
**S**oftware **L**inked **I**nteractive **C**ompetitive **E**nvironment

**Detailed Design**

**Powerpoint Presentation**

---

Requested by:

**Dr. Darren Lim**
Associate Professor
Siena College
Loudonville, NY

Prepared by:

P=NP solutions

Renee Solheim – Team Leader
Anthony Parente – Systems Administrator
Matthew Ferritto – Webmaster
Zachary Fitzsiimmons – Lead Developer

March $1^{st}$, 2012

# Contents

# Chapter 1

# SLICE Detailed Design

## 1.1 Product Overview

Competitive programming contests are held at all different levels and scales, with a few teams to up to many participants and from single network sessions to across the world through the Internet. Currently Dr. Darren Lim holds a programming contest at Siena College for local area high school students. In the Spring of 2013, Siena College will be holding the Consortium for Computer Science in Small Colleges Northeast (CCSCNE), which includes a programming contest. The current software is not user friendly and consists of several different parts. The goal of our software, SLICE, is to allow fast, both manual and automatic judging during a competitive programming contest and to integrate problem submission, judging, and scoreboard of a contest into a single application.

## 1.2 User Case Narratives

### 1.2.1 Participant User Case Narrative

Participants will each be able to log into SLICE with a provided username and password generated by the Administrative User and provided in a hardcopy sealed envelope. After a Participant logs in, the Participant will be able to view the contest scoreboard, submit problem-solutions or submit clarification-requests. For a problem-solution submission, a Participant will select one of the contest's problems as the associated problem and one of a list of programming languages for the language. For clarification-requests a Participant can make a general request or associate it with a problem. Participants receive a response from a Judge concerning the submission, stating one of the following: Correct Solution, Compiler Error, Running Time Error, Wrong Answer or Time-Limit Exceeded. Participants will also be able to receive both individual responses to clarification-requests from any Judge as well as publically broadcasted messages from any Judge or the Administrative User.

### 1.2.2 Team Advisor User Case Narrative

Team Advisors will be able to log into SLICE to view the problems and scoreboard. Team Advisors accounts will be created by the Administrative User in the same way that Participant Users are created. A Team Advisor's username and password to access SLICE and is provided to each Team Advisor in a hardcopy sealed envelope. The username and password for the Team Advisor account will be generated by the Administrative User. In addition to viewing the problems and scoreboard, Team Advisors will receive all publically broadcasted contest clarifications.

### 1.2.3 Judges User Case Narrative

Judges will each have the ability to log onto SLICE with a username and password generated by the Administrative User and provided to each Judge in a hardcopy sealed envelope. Once logged in, a Judge will be able to view Participant submitted solutions that the system will compile. If a submission compiles, a Judge will then run the submitted programs using test input provided in SLICE that corresponds to the associated question. A Judge then submits a response to the Participant stating one of the following: Correct Solution, Compiler Error, Running Time Error, Wrong Answer or Time-Limit Exceeded. If the submission was a correct solution to the corresponding problem, a Judge with extra permissions in SLICE to update the scoreboard, if there are errors present. A Judge will be able to either respond (in general or concerning a specific problem) to a specific participant or all Participants and Team Advisors by using a messaging prompt.

### 1.2.4 Administrative User Case Narrative

The Administrative User will be able to log into SLICE with a unique username and password. Once logged in, the Administrative User will be able to create and manage the Judge, Participant, and Team Advisor accounts. The permissions for one Judge to have Scoreboard Judge ability is also managed by the Administrative User. The Administrative user can view the scoreboard, but only the Scoreboard Judge can edit the scoreboard. The usernames and passwords for each individual account is put into a hardcopy sealed envelope and distributed to the associated Judges, Participant or Team advisors. The Administrative User will be able to send broadcast messages to all users, or any subset i.e. all Judges, the Scoring Judge, all Participants or all Team Advisors. The Administrative User will manage the set of problems, the test input, and correct output for each contest problem. The Administrative User will also be in charge of the length of the test contest, contest start and stop times and the allowed programming languages.

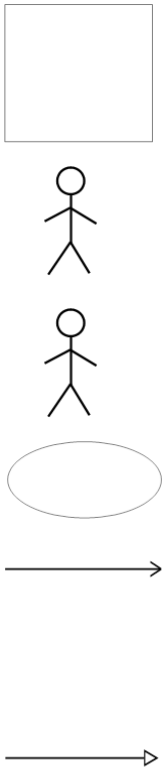
## 1.3 UML Diagrams

### 1.3.1 UML – Use Case Diagram

UML (Unified Modeling Language) Use Case Diagrams present a graphical overview of the functionality for a system. It uses Actors (human and non-human) to represent the users, and their goals, represented by Uses, as well as any dependencies or inheritance between them, represented by specific lines or arrows. The diagram shows what system functions are the performed by each actor.

### 1.3.2 Diagram Legend

We adhere to the UML standard for our UML Use Case Diagrams, but to better explain our diagram we desribe each symbol below:



**System Boundary:** Where the users, human and nonhuman, located on the outside of the boundary, interact with the uses, located on the inside of the boundary, within the system

**Left Side Actor:** Human users which interact with the uses within the system

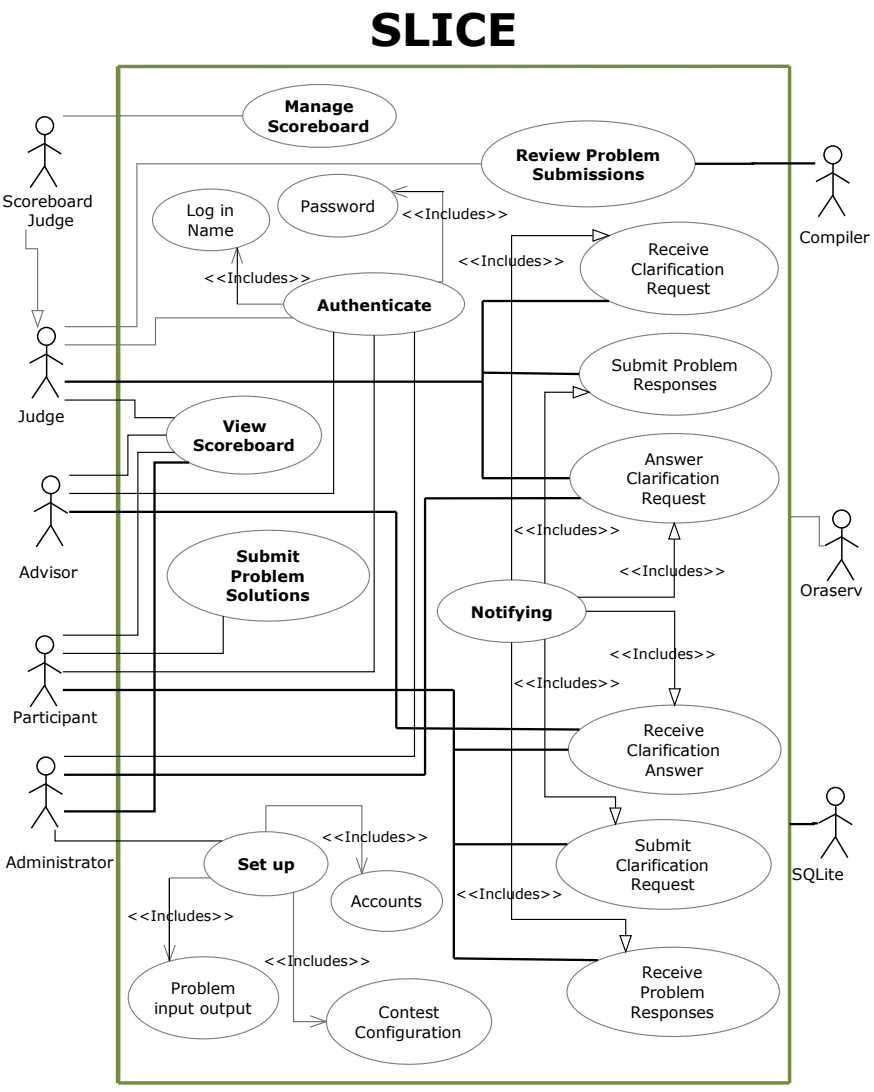
**Right Side Actor:** Non-Human users which interact with the uses within the system

**Uses:** The Activities that interact with the actors outside of the system

**Includes Arrow:** Specific items that are included within the connected

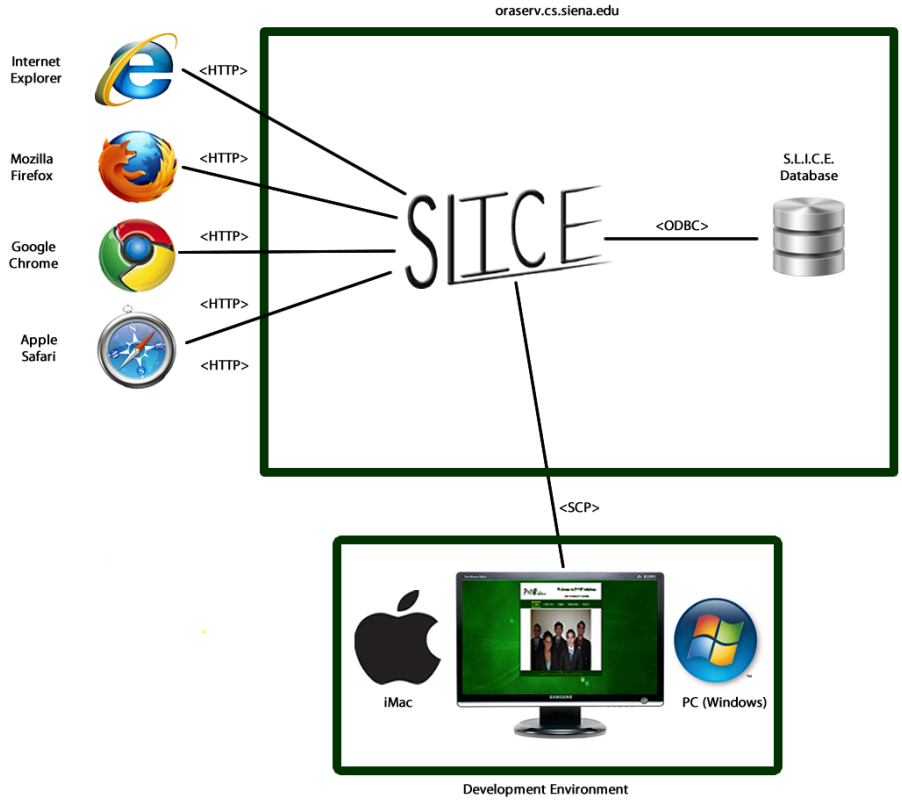
**Inherits Arrow:** Shows subuses of the given uses or actors which are not necessarily accessed when the parent is accessed

### 1.3.3 UML Use Case Diagram for SLICE



### 1.3.4 UML – Deployment Diagram

Deployment Diagrams are a Unified Modeling Language (UML) diagram used to show how different development and execution environment interact in a system. The following diagram shows the physical layout of SLICE.



### 1.3.5 UML – Activity Diagram

Activity Diagrams are a type of flowchart that represents the workflow of the components in our system, SLICE. It shows the overall flow of control and sequences of actions.

#### Legend

We adhere to the UML standard for our UML Activity Diagrams, but to better explain our diagrams we describe each symbol below:

**Start Node:** The starting point for the described activity

**End Node:** The end point for the described activity

**Activity Node:** An activity at a certain step in the described activity

**Decision:** A decision point for the described activity

**Object:** An object used in the activity

**Entity Connector:** The connection used between nodes, decisions, objects and joins

**Join:** Joins several connections together

## Diagrams

Figure 1.1: Activity Diagram for Editing the Scoreboard

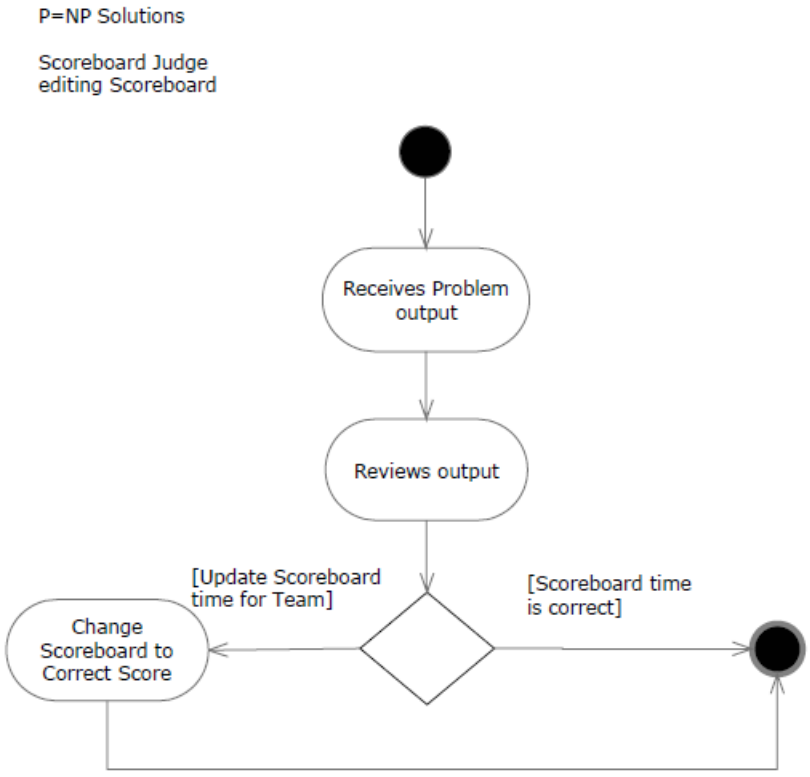
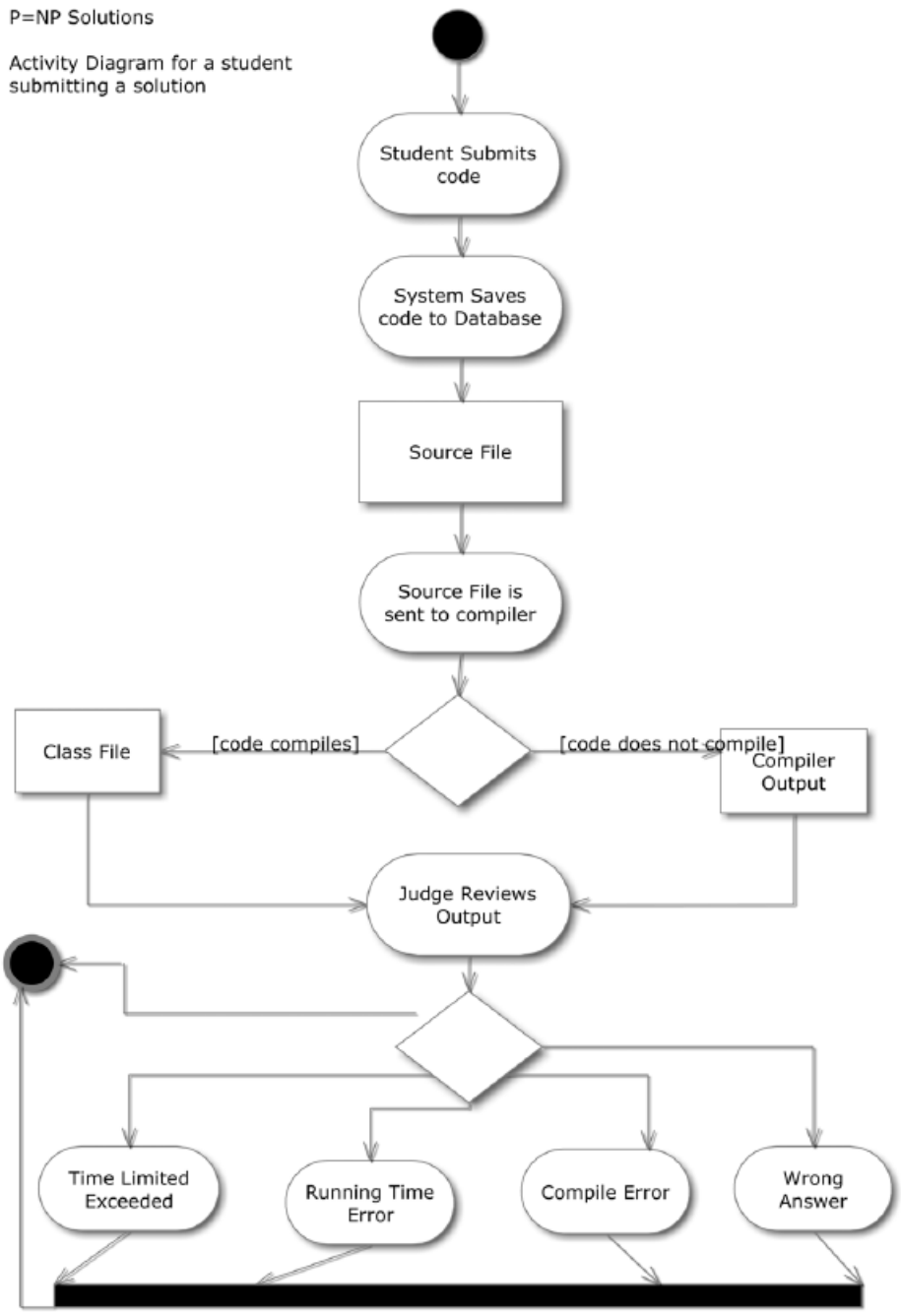


