
- A POSIX compliant shell in /bin/sh (e.g. bash or ash)
- A statically compiled POSIX shell
- glibc >= 2.1
- A lot of standard (GNU) programs, a probably incomplete list: hostname, date, dirname, basename, touch, chmod, cp, mv, cat, grep, diff, wc, mkdir, mkfifo, mount, sleep, head, tail, pgrep
- sudo to gain root privileges
- Apache htpasswd
- xsltproc from the GNOME XSLT library package.
- A LaTeX installation to regenerate the team PDF-manual with site specific configuration settings included.

The following items are optional, but may be required to use certain functionality.

- **phpMyAdmin**, to be able to access the database in an emergency or for data import/export
- An **NTP daemon** (for keeping the clocks between system and judge hosts in sync)
- **libcurl** (to use the command line submit client with the web interface)
- **libmagic** (for command line submit client to detect binary file submissions)
- **GeSHi** or PEAR Text Highlighter class (to use syntax highlighting in the Show Source section for judge interface)
- **PECL xdiff extension** (to reliably make diffs between submissions, *ACM Online Judge* will try alternative approaches if it's not available, i.e checker programs
written by juries)

- **PHP zip Extension** (to upload problem data via zip bundles)
- **beep** for audible notification of errors, submissions and judging, when using the default alert script.

### 3.2.2.2 Software Attributes

In this section software attributes are specified in order to verify them objectively.

1) **Usability**

   **As far as possible, the system should provide a simple, responsive interface.**

   Although ACM Online Judge System is composed of diverse systems, applications, and processes, the underlying architecture should be transparent to the administrators. The system should be configurable from a single source at a central administrative position, and should provide a central, easy-to-use interface that will allow administrators to configure the user interface and features in a way that reduces page clutter.

   A system will be considered to meet this requirement if it has a single administrative interface rather than individual links for editing each page. Furthermore, this interface must allow administrators to easily change themes and other setting that affect page layout across the entire judge system.

2) **Reliability**

   **The system must be backed up on a configurable schedule**

   Back-up requirements will need to be determined, based on individual components of the system. The system should be backed up with a frequency that ensures system and all data is protected. Since the updates and changes will be done to the database via web-interface, it should be backed up on a nightly basis, with options for weekly, as well as off-site backup when necessary.

   The system must have the ability and capacity to restore back-up data within five hours so that the system is never offline for an inordinate period of time.

3) **Availability**
The system should be available 24 hours a day, 7 days a week

This statement provides a general sense of system availability. It is not intended to demand the system maintain reliability, or to require the system to be highly available. It should not exclude scheduled downtime. It is only intended to convey the expectation that our users should have access to the system during organized contest times which makes this attribute number one priority; moreover the times when users are practicing or training. 99% up-time should be considered sufficient to meet this requirement.

4) Maintainability

The system must be maintainable without substantial modification

Due to limited number of administrators and support staff, it is important that the system be mostly self-sustaining. This will reduce the number of hours spent maintaining the system and simplify maintenance tasks.

5) Performance

The system should support at least 1000 concurrent users.

This statement provides a general sense of reliability when the system is under load. It is important that a substantial number of actors be able to access the system at the same time. The times when the system will be under the most stress are likely during organized live contests; therefore, it must be able to handle at least 1,000 concurrent users.

6) Security

The system must comply with the permission roles

Security is the most important attribute of the system. System should not allow unauthorized accesses. It should not mix the roles of the users. For instance, contestants should not be allowed to reach the solutions of the problems or access to the problem test data. A minimum of %99 time, the system should prohibit the actions that are not permitted for a users.
4 Data Model and Description

In this section all the data models stored in the ACM Online Judge server databases are described.

4.1 Data Description

Data description of each data entity is described here.