Figure 1.2: Activity Diagram for Problem Submissions



## 1.4 SLICE Website Map

The Website Map illustrates how each user navigates the different pages and views of the website in SLICE.

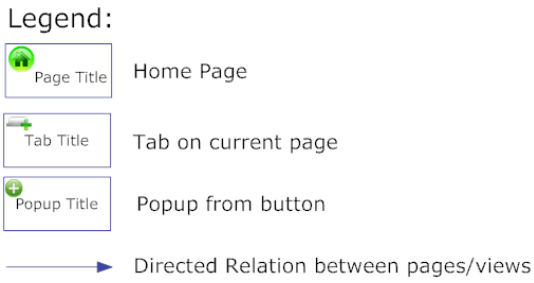
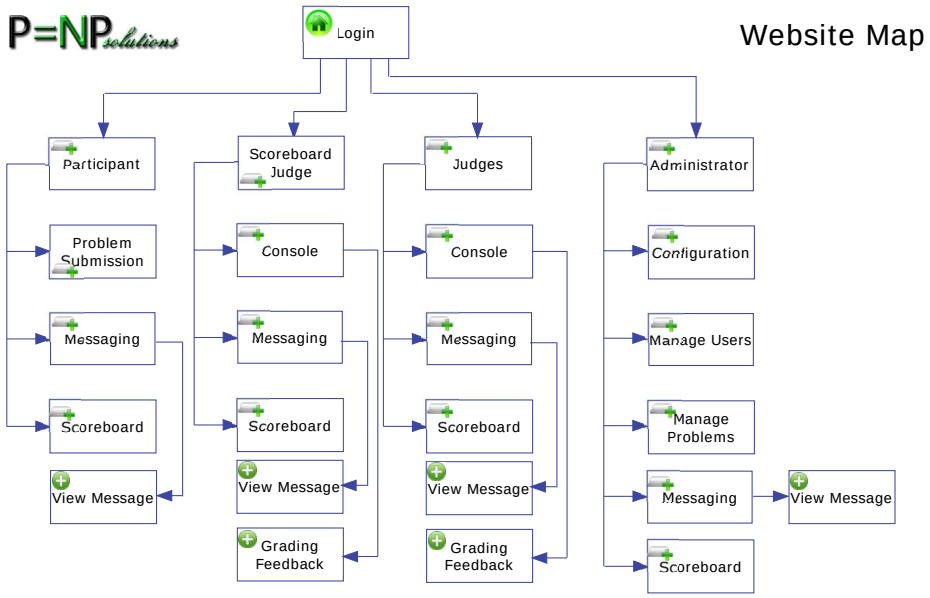



Figure 1.3: SLICE Website Map

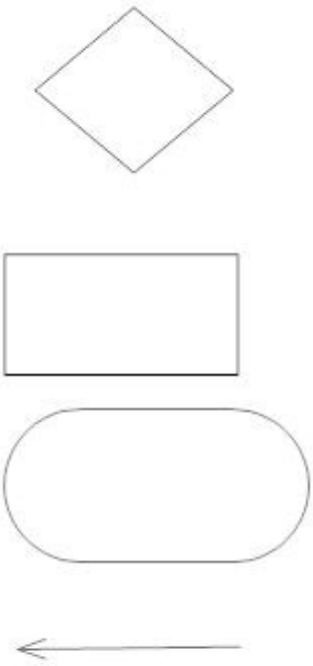


## 1.5 SLICE ER Diagram

An ER diagram or Entity-Relationship Diagram illustrates relations between entities in a database.

### Legend

We adhere to the conventions of standard ER Diagrams, but to better explain our diagrams we describe each symbol below:



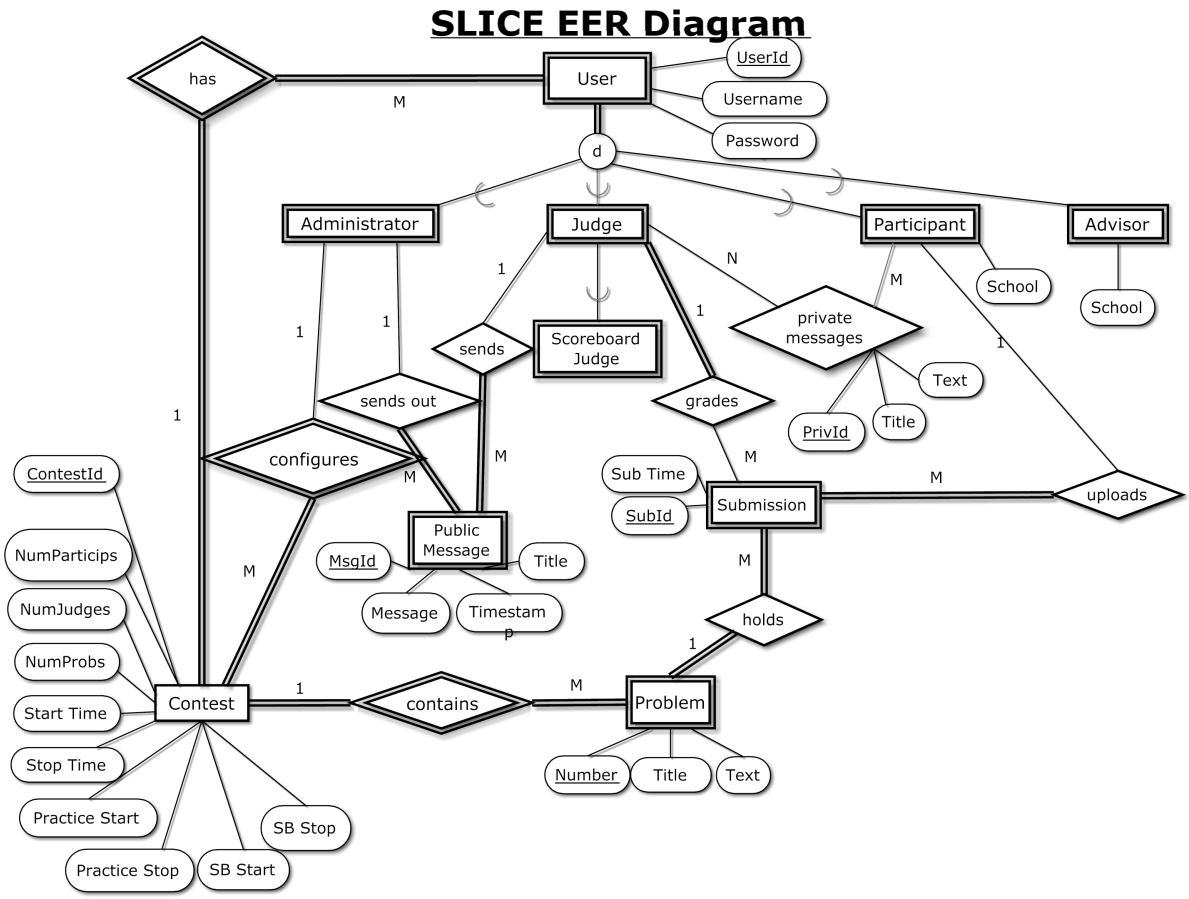
**Relation:** A relation between two entities

**Entity:** A data entity

**Attribute:** An attribute associated with an entity or relation

**Connection:** The connection used between entities, attributes and relations

Figure 1.4: SLICE ER Diagram



## 1.6 SLICE Database Schema

The database schema for SLICE is described by class definitions in the models.py file. We have included this file below.

Figure 1.5: SLICE models.py Database File – Part 1

```
from django.db import models
from django.contrib.auth.models import User

class Contest(models.Model):
    title = models.CharField("Contest Title", max_length=50)
    num_probs = models.CharField("Number of Problems", max_length=30)
    start_time = models.DateTimeField("Contest Start Time")
    end_time = models.DateTimeField("Contest End Time")
    practice_start = models.DateTimeField("Practice Start Time")
    practice_end = models.DateTimeField("Practice End Time")
    sb_start = models.DateTimeField("Scoreboard Start Time")
    sb_end = models.DateTimeField("Scoreboard End Time")

class Administrator(models.Model):
    user = models.ForeignKey(User, primary_key=True)
    contest= models.ForeignKey(Contest, verbose_name="ContestID")
    username = models.CharField("Username", max_length=30)
    password = models.CharField("Password", max_length=30)

class Judge(models.Model):
    user = models.ForeignKey(User, primary_key=True)
    contest_id = models.ForeignKey(Contest, verbose_name="ContestID")
    name = models.CharField("Judge Name", max_length=30)
    username = models.CharField("Username", max_length=30)
    password = models.CharField("Password", max_length=30)

class ScoreboardJudge(models.Model):
    user = models.ForeignKey(User, primary_key=True)
    judge_id = models.ForeignKey(Judge, verbose_name="JudgeID")
    contest_id = models.ForeignKey(Contest, verbose_name="ContestID")
    username = models.CharField("Username", max_length=30)
    password = models.CharField("Password", max_length=30)

class Participant(models.Model):
    user = models.ForeignKey(User, primary_key=True)
    contest_id = models.ForeignKey(Contest, verbose_name="ContestID")
    name = models.CharField("Team Name", max_length=30)
    school = models.CharField("School Name", max_length=30)
    username = models.CharField("Username", max_length=30)
    password = models.CharField("Password", max_length=30)
    pri_msg = models.ManyToManyField(Judge, verbose_name="Private Message", through="PrivateMessage")

class PrivateMessage(models.Model):
    participant = models.ForeignKey(Participant)
    judge = models.ForeignKey(Judge)
    msg_title = models.CharField("Message Title", max_length=30)
    msg_text = models.CharField("Message Text", max_length=500)

class Advisor(models.Model):
    user = models.ForeignKey(User, primary_key=True)
    contest_id = models.ForeignKey(Contest, verbose_name="ContestID")
    name = models.CharField("Advisor Name", max_length=30)
    school = models.CharField("School Name", max_length=30)
    username = models.CharField("Username", max_length=30)
    password = models.CharField("Password", max_length=30)
```

Figure 1.6: SLICE models.py Database File – Part 2

```
class Problem(models.Model):
    contest_id = models.ForeignKey(Contest, verbose_name="ContestID")
    title = models.CharField("Problem Title", max_length=30)
    text = models.CharField("Problem Text", max_length=500)

class Submission(models.Model):
    problem_id = models.ForeignKey(Problem, verbose_name="Problem")
    judge_id = models.ForeignKey(Judge, verbose_name="Judge")
    participant_id = models.ForeignKey(Participant, verbose_name="Participant")
    sub_time = models.DateTimeField("Submission Time")

class PublicMessage(models.Model):
    admin_id = models.ForeignKey(Administrator, verbose_name="Administrator")
    judge_id = models.ForeignKey(Judge, verbose_name="Judge")
    msg_title = models.CharField("Message Title", max_length=30)
    msg_text = models.CharField("Message Text", max_length=500)
    timestamp = models.DateTimeField("Message Timestamp")
```

## 1.7 Functional Requirements Inventory

For a general functional requirement, SLICE will be able to run on all popular browsers including, Internet Explorer, Mozilla Firefox, Google Chrome and Safari.

Below is a list of functional requirements for each human user that interacts with the system. One major functional requirement is that SLICE is scalable to up to 30 Participant users.

### 1.7.1 Participant User

- Will be able to log onto SLICE with a provided username and password while the contest is active
  - An incorrect username and/or password or invalid time period will result in an appropriate error message
- Will be able to log off of SLICE while the contest is active
- Will be automatically logged off of SLICE if the contest becomes inactive
- Will be able to view the contest scoreboard while active
- Will be able to submit problem submission sourcecode
  - Will be able to associate a contest problem with their problem submission
- Will be able to participate in messaging capabilities with the Judges
  - Will be able to send clarification-requests
    - Will be able to associate a problem or send general clarification-requests
  - Will receive results from problem submissions
- Will be able to receive publically broadcasted messages

### 1.7.2 Team Advisor User

- Will be able to log onto SLICE with a provided username and password
  - An incorrect username and/or password will result in an appropiate error message
- Will be able to view the contest scoreboard while active
- Will be able to receive all publically broadcasted messages

### 1.7.3 Judge User

- Will be able to log onto SLICE with a provided username and password
  - An incorrect username and/or password will result in an appropiate error message
- Will be able to log off of SLICE at any point
- Will be able to view the contest scoreboard at any point
- Will be able to review Participant problem submissions
  - Willl be able to compile problem submission sourcecode on a central server
  - Will receive compiile-time feedback
  - Will be able to execute problem submissions with provided test input
  - Will be able to compare the output to provided test output
- Will be able to participate in messaging capabilities with the Participants
  - Will be able to answer both general and contest problem specific clarification-requests
  - Will be able to give results to the Participant concerning problem submissions
- Will be able to send publically broadcasted messages
- A Judge will extra permissions called the Scoreboard Judge will be able to update the contest scoreboard
- The Scoreboard Judge will be able to recieve data consisting of correct submissions with corresponding, problem number, timestamp and Participant

### 1.7.4 Administrative User

- Will be able to log onto SLICE with a provided username and password
  - An incorrect username and/or password will result in an appropriate error message
- Will be able to log off of SLICE at any point
- Will be able to participate in messaging capabilities will all users or any subset of users
- Will be able to complete the initial contest set-up
  - Will be able to create all other user accounts (Participants, Team Advisors, Judges)
  - Will be able to grant one Judge User, Scoreboard Judge permissions
  - Will be able to list contest problems and associate test input and output
  - Will be able to set contest start and stop times
  - Will be able to set the practice time for the contest
  - Will be able to choose the list of allowed programming languages
  - Will be able to set the time that the contest scoreboard will be active during the contest
  - Will be able to view the scoreboard at any time.

## 1.8 Non-Functional Requirements

The non-functional requirement inventory is a list of non-functional system requirements. This list is composed of requirements that specify how the system should be. As more information is gathered about the project, this list may be subject to change:

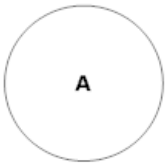
- The system will be easily maintained.
- The system will be stable.
- The system will be viewable on multiple browsers.
- The system will run efficiently.
- The system will be user friendly.

## 1.9 Data Flow Diagrams

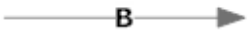
The Data Flow Diagram will show how data flows and is transformed in SLICE. The following diagrams are used as a tool to better illustrate where data is being creating, modified and going in SLICE. We will first describe the symbols used in our diagrams and how they are representing the data.
The Context Diagram is then shown, which will give a general overview of each of our data entities that interacts with the system. The Level Diagram shows more details about each process and the entities that it interacts with. Directed arrows illustrate the flow of data in this diagram. We also have a Level 1 Diagram that describes one of the major processes of the system in a greater deatil than the Level 0 Diagram and is labeled to a smaller granularity of detail.

### 1.9.1 Diagram Legend

Data Flow diagrams are a means of representing a system at any level with a graphic network of symbols. Showing processes, data flows, data stores, and external entities. A key and definition for each symbol has been provided below:

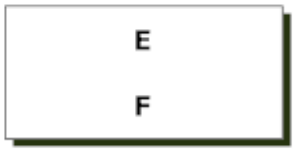



**Process:** The process symbol represents an activity that transforms or manipulates data. The name of the process is within the circle and represented by the letter A.




**Data Flow:** The data flow symbol represents the movement of data. Data flow are labeled with the description of the data that is being moved through it. The is represented by the letter B.

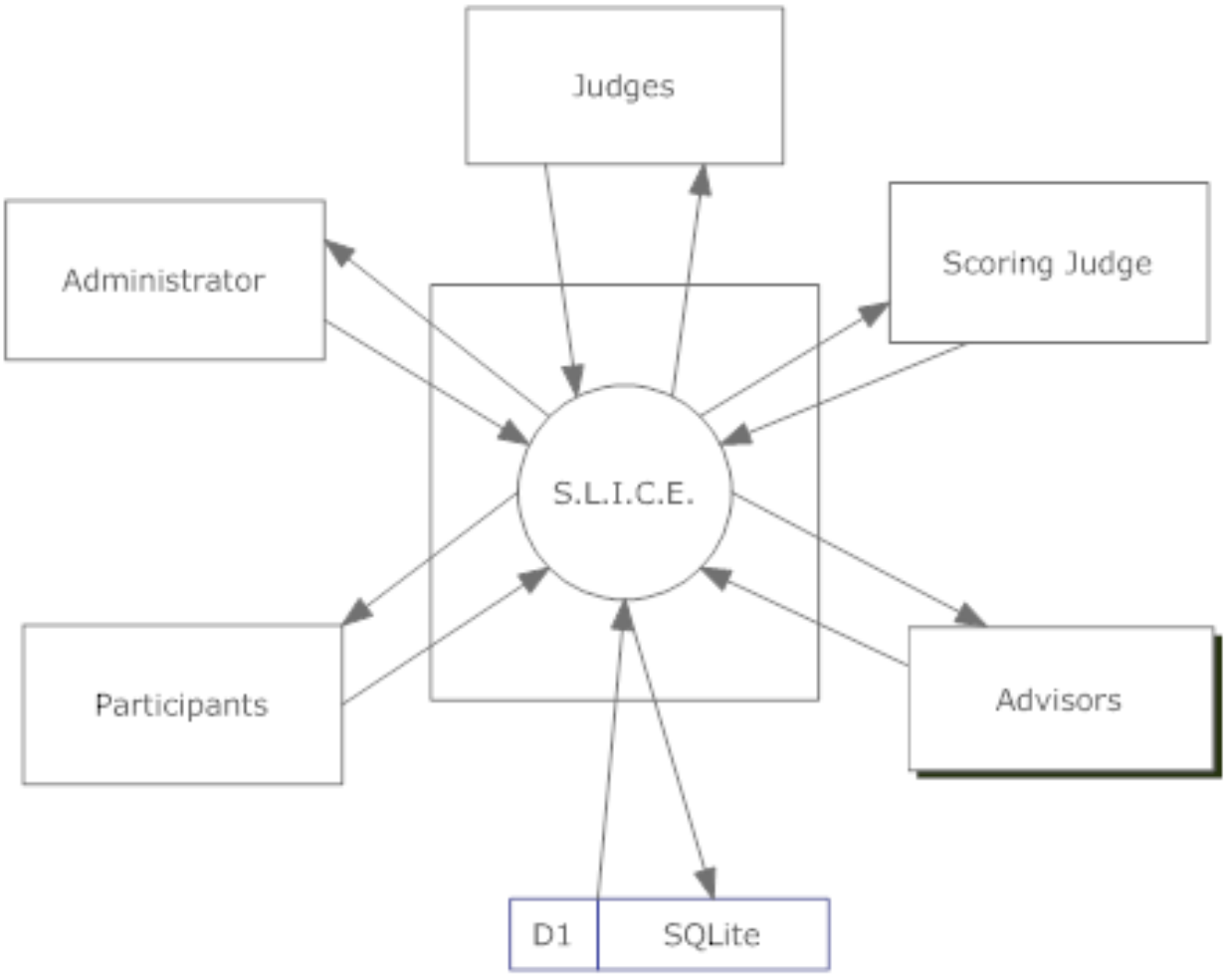

**Data Stores:** The data store symbol represents some location where data is temporarily held. The letter C represents the data store type and the letter D represents the data store name.




**External Entities:** External entities are sources/sinks which contribute data or information to the system or which receive data/information from it. The letter E represents the unique identifier for the entity and the letter F represents the name of the entity.