4.1.1 Contestant Data Entity

<table>
<thead>
<tr>
<th>Data Item</th>
<th>Type</th>
<th>Description</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>Text</td>
<td>Username provided when registering</td>
<td></td>
</tr>
<tr>
<td>Password</td>
<td>Text</td>
<td>Password of user</td>
<td></td>
</tr>
<tr>
<td>Email Address</td>
<td>Text</td>
<td>User’s e-mail address</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Text</td>
<td>The actual name of the user</td>
<td></td>
</tr>
<tr>
<td>Information</td>
<td>Text</td>
<td>Contestant’s necessary information</td>
<td></td>
</tr>
<tr>
<td>Problems Solved</td>
<td>Pointer</td>
<td>Problem entity</td>
<td>May be more than one</td>
</tr>
<tr>
<td>Contests Attended</td>
<td>Pointer</td>
<td>Contest Entity</td>
<td>May be more than one</td>
</tr>
<tr>
<td>Friends</td>
<td>Pointer</td>
<td>Users Entity</td>
<td>May be more than one</td>
</tr>
</tbody>
</table>

4.1.2 Judge Data Entity

<table>
<thead>
<tr>
<th>Data Item</th>
<th>Type</th>
<th>Description</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>Text</td>
<td>Username provided when registering</td>
<td></td>
</tr>
<tr>
<td>Password</td>
<td>Text</td>
<td>Password of the user</td>
<td></td>
</tr>
<tr>
<td>Email Address</td>
<td>Text</td>
<td>User’s e-mail address</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Text</td>
<td>The actual name of the user</td>
<td></td>
</tr>
<tr>
<td>Information</td>
<td>Text</td>
<td>Judge’s necessary information i.e country, school</td>
<td></td>
</tr>
</tbody>
</table>
4.1.3 Administrator Data Entity

<table>
<thead>
<tr>
<th>Data Item</th>
<th>Type</th>
<th>Description</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>Text</td>
<td>Username provided when registering</td>
<td></td>
</tr>
<tr>
<td>Password</td>
<td>Text</td>
<td>Password of the user</td>
<td></td>
</tr>
<tr>
<td>Email Address</td>
<td>Text</td>
<td>User’s e-mail address</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Text</td>
<td>The actual name of the user</td>
<td></td>
</tr>
</tbody>
</table>

4.1.4 Problem Data Entity

<table>
<thead>
<tr>
<th>Data Item</th>
<th>Type</th>
<th>Description</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Text</td>
<td>Name of the problem</td>
<td></td>
</tr>
<tr>
<td>Content</td>
<td>Text</td>
<td>Description of the problem</td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>Text</td>
<td>Category of the problem</td>
<td>May be several</td>
</tr>
<tr>
<td>Statistics</td>
<td>Text</td>
<td>Detailed statistics of problem, number correct of submission etc</td>
<td></td>
</tr>
<tr>
<td>Author</td>
<td>Pointer</td>
<td>Judge entity</td>
<td>May be more than one</td>
</tr>
<tr>
<td>Published</td>
<td>Boolean</td>
<td>Users can reach the problem</td>
<td></td>
</tr>
</tbody>
</table>

4.1.5 Contest Data Entity

<table>
<thead>
<tr>
<th>Data Item</th>
<th>Type</th>
<th>Description</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Text</td>
<td>Name of the contest</td>
<td></td>
</tr>
</tbody>
</table>
### Problems

<table>
<thead>
<tr>
<th>Problems</th>
<th>Pointer</th>
<th>Problem entity</th>
<th>These problems are published after contest</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Author</th>
<th>Pointer</th>
<th>Judge entity</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Contest Time</th>
<th>Date</th>
<th>Start date and time of contest</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Duration</th>
<th>Integer</th>
<th>Duration of contest in minutes</th>
</tr>
</thead>
</table>