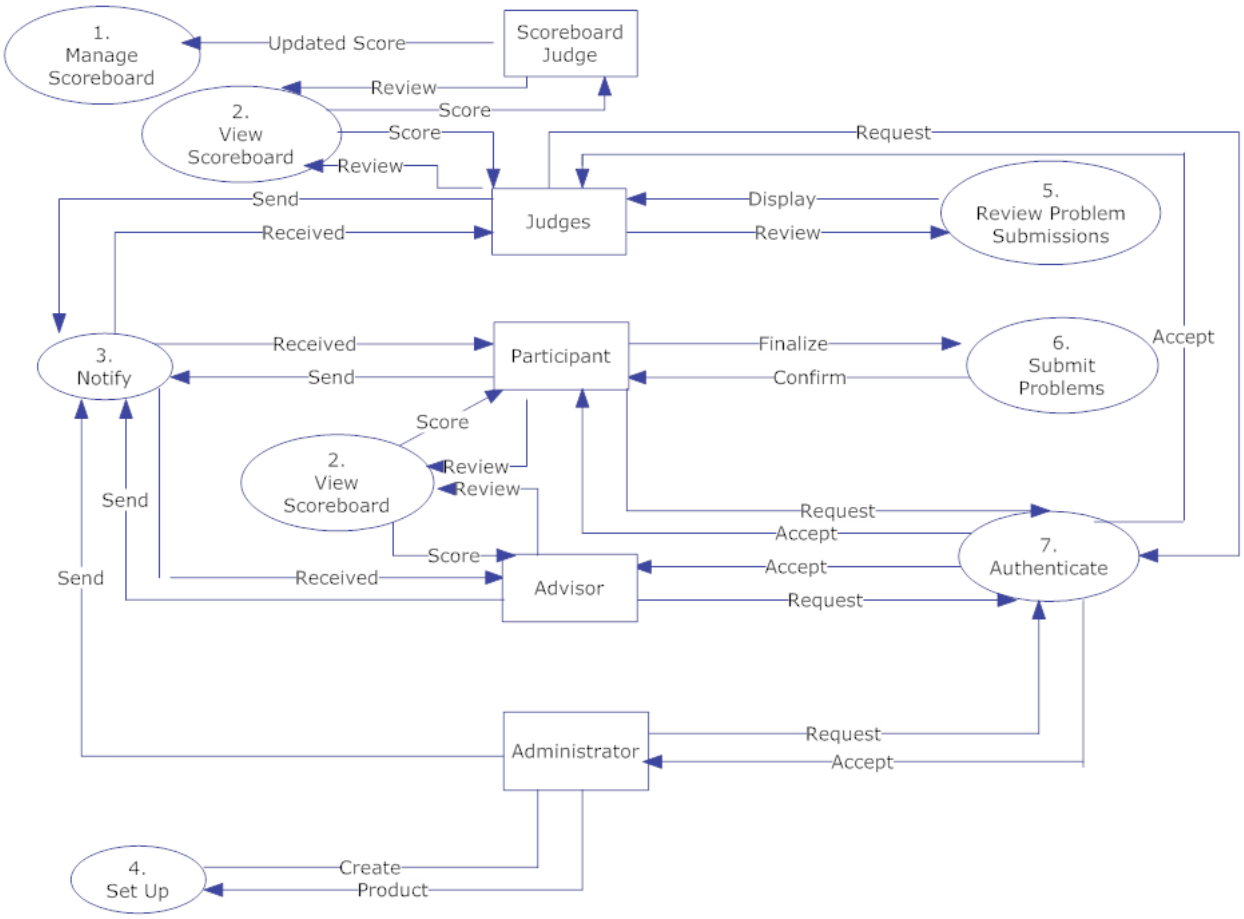
### 1.9.2 Context Diagram

Figure 1.7: Context Diagram for major data entities



### 1.9.3 Level 0

Figure 1.8: Level 0 Diagram for all major processes



### 1.9.4 Level 1 Diagrams

Figure 1.9: Level 1 Diagram of Authenticate

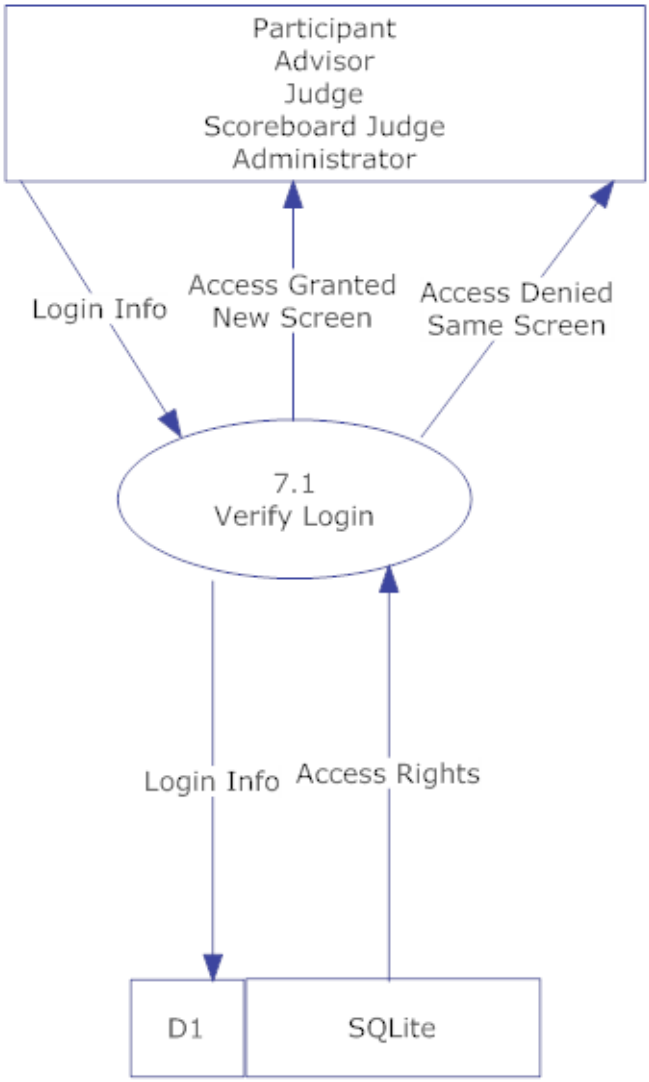


Figure 1.10: Level 1 Diagram of Manage Scoreboard

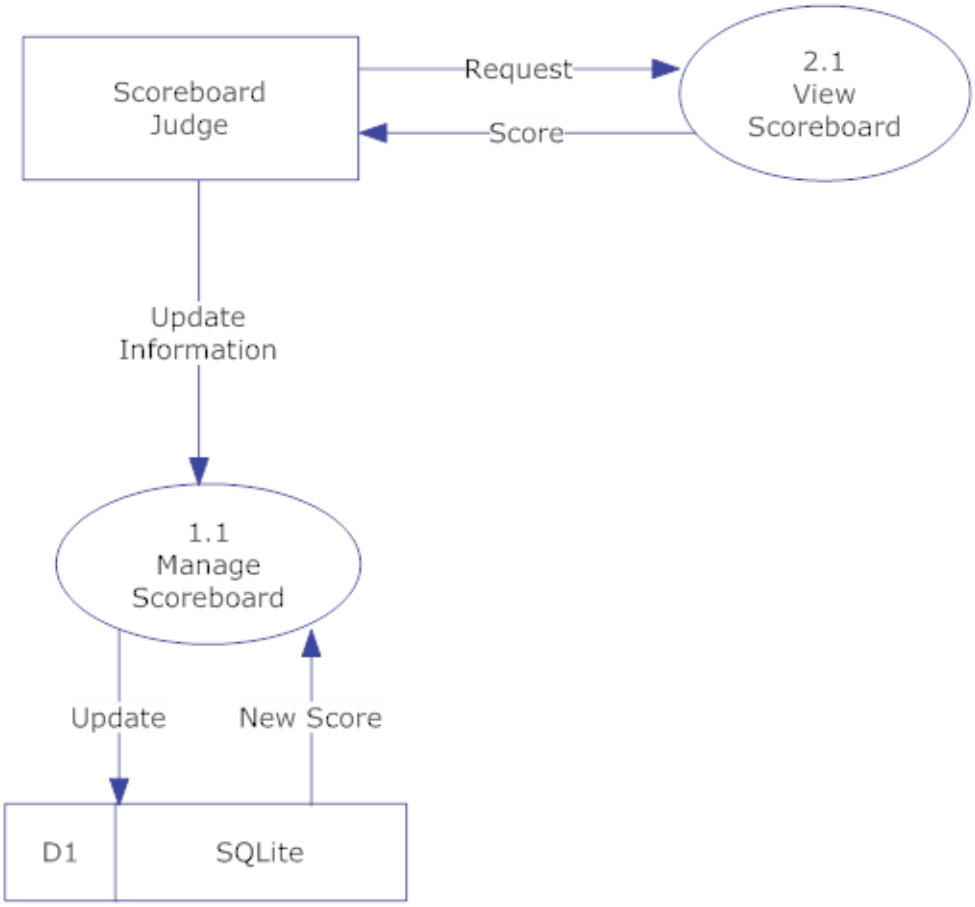


Figure 1.11: Level 1 Diagram of View Scoreboard

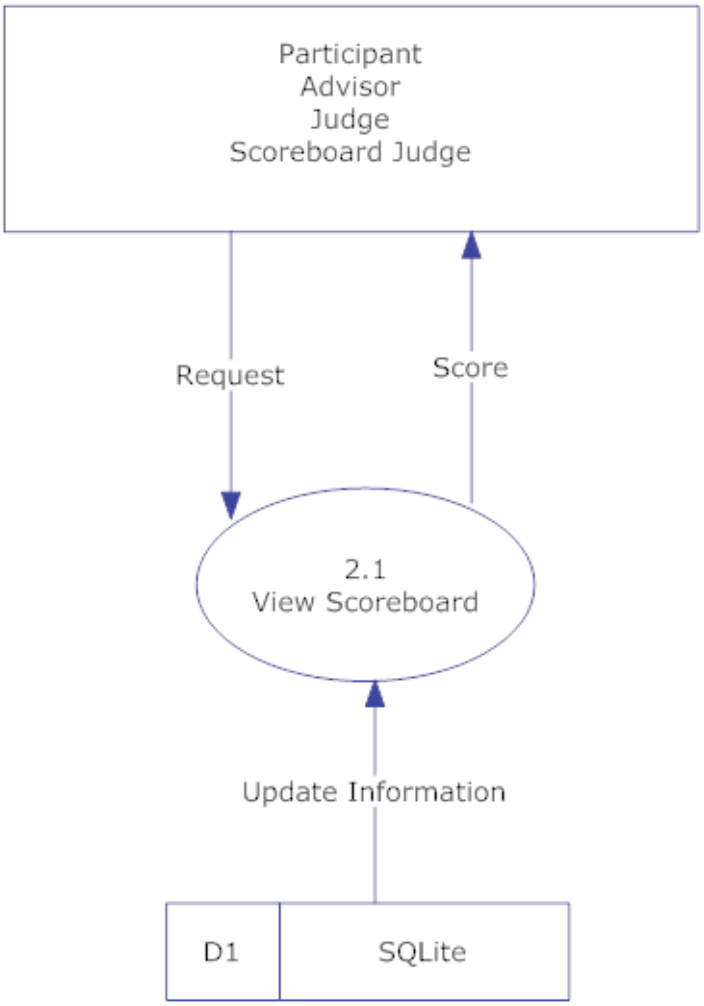


Figure 1.12: Level 1 Diagram of Notify

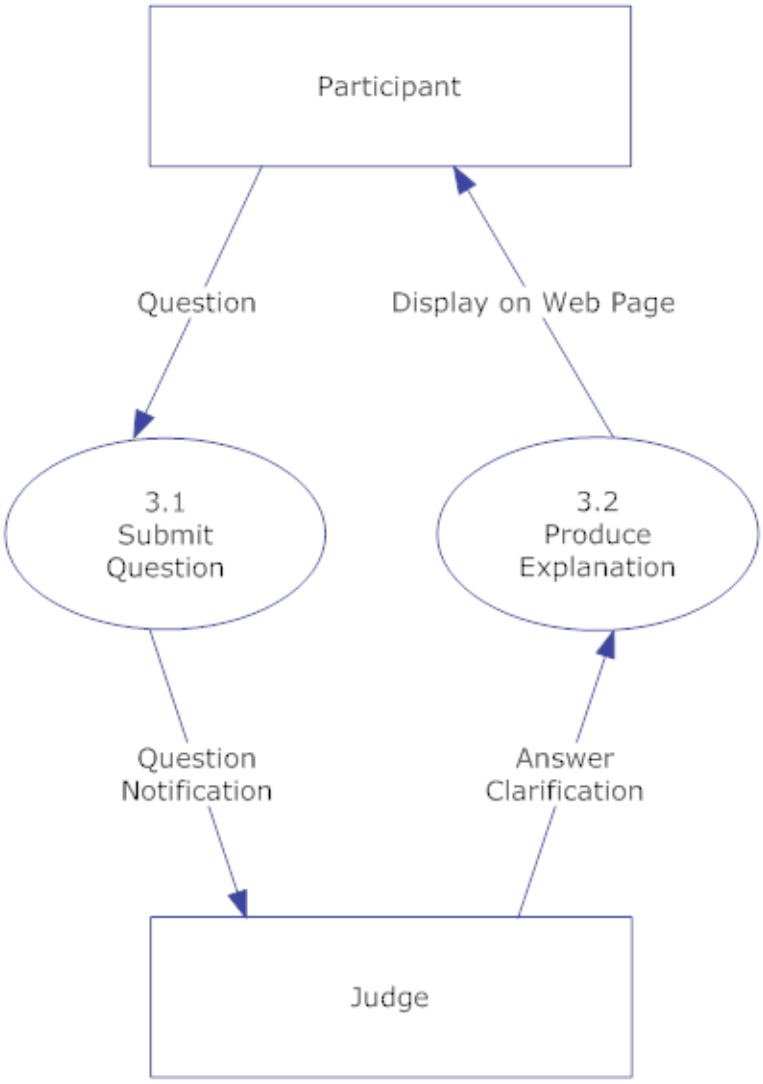


Figure 1.13: Level 1 Diagram of Set Up

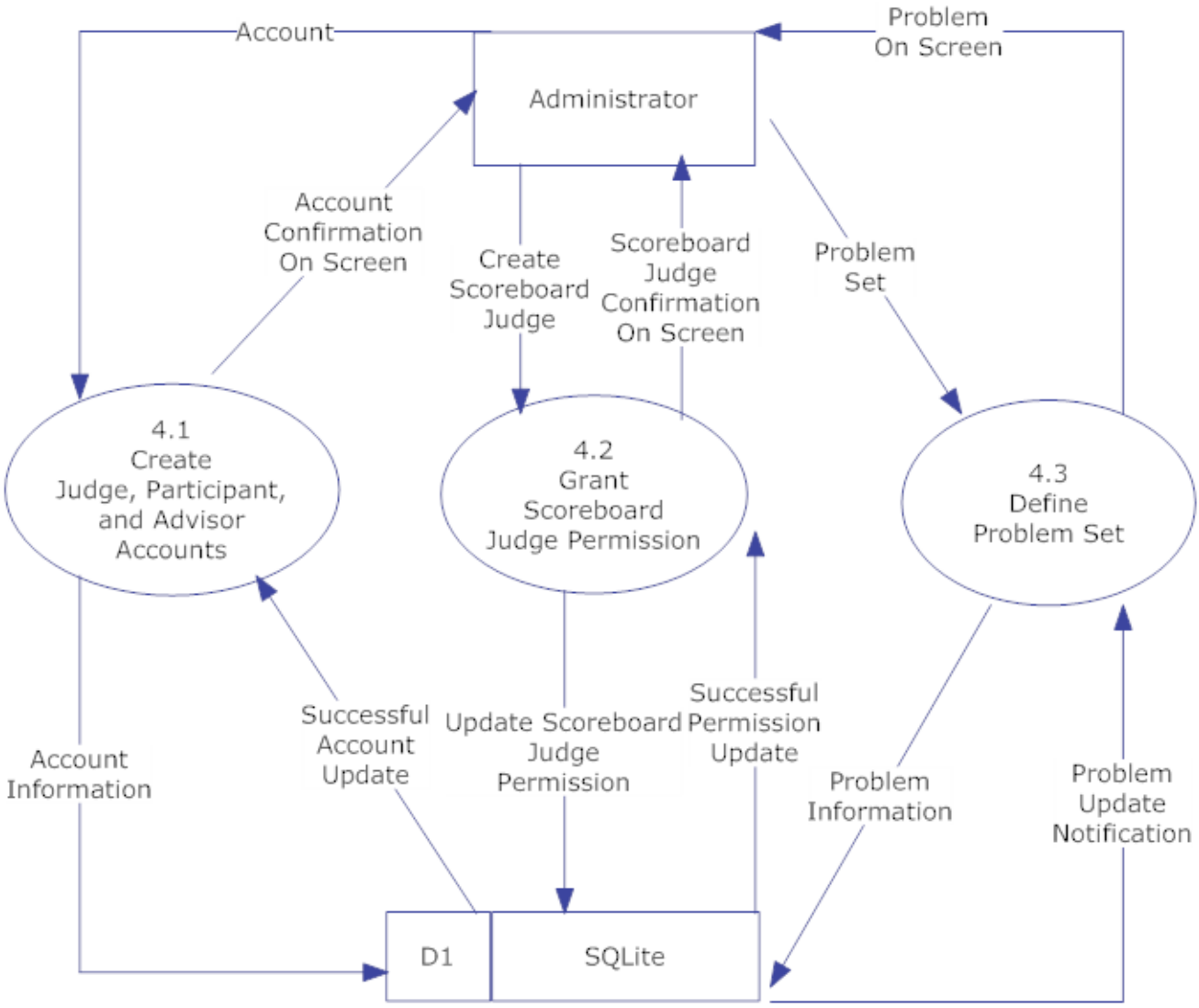


Figure 1.14: Level 1 Diagram of Review Problems

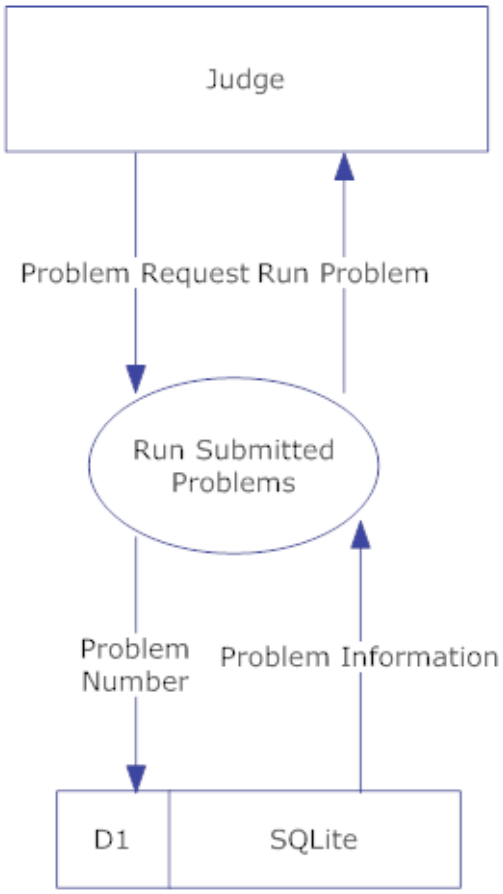
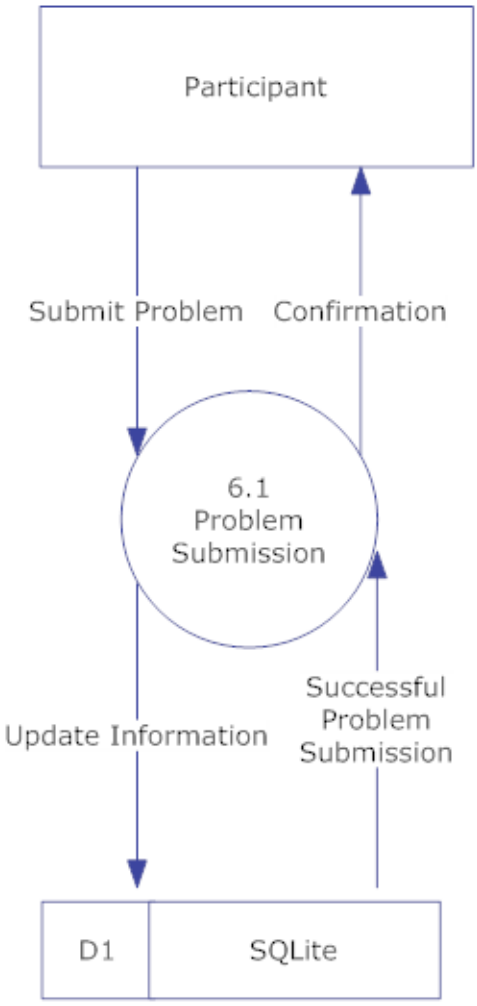


Figure 1.15: Level 1 Diagram of Submit Problems



### 1.9.5 Level 2 Diagrams

Figure 1.16: Level 2 Diagram of Manage Scoreboard

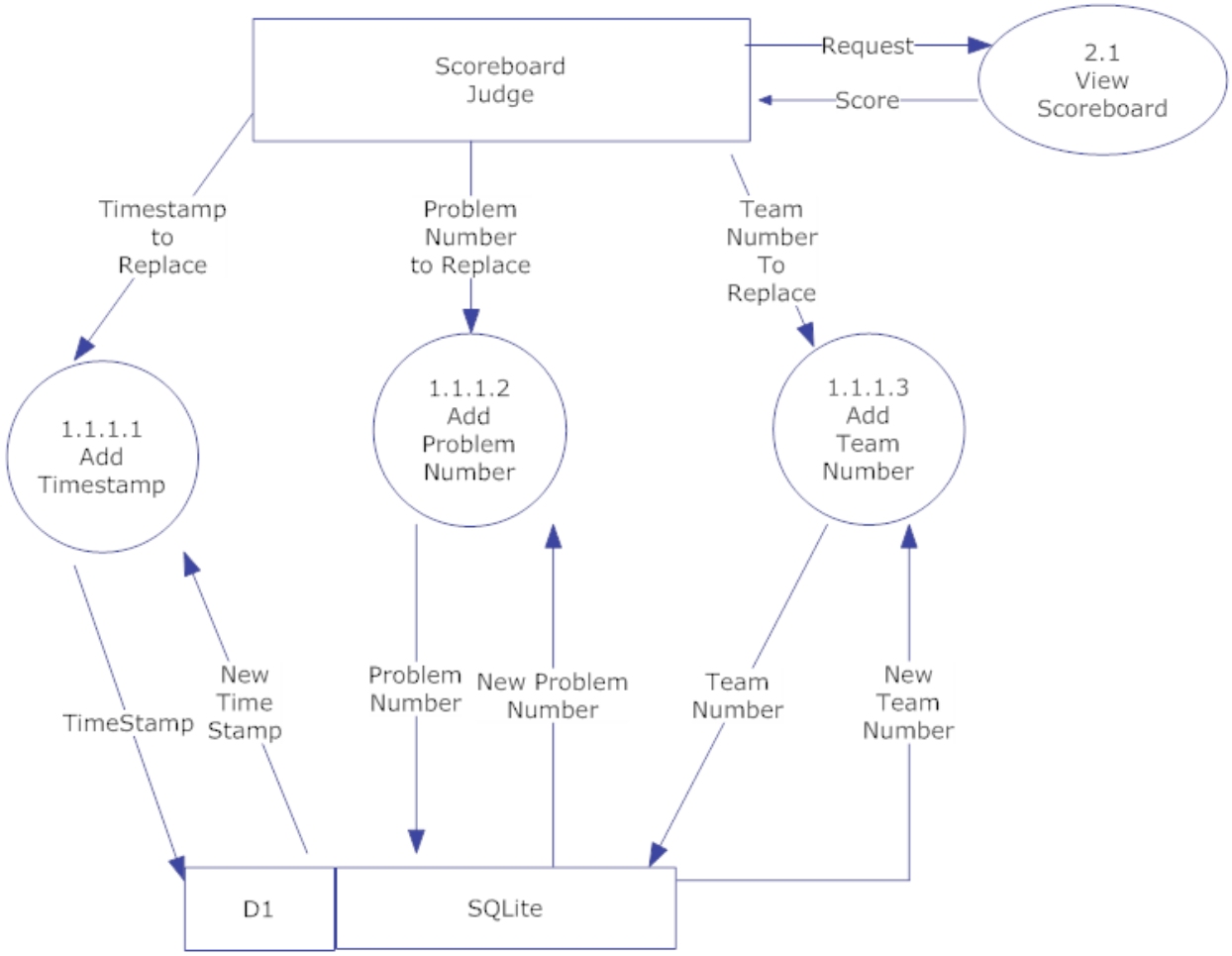


Figure 1.17: Level 2 Diagram of Notify

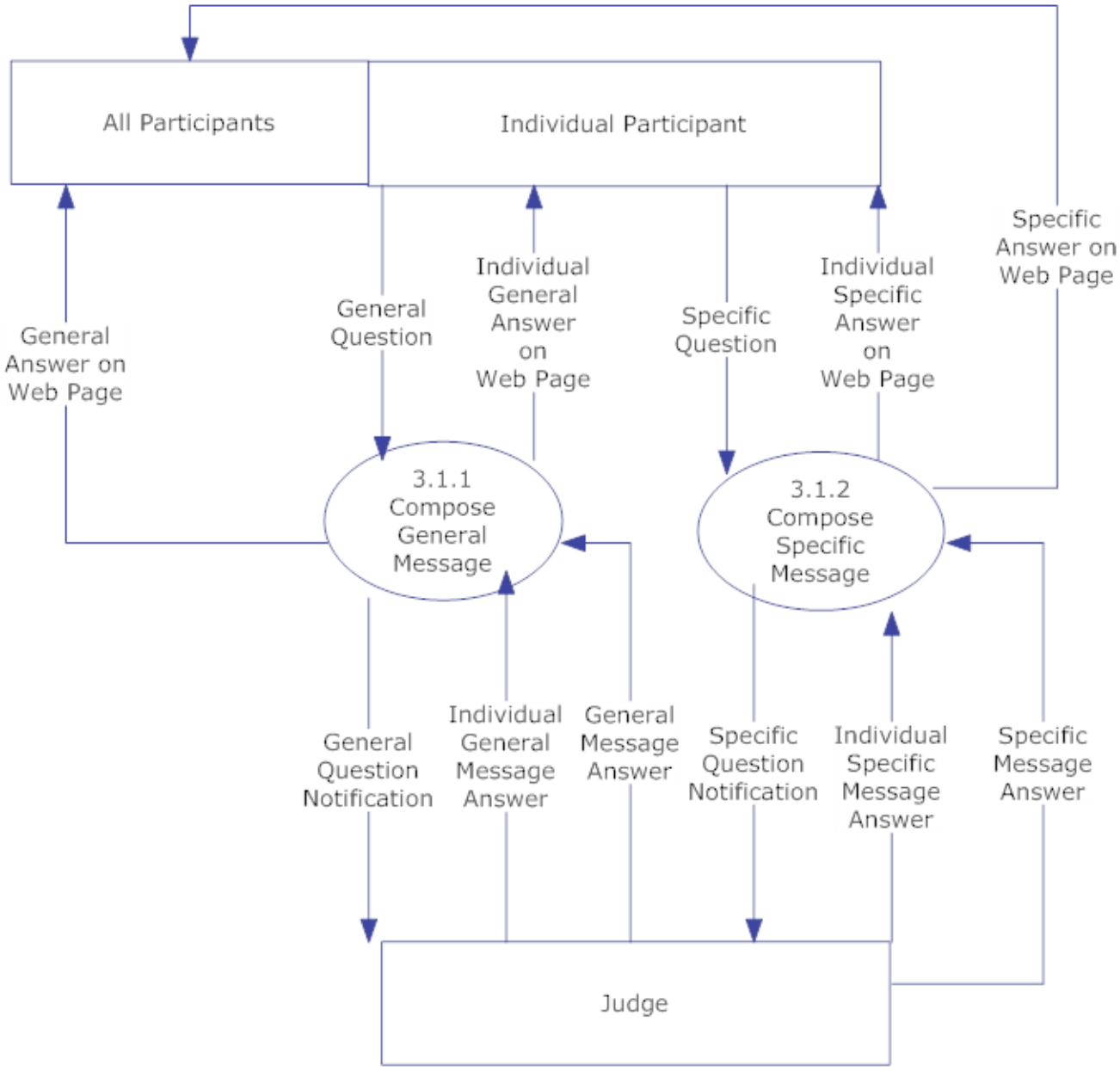
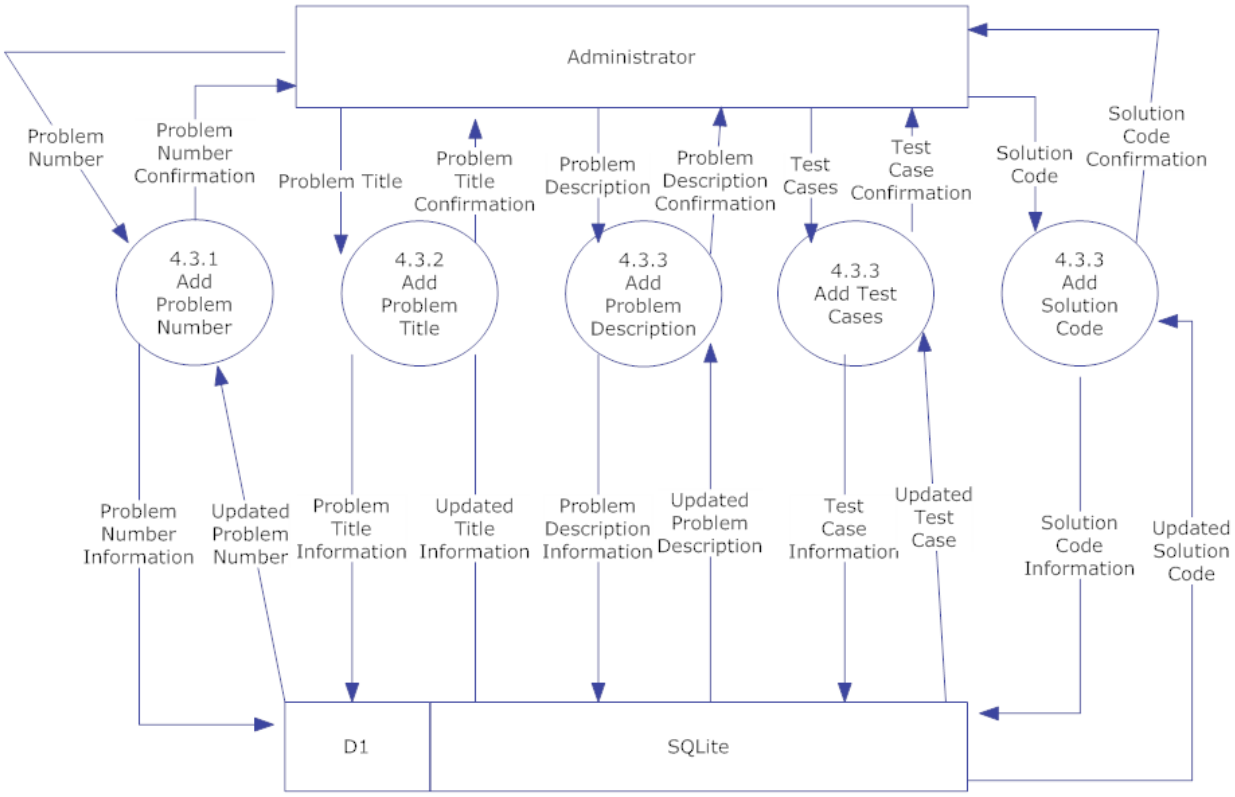
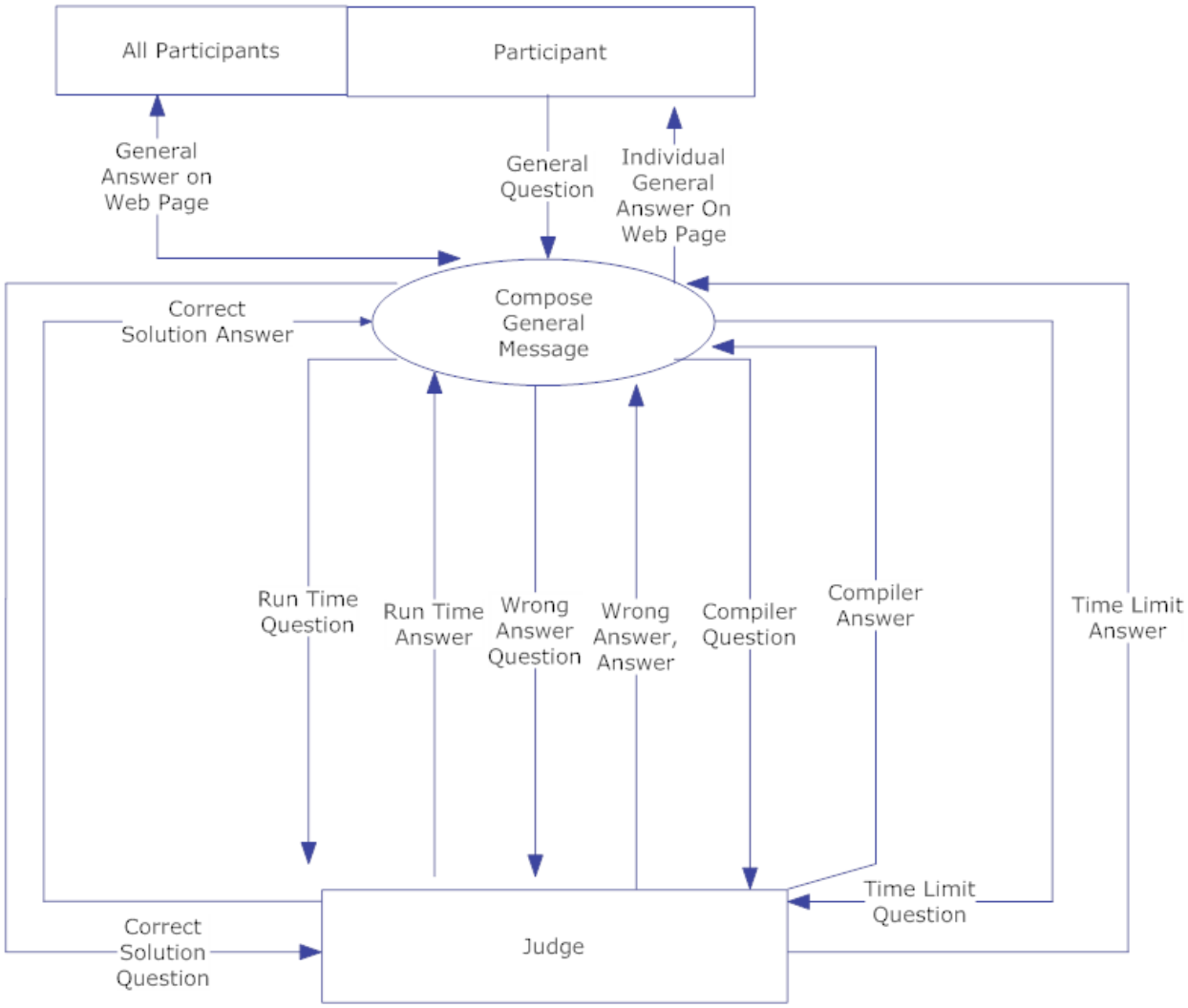


Figure 1.18: Level 2 Diagram of Define Problem Set



### 1.9.6 Level 3 Diagrams

Figure 1.19: Level 3 Diagram of Notify



## 1.10 Logical Data Dictionary Description

A Logical Data Dictionary is an inventory of all data entities and variables used within a system. All data entered, created, changed or stored by the system is included in the Logical Data Dictionary. Our data dictionary contains the following columns for each data entity described below:

- Data Name - The name of the data entity
- Applicable to - The views that the data is used or displayed in
- Data Type - How the data entity will be stored.
- Data Size - How large that data entity can be
- Description - What the data entity is used for
- Acceptable Input - A descriptor of allowed input
- Good Example of Input - An acceptable example input
- Notes - Additional comments on an entity The data dictionary is available on the $P = NP_{solutions}$ website as a Microsoft Excel File.

The Data Dictionary is located in Appendix B.

## 1.11 Prototype Screens

The following images were used for feature discovery and outline what SLICE will look like ascetically.