### 4.1.6 Articles Data Entity

<table>
<thead>
<tr>
<th>Data Item</th>
<th>Type</th>
<th>Description</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Text</td>
<td>Name of the article</td>
<td></td>
</tr>
<tr>
<td>Content</td>
<td>Text</td>
<td>Content of the article</td>
<td></td>
</tr>
<tr>
<td>Author</td>
<td>Pointer</td>
<td>Judge entity</td>
<td>Principal author of the article</td>
</tr>
<tr>
<td>Other Authors</td>
<td>Text</td>
<td>Other authors is any; else null</td>
<td>Not a pointer to judge entity.</td>
</tr>
<tr>
<td>Category</td>
<td>Text</td>
<td>Category of the article</td>
<td></td>
</tr>
<tr>
<td>Published</td>
<td>Boolean</td>
<td>Users can read the article</td>
<td></td>
</tr>
</tbody>
</table>

### 4.2 Data Objects

Above data descriptions and their major attributes are described in this section.
4.2.1 User data objects

**Contestant**

- **Attributes**
  - private string userName
  - private string password
  - private string emailAddress
  - private string name

- **Operations**
  - public Contestant( )
  - public string getUserName( )
  - public void setUserName(string val)
  - public string getPassword( )
  - public void setPassword(string val)
  - public string getEmailAddress( )
  - public void setEmailAddress(string val)
  - public string getName( )
  - public void setName(string val)
  - public void viewProblem(Problem prob)
  - public void submitSolution(Problem prob)
  - public void viewResults( )
  - public void searchFriend(Contestant name)
  - public bool sendFriendRequest( )
  - public bool sendDialogRequest(Contestant name)
  - public void receiveFriendRequest( )
  - public void sendDialogRequest(Judge name)

**Judge**

- **Attributes**
  - private string userName
  - private string password
  - private string emailAddress
  - private string name

- **Operations**
  - public Judge( )
  - public string getUserName( )
  - public void setUserName(string val)
  - public string getPassword( )
  - public void setPassword(string val)
  - public string getEmailAddress( )
  - public void setEmailAddress(string val)
  - public string getName( )
  - public void setName(string val)
  - public void setProblems(Problem prob)
  - public void viewContestants(Contestant contest)
  - public void writeArticle(Article article)
  - public void receiveExportRequest( )

**Administrator**

- **Attributes**
  - private string userName
  - private string password
  - private string emailAddress
  - private string name

- **Operations**
  - public Administrator( )
  - public string getUserName( )
  - public void setUserName(string val)
  - public string getPassword( )
  - public void setPassword(string val)
  - public void setEmailAddress( )
  - public void setName(string val)
  - public void setProblems(Problem prob)
  - public void viewContestants(Contestant contest)
  - public void writeArticle(Article article)
  - public void viewSubmissions( )

*Illustration 5: User Data Objects*
### 4.2.2 Problem, Contest and Article data objects

<table>
<thead>
<tr>
<th>Problem Attributes</th>
<th>Contest Attributes</th>
<th>Article Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>private string name</td>
<td>private string name</td>
<td>private string name</td>
</tr>
<tr>
<td>private string content</td>
<td>private Judge: author</td>
<td>private string content</td>
</tr>
<tr>
<td>private string category</td>
<td>private string date</td>
<td>private string category</td>
</tr>
<tr>
<td>private string statistics</td>
<td>private int duration</td>
<td>private bool published</td>
</tr>
<tr>
<td>private bool published</td>
<td></td>
<td>private Judge: author</td>
</tr>
<tr>
<td>private Judge: author</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem Operations</th>
<th>Contest Operations</th>
<th>Article Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>public Contest( )</td>
<td>public string getName( )</td>
<td>public Article( )</td>
</tr>
<tr>
<td>public string getName( )</td>
<td>public void setName( string val )</td>
<td>public string getFileName( )</td>
</tr>
<tr>
<td>public void setName( string val )</td>
<td>public Judge: getAuthor( )</td>
<td>public void setArticle( string val )</td>
</tr>
<tr>
<td>public void setColor( )</td>
<td>public void setAuthor( Judge: val )</td>
<td>public string getCategory( )</td>
</tr>
<tr>
<td>public string getCategory( string val )</td>
<td>public void setStatistics( string val )</td>
<td>public void setDuration( )</td>
</tr>
<tr>
<td>public void setStatistics( string val )</td>
<td>public bool getPublished( )</td>
<td>public void setPublished( bool val )</td>
</tr>
<tr>
<td>public void setPublished( bool val )</td>
<td>public Judge: getAuthor( )</td>
<td>public void setAuthor( Judge: val )</td>
</tr>
<tr>
<td>public void setAuthor( Judge val )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Illustration 6: Problem, Contest and Article Data Objects