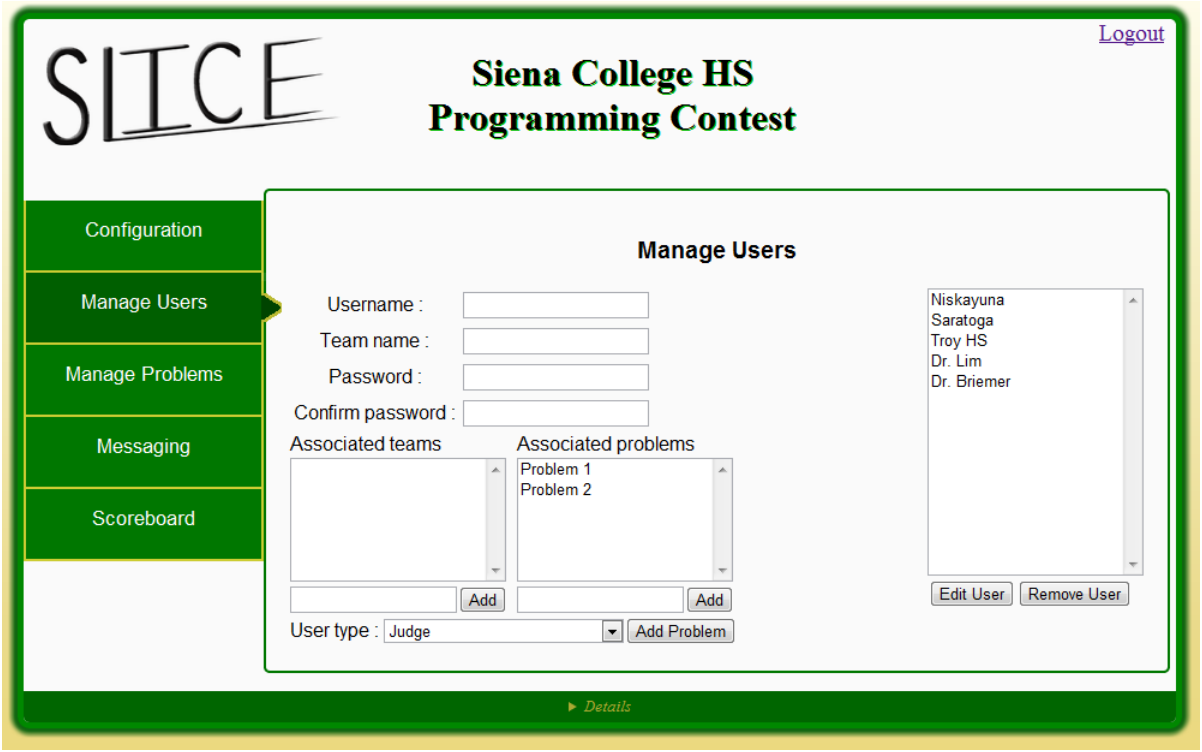
### 1.11.1 General View

**Login Screen for SLICE**

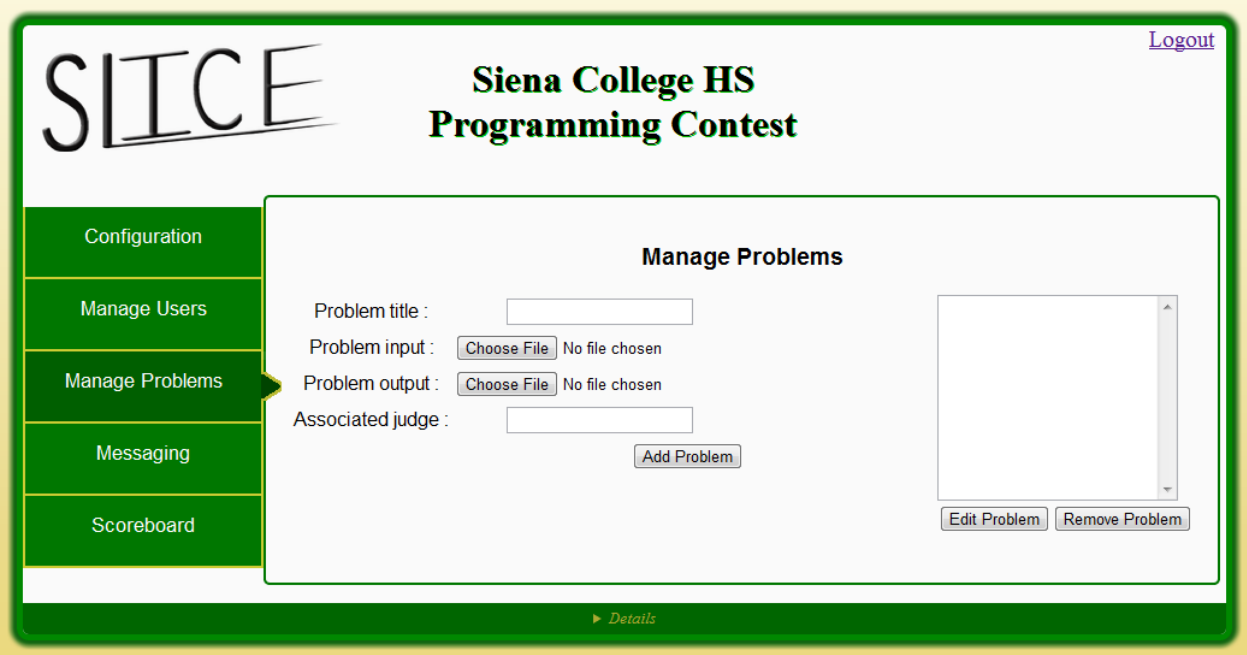
### 1.11.2 Administrative User Views

**Contest Configuration Page**

## Manage Users Page

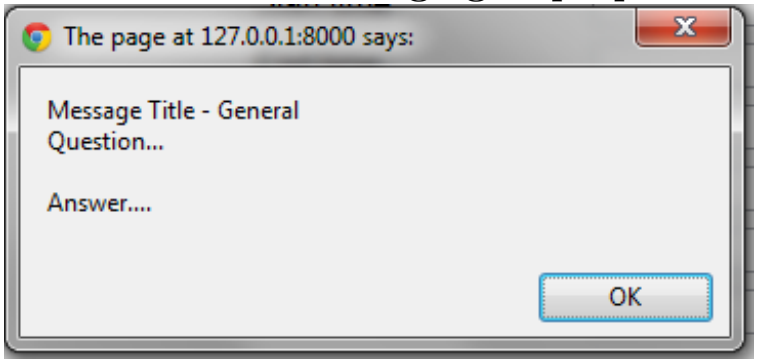


## Manage Problems

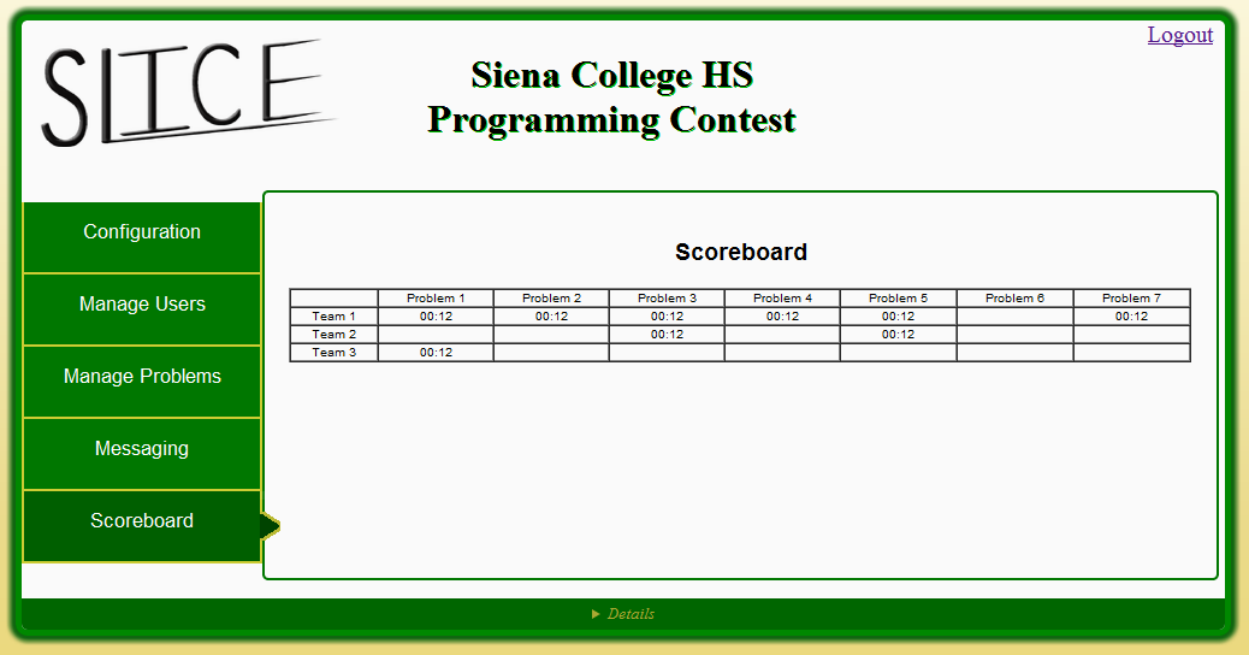


## Messaging Page


## Administrator Messaging Pop-up View



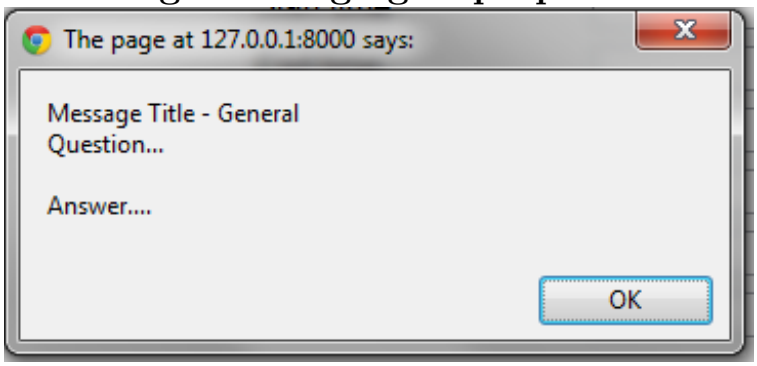
## Scoreboard Page



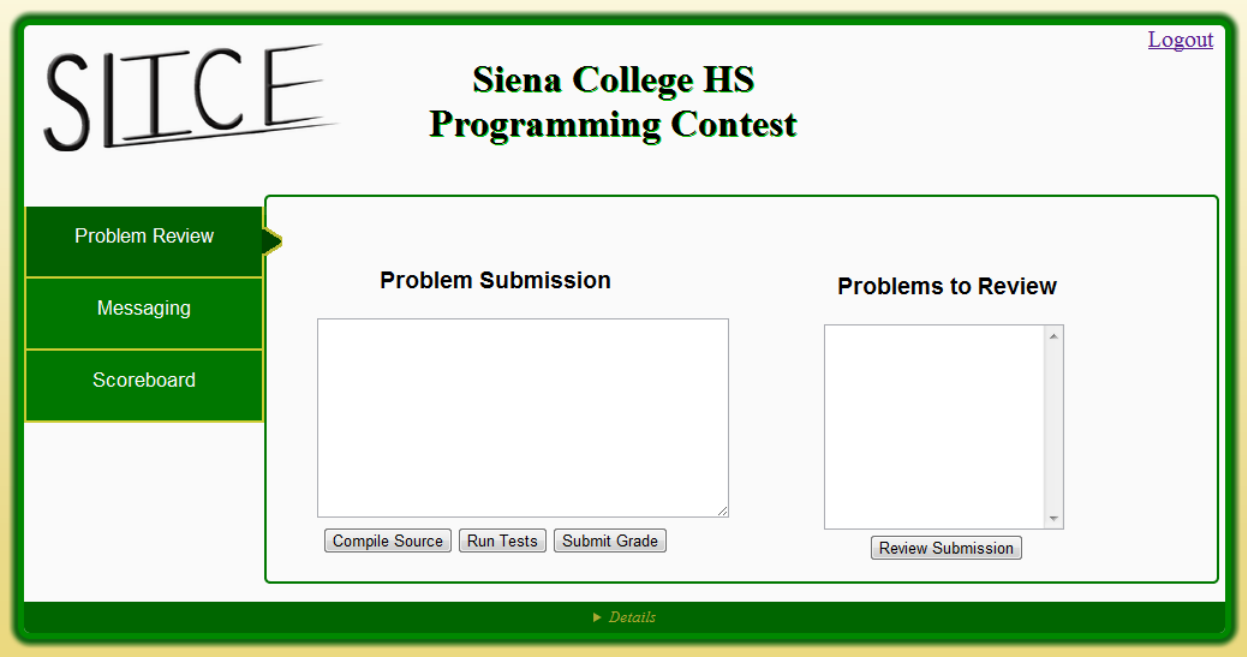
### 1.11.3 Judge User Views

**Message Page**

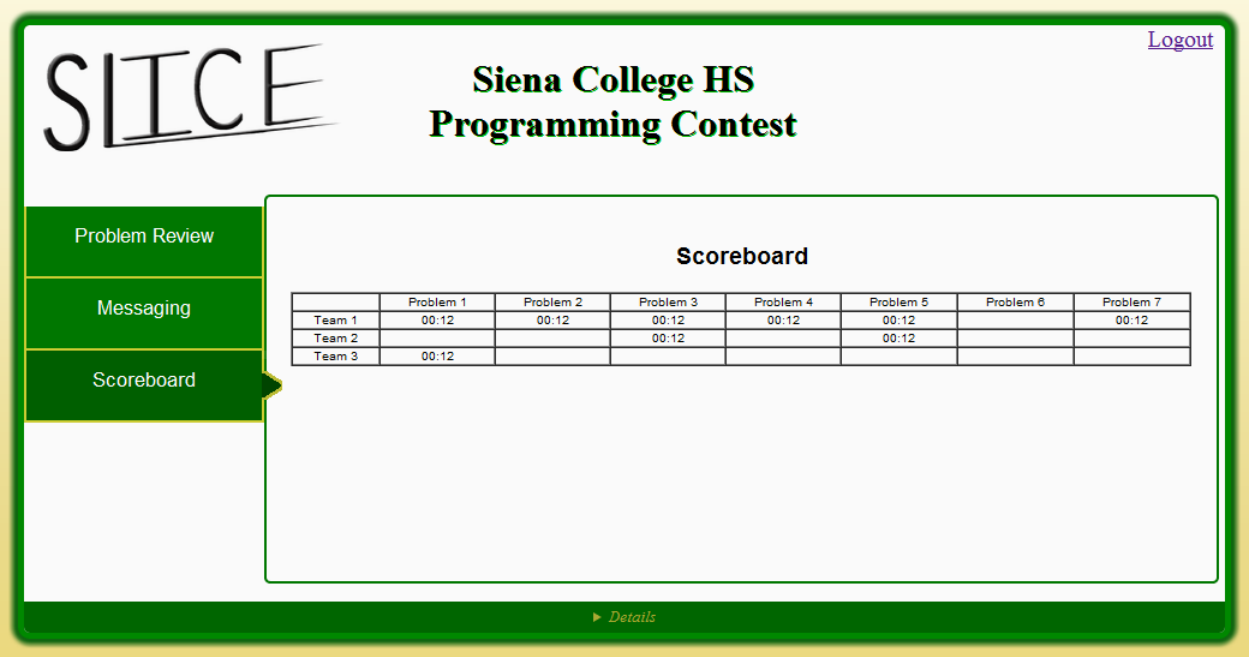

**Judge Messaging Pop-up View**



## Problem Review Page

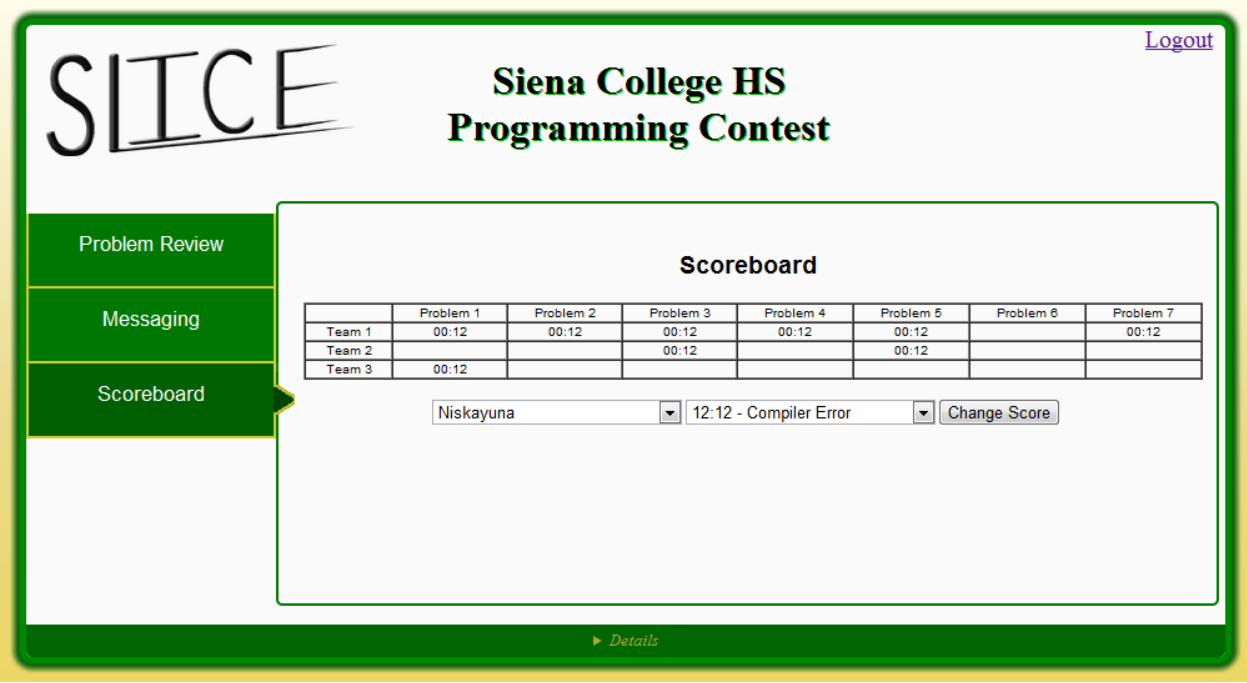


## Scoreboard Page



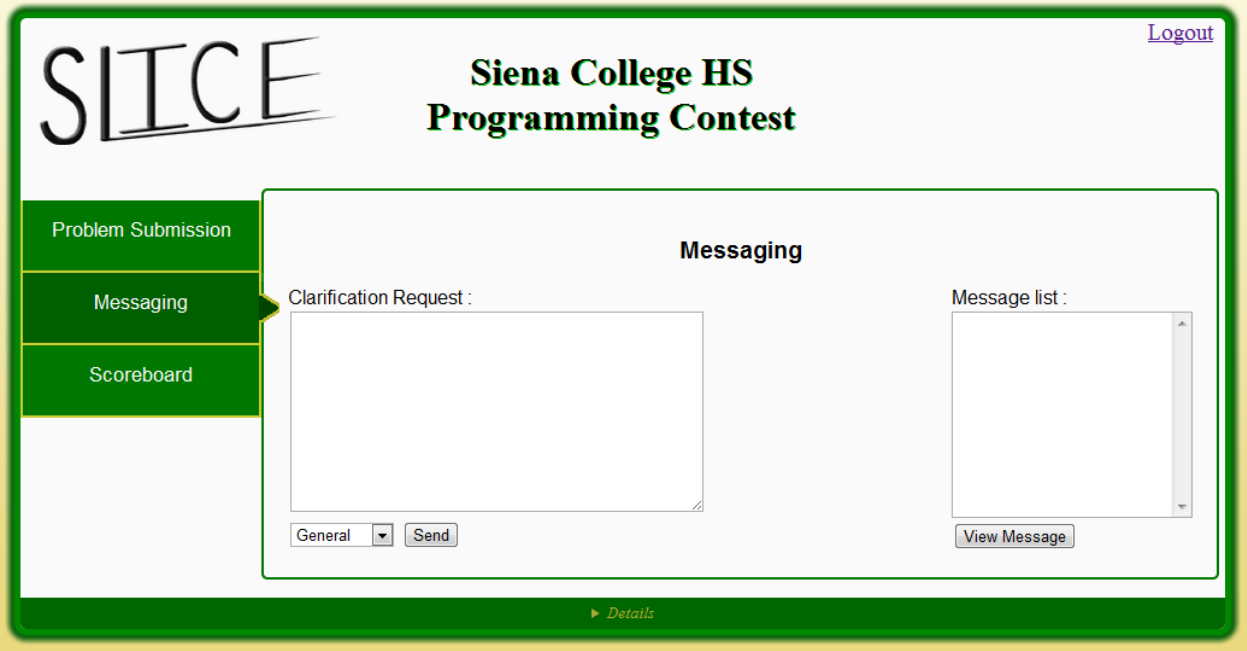

| | Problem 1 | Problem 2 | Problem 3 | Problem 4 | Problem 5 | Problem 6 | Problem 7 |
|---|---|---|---|---|---|---|---|
| Team 1 | 00:12 | 00:12 | 00:12 | 00:12 | 00:12 | | 00:12 |
| Team 2 | | | 00:12 | | 00:12 | | |
| Team 3 | 00:12 | | | | | | |

## Scoreboard Judge Scoreboard View

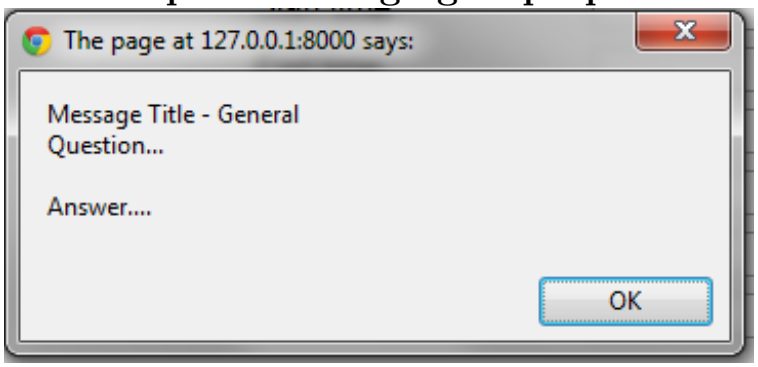


### 1.11.4 Participant User Views

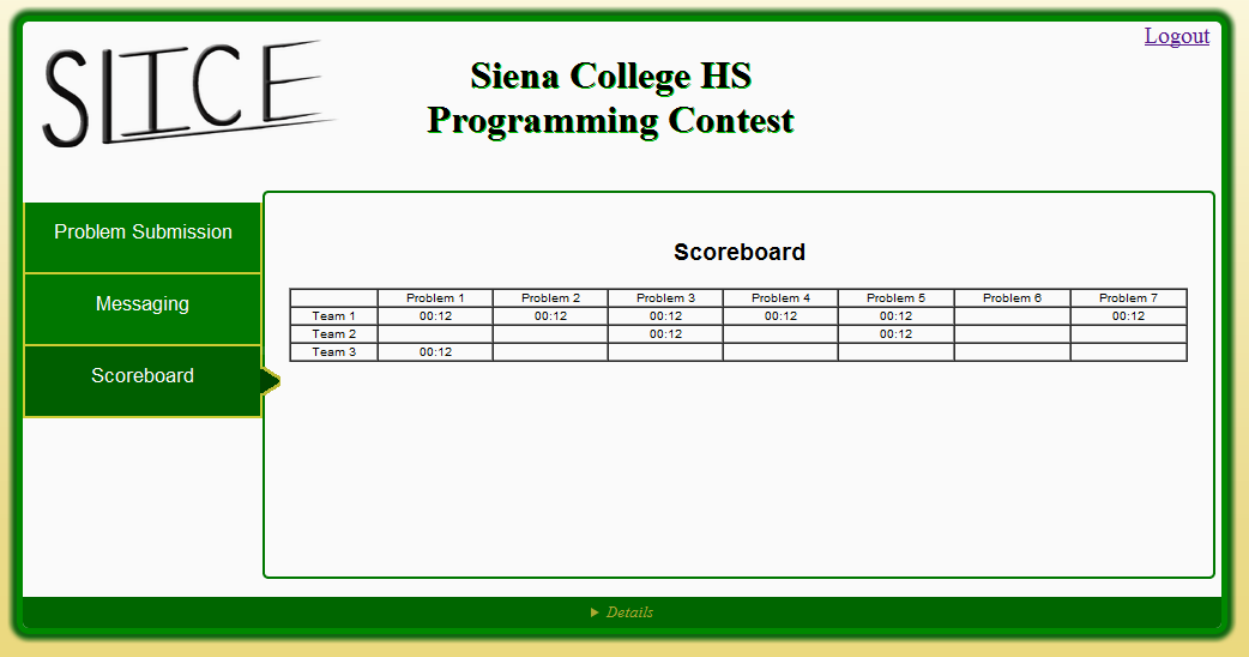
#### Messaging Page




#### Participant Messaging Pop-up View

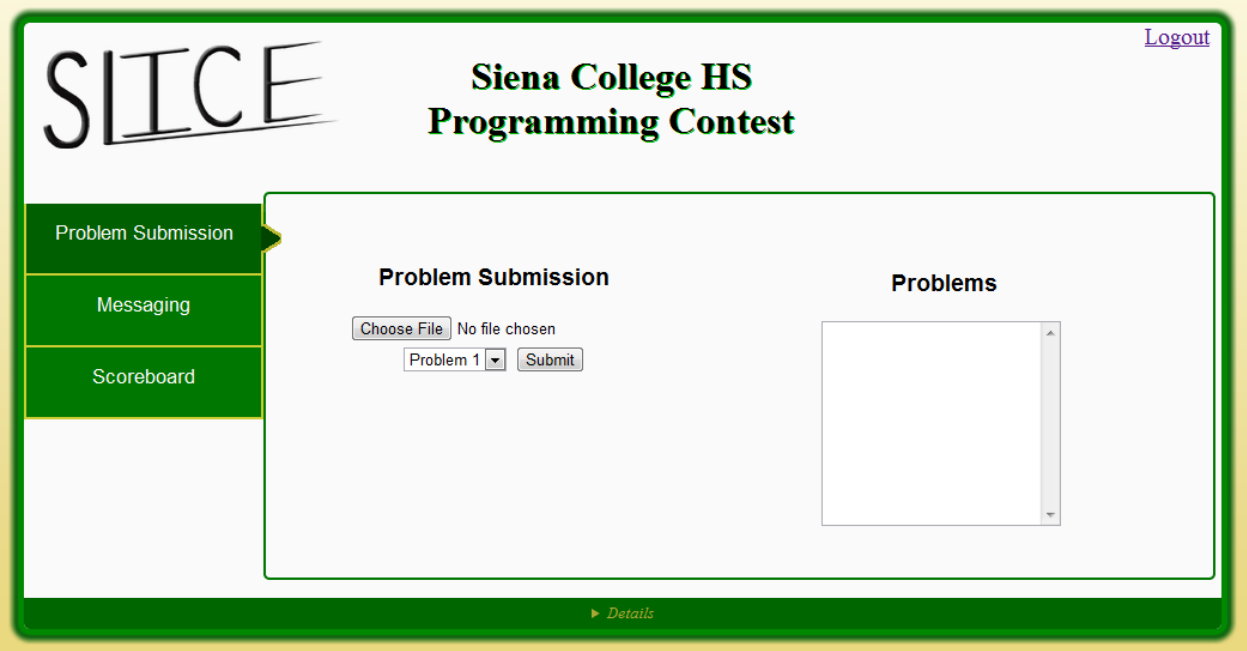


## Scoreboard Page




| | Problem 1 | Problem 2 | Problem 3 | Problem 4 | Problem 5 | Problem 6 | Problem 7 |
|---|---|---|---|---|---|---|---|
| Team 1 | 00:12 | 00:12 | 00:12 | 00:12 | 00:12 | | 00:12 |
| Team 2 | | | 00:12 | | 00:12 | | |
| Team 3 | 00:12 | | | | | | |

## Submisson Page



## 1.12 Integration and Regression Testing

### 1.12.1 Integration Testing

Integration testing involves the combination and testing of individual software modules as a group. The individual modules themselves will have already been unit tested. This ensures that any problems that arise within integration testing are a result of integration, not the individual software modules. This enables the system as a whole to perform higher order functions.

### 1.12.2 Regression Testing

Regression testing seeks to uncover any defects in existing functionality after changes have been made to the system. These changes can be functional enhancements, patches, or configuration changes. Regression testing ensures that any changes implemented do not create any new defects. This will be implemented by redoing previous tests and observing the results.

## 1.13 Development and Production Environment

Our PC is an Optiplex 760 (name: seb2)

- Windows Vista Enterprise (32 bit) with service pack 1 installed
- An Intel Core2 Duo CPU processor (E7500 @ 2.93GHz)
- 4 GB of Memory

Our Mac is an iMac (model identifier: iMac5,1)

- Mac OS X, version 10.6.4
- An Intel Core 2 Duo Processor (667 MHz)
- 1 GB of Memory

Our server is an x86_64 PC

- Hostname: oraserv.cs.siena.edu
- CentOS 5.2 (final)
- Kernal: 2.6.18-92.el5
- Intel Xeon 2.66 GHz CPU
- 8 GB of Memory
- Java SE Runtime Environment (build 1.6.0_10-rc-b28)
- GCC Version 4.1.2 20071124 (Red Hat 4.1.2-42)
- Python 2.7.2
- Django 1.4 pre-alpha
- Python setuptools 0.6
- mod_wsgi Apache module

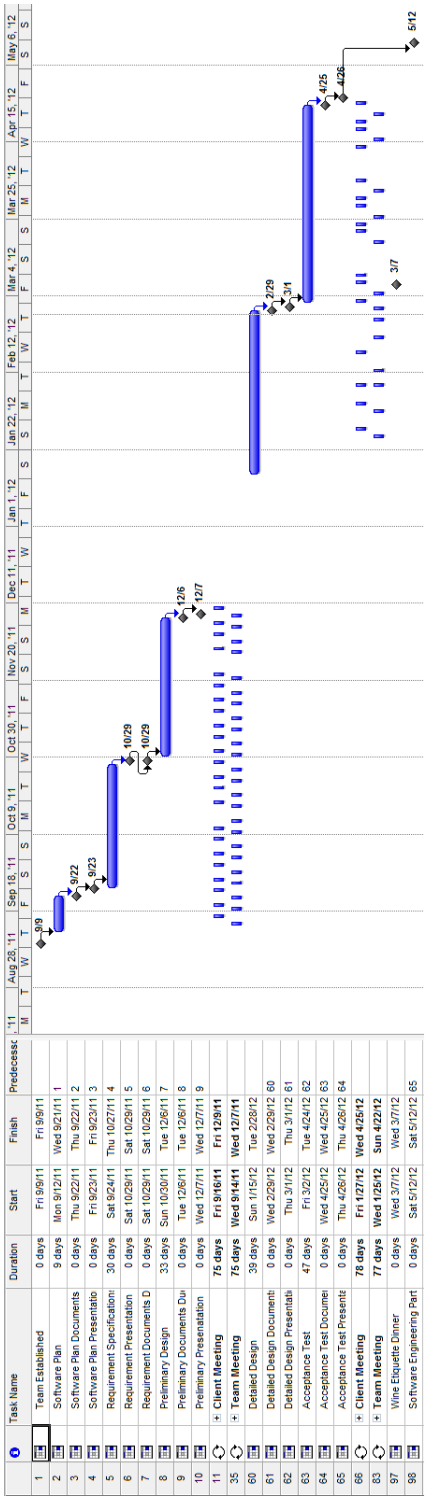
Our operating environment is a web based application that is only to be used within the SoS (Siena School of Science) network. The maintenance environment includes all of the hardware and software that is used to modify SLICE both in programming code and in visual appearance. This includes the computers in the Software Engineering Lab (stated above) and the software Adobe Dreamweaver, Adobe Fireworks, Adobe Photoshop, Internet Explorer, Mozilla Firefox, Google Chrome, Safari, etc

# Appendix A

# Glossary of Terms and Gantt Chart

- Actor: An entity in UML Use Case Diagrams and UML Activity Diagrams. It represents the human and non-human external entities that interact with the system
- Activity Diagram: A diagram based on the Unified Model Language (UML). This represents the processes that comprise a certain activity within the system. These diagrams are generally created with the perspective of an actor in mind.
- Compiler: A program that reads in source code and generates an executable.
- Data Flow Diagrams: Used to show how data is moved and processed within a system. There are various levels, each providing more detail then the next.
- Data Flows: A component of a Data Flow Diagram that represents the movement of data.
- Data Stores: A component of a Data Flow Diagram that represents a location in which information or data is stored.
- Django: A Python Web framework
- External Entities: A component of a Data Flow Diagram that represents any human or non-human user of a Software System.