### 4.3 Data Relationships

Relations among data objects are described in this section using ERD-diagrams.
5 Planning

This section describes the Algorithmium team structure, member roles and briefly describe process model we are going to use. Then we will give plan estimation.

5.1 Team Structure

We decided our team structure to be Democratic Decentralized (DD). Our team is relatively small, and there is no team leader. Communications within the team are horizontal. Decisions about projects are taken by all members of the team so that each member contributes to project equally. Also, in democratic model each member is encouraged to contribute to the project and would come up with innovative ideas.
5.1.1 Member Roles

The summarizer, documentation person and initiator is Kaan Soral. He is also responsible for the initialization of and updates to the project website. He will be the initiator of the project.

The contact person, project manager, documentation person is Muhammet Orazov. He will be in contact with the instructor, teaching assistant and other company executives. He also makes documentation in the team but submitted material such as reports will be written by the help of all group members.

The recorder and documentation person is Mihail Ponamaryov. He will mainly take care of remarkable notes in the meetings. He will also be responsible for the schedule of the team meetings as well as specializing in information hiding principle of the group.

5.2 Process Model

As we will do the project in the content of Ceng491 course, requirements and design parts of the project will be improved sequentially and they will be completed by the end of the first term. Therefore, we cannot use any incremental or evolutionary model. In addition, requirements are well defined and reasonably stable so there will be limited number of development efforts in the project. We will try to make a realistic and safe requirements analysis so that we will have a robust design, which will prevent us from returning back to design phase. Taking all of these into consideration, waterfall process model is the most appropriate model for our project. However, there will be some modifications in this model such that coding, testing, debugging and integration will be performed in one phase under the name 'construction' as seen in figure below.
5.3 Estimation

Below we have provided our team's Grannt Chart.

<table>
<thead>
<tr>
<th>ACM Online Judge</th>
<th>October</th>
<th>November</th>
<th>December</th>
<th>January</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Task</strong></td>
<td>Week-1</td>
<td>Week-2</td>
<td>Week-3</td>
<td>Week-4</td>
</tr>
<tr>
<td>1. Introduction</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>to the Project</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Analysis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Market Research</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Literature Survey</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modeling Diagrams</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scheduling</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deliverables #2: $RS$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Design</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task Distribution</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Web Interface</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Submission</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Design</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Designing System</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Creating Software Design</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deliverables #3: $R$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. Implementation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prototype</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implementation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Preparing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Presentations</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deliverables #4: Detailed Design Report</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. Demo</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Preparing for Demo</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deliverables #5: Prototype Demo</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- More than one team member
- Kaani Soral
- Muhammet Orlov
- Mihail Fonemaryov
6 Conclusion

This report aimed to show our project’s requirement details in terms of several aspects. First, a brief summary of ACM Online Judge is introduced. Then, a market and literature research is carried out and results are established. And at the main part, the requirement details of the project are described. As a last work, project’s schedule is presented. This report tried to focus on the aspects which thought to be important. So, there is no part that reflects irrelevant or useless information. This report was very useful for clarifying the project’s scope. Also, it’ll be beneficial for further planning.

From the beginning to the end, the whole process will be heavy and challenging for us, but with the contribution of our group members and teaching assistant, we can come over all of the difficulties. We believe that this product will take its place in the market and users will see what they expected.