- Functional Requirements Inventory: Define what the system will be able to do and what is testable about the system.
- Hardware: The physical parts of a computer, such as the hard drive and the CPU.
- HTML: Hypertext Markup Language is the scripting language used to describe the information contained on a website. HTML utilizes Cascading Style Sheets (CSS) to generate the style of the page. HTML and CSS are parsed by web browsers, such as Internet Explorer and Firefox, to render the websites for users Software: The intangible components of a computer and server. It is a set of machine-level instructions that is run from within the memory, and is used to perform a specific set of functions. Examples include Microsoft Word, Adobe Photoshop, and Mozilla Firefox.
- Microsoft Internet Explorer a web-browser developed by Microsoft
- Non-Functional Requirements: Specifies how a product is supposed the be in relationship to the functional requirements.
- Process: A component of a Data Flow Diagram that represents an activity that transforms or manipulates data.
- UML: Unified Modeling Language is the industry-standard language for the specification. Visualization, construction, and documentation of the components of software systems.
- Unit Testing: a method by which individual units of source code are tested to determine if they are fit for use. A unit is the smallest testable part of an application
- Use Case Diagram: Represents the high-level functions of the system. Also depicts how actors interact with each of those functions.
- User Case Narrative: an explanation of the functions and abilities users have for a specific Software System.



| | Task Name | Duration | Start | Finish | Predecessors |
|---|---|---|---|---|---|
| 1 | Team Established | 0 days | Fri 9/9/11 | Fri 9/9/11 | |
| 2 | Software Plan | 9 days | Mon 9/12/11 | Wed 9/21/11 | 1 |
| 3 | Software Plan Documents | 0 days | Thu 9/22/11 | Thu 9/22/11 | 2 |
| 4 | Software Plan Presentatio | 0 days | Fri 9/23/11 | Fri 9/23/11 | 3 |
| 5 | Requirement Specification | 30 days | Sat 9/24/11 | Thu 10/27/11 | 4 |
| 6 | Requirement Presentation | 0 days | Sat 10/29/11 | Sat 10/29/11 | 5 |
| 7 | Requirement Documents D | 0 days | Sat 10/29/11 | Sat 10/29/11 | 6 |
| 8 | Preliminary Design | 33 days | Sun 10/30/11 | Tue 12/6/11 | 7 |
| 9 | Preliminary Documents Du | 0 days | Tue 12/6/11 | Tue 12/6/11 | 8 |
| 10 | Preliminary Presenatation | 0 days | Wed 12/7/11 | Wed 12/7/11 | 9 |
| 11 | **Client Meeting** | **75 days** | **Fri 9/16/11** | **Fri 12/9/11** | |
| 35 | **Team Meeting** | **75 days** | **Wed 9/14/11** | **Wed 12/7/11** | |
| 60 | Detailed Design | 39 days | Sun 1/15/12 | Tue 2/28/12 | |
| 61 | Detailed Design Documenti | 0 days | Wed 2/29/12 | Wed 2/29/12 | 60 |
| 62 | Detailed Design Presentati | 0 days | Thu 3/1/12 | Thu 3/1/12 | 61 |
| 63 | Acceptance Test | 47 days | Fri 3/2/12 | Tue 4/24/12 | 62 |
| 64 | Acceptance Test Documei | 0 days | Wed 4/25/12 | Wed 4/25/12 | 63 |
| 65 | Acceptance Test Presenta | 0 days | Thu 4/26/12 | Thu 4/26/12 | 64 |
| 66 | **Client Meeting** | **78 days** | **Fri 1/27/12** | **Wed 4/25/12** | |
| 83 | **Team Meeting** | **77 days** | **Wed 1/25/12** | **Sun 4/22/12** | |
| 97 | Wine Etiquette Dinner | 0 days | Wed 3/7/12 | Wed 3/7/12 | |
| 98 | Software Engineering Part | 0 days | Sat 5/12/12 | Sat 5/12/12 | 65 |

# Appendix B

# Data Dictionary

Below are the screen shots of P=NP solutions Data Dictionary excel file.

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | **Team Name** | **P=NP Solutions** | | | | | | |
| 2 | **Project Name** | **S.L.I.C.E.** | | | | | | |
| 3 | **Client Name** | **Dr. Darren Lim** | | | | | | |
| 4 | | | | | | | | |
| 5 | Data Name | Applicable to | Data Type | Data Size | Description | Acceptable Input | Good Example of Input | Notes |
| 11 | currTime | Authenticate, Submission, Scoring, Messaging, Scoreboard, Configuration | String | 5 Chars | Current Time in Contest | 0-9,: | 01:00 | |
| 12 | resButton | Authenticate | boolean | 4-5 Chars | Login Reset | TRUE, FALSE | TRUE | |
| 13 | resButton2 | Configuration | boolean | 4-5 Chars | Configuration Reset | TRUE, FALSE | TRUE | |
| 14 | pracStart | Authenticate, Submission, Scoring, Messaging, Scoreboard, Configuration | String | 5 chars | Practice start-time | 0-9, : | 12:50 | |
| 15 | pracEnd | Authenticate, Submission, Scoring, Messaging, Scoreboard, Configuration | String | 5 chars | Practice end-time | 0-9, : | 13:00 | |
| 16 | pracDur | Authenticate, Submission, Scoring, Messaging, Scoreboard, Configuration | String | 2 Chars | Practice duration in minutes | 0-9 | 10 | |
| 17 | contStart | Authenticate, Submission, Scoring, Messaging, Scoreboard, Configuration | String | 5 chars | Contest start-time | 0-9, : | 13:00 | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 18 | contEnd | Authenticate, Submission, Scoring, Messaging, Scoreboard, Configuration | String | 5 chars | Contest end-time | 0-9, ; | 17:00 | |
| 19 | contDur | Configuration | String | 2 Chars | Practice duration in hours | 0-9 | 5 | |
| 20 | submissionState | Submission, Scoring, Scoreboard | boolean | 4-5 Chars | A problem submission's graded state | TRUE, FALSE | TRUE | |
| 21 | submissionTime | Submission, Scoring, Scoreboard | String | 8 Chars | A problem submission's time-stamp | 0-9, ; | 13:33:33 | Automatically generated by SLICE |
| 22 | submissionFeed | Submission, Scoring, Scoreboard | Enum | Enumerated Type | Submission Feedback | Correct Answer, Wrong Answer, Run-Time Error, Compiler Error, Formatting Error | Correct Answer | |
| 23 | scoreFreeze | Configuration, Scoreboard | String | 5 chars | Time from contest end time the scoreboard is inactive to participants | 0-9, ; | 02:00 | |
| 24 | judgeList | Configuration Judge Pop-Up | List of Strings | Up to 15 Strings | The list of all Judge users | A-Z, a-z, 0-9 | {Judge1, Judge 2} | |
| 25 | participantList | Configuration Participant/Advisor Pop-Up | List of Strings | Up to 60 Strings | The list of all participants and team advisors | A-Z, a-z, 0-9 | {sienaGold1, sienaGold1Adv} | |
| 26 | selectedUser | Configuration Participant/Advisor Pop-Up and Judge Pop-Up | String | 8-15 Chars | The selected user in the configuration Pop-Up | A-Z, a-z, 0-9 | seinaGold1 | |
| 27 | editUser | Configuration Participant/Advisor Pop-Up and Judge Pop-Up | boolean | 4-5 Chars | Edit the currently selected user | TRUE, FALSE | TRUE | |
| 28 | isTeamAdvisor | Configuration Participant/Advisor Pop-Up | boolean | 4-5 Chars | Flag the user as a Team Advisor | TRUE, FALSE | TRUE | |

| 5 | Data Name | Applicable to | Data Type | Data Size | Description | Acceptable Input | Good Example of Input | Notes |
|---|---|---|---|---|---|---|---|---|
| 29 | currPass | Administrator Change Password Pop-Up | String | 8-15 Chars | Password | ASCII 48-126 | r0ll1ngthun3;! | |
| 30 | newPass | Administrator Change Password Pop-Up | String | 8-15 Chars | Password | ASCII 48-126 | r0ll1ngthun3;! | |
| 31 | confirmPass | Administrator Change Password Pop-Up | String | 8-15 Chars | Password | ASCII 48-126 | r0ll1ngthun3;! | |
| 32 | personalMessList | Administrator Messaging, Participant Messaging, Judge Messaging | List of (String,String) Tuples | Up to 100 Messages | Personally sent messages | (ASCII 48-126, ASCII 48-126) | {(Does the output for problem 1 needs a new line?, Problem 1), (What is the example output for problem 2?, Problem 2)} | |
| 33 | publicMessList | Administrator Messaging, Participant Messaging, Judge Messaging, Team Advisor Messaging | List of (String,String) Tuples | Up to 100 Messages | Publically send messages | (ASCII 48-126, ASCII 48-126) | {(Please send only source files, not .class files., General)} | |
| 34 | selectedMess | Administrator Messaging, Participant Messaging, Judge Messaging, Team Advisor Messaging | (String, String) Tuple | (Up to 500 Chars, Up to 50 Chars) | The user selected message | ASCII 48-126 | (Please send only source files, not .class files., General) | |
| 35 | userMess | Administrator Messaging, Participant Messaging, Judge Messaging, Team Advisor Messaging | (String, String) Tuple | (Up to 500 Chars, Up to 50 Chars) | The user created message | ASCII 48-126 | (Does the output for problem 1 needs a new line?, Problem 1) | |
| 36 | scoresTable | Scoreboard | Two Dimensional Array of Strings | Number of Participants x Number of Problems | The Scoreboard information | A-Z, a-z, 0-9 | {{Team1,01:00,02:00}, {Team2,00:30,4:00}} | |
| 37 | isScoringJudge | Configuration, Scoreboard | boolean | 4-5 Chars | Flag for the Judge with scoreboard permissions | TRUE, FALSE | TRUE | |
| 38 | | | | | | | | |

# Appendix C

# Test Plan

## C.1 Testing Approach

The method our team, P=NP, will be using to test SLICE is to follow the guidelines outlined within this document. This method is to make certain that all Functional and Non-Functional requirements are met. The process will be a thorough, multi-step process validating SLICE. Smaller individual concepts all the way up to the completed system will all be tested.

Each step of this process consists of a Unit Test script. The script will be run repetitively until that test case is considered passed. Every test case within the Unit Test must pass. This process is done with each Unit Test. When each Unit Test is passed the Team will then test to see if the Units can work properly together.

Finally the system must be tested as a whole to guarantee that all requirements are met. On March 29th, 2012 P=NP will meet up with ExoNet to test each others projects on a small scale. We will then come back together on April 12, 2012 along with the rest of our software engineering class and others in the computer science department. This is so we can continue to test SLICE as a whole and to test its capability of managing 30 teams at once. We will perform a mock programming contest.

If at any point an error is found, P=NP will develop a solution and must repeat the testing procedure. This is to make sure that the problem is now fixed but to also test that the solution did not create a new problem.

Testing will be finished when everything is passed and P=NP has, as a team, determined all Functional and Non-Functional requirements have been met.

## C.2 Testing Plan Identifier

The reason for the Test Plan for S.L.I.C.E is it is necessary to test all the requirements to make sure the program is functional. The Test Plan will describe how the project is supposed to work. Throughout the testing process, the Test Plan will change when a requirement is altered or even removed. A detailed document will be provided showing changes to any requirements and if one had failed or passed during any point of the Test Plan. An updated version of this document will be provided at the end of the Detailed Design phase. The final version will be provided at the end of the Acceptance Test phase.

## C.3 Test Items and Function Requirements Inventory

The Functional Requirements will be tested using a detailed checklist. The checklist is used to guide the members of P=NP to assure that each requirement is tested and tested properly. The purpose of this is to guarantee that all of the clients, Dr. Lim, functional requirements are met and are met for every user.

We have five unique users of SLICE. They are the Team Advisor, Administrator, Judge, Scoreboard Judge, and Participant. We also have functional requirements for SLICE itself.

### C.3.1 Administrative

**Y/N** Will be able to log into SLICE with a unique username and password

**Y/N** Will be able to create the Judge accounts

**Y/N** Will be able to create the Participant accounts

**Y/N** Will be able to create the Team Advisor accounts

**Y/N** Will be able to give Scoreboard Judge abilities to one judge

**Y/N** Will be able to send broadcast messages to all users

**Y/N** Will be able to send broadcast messages to any subset of users

**Y/N** Will be able to manage the set of problems

**Y/N** Will be able to manage test input

**Y/N** Will be able to manage correct output for each contest problem

**Y/N** Will be able to designate the length of the contest

**Y/N** Will be able to set the contest start time

**Y/N** Will be able to set the contest stop time

**Y/N** Will be able to set the practice time for the contest

**Y/N** Will be able to restrict the view of the scoreboard
at a set time towards the end of the contest

**Y/N** Will be able to choose the list of allowed programming languages

**Y/N** Will be able to log out of SLICE at any point

### C.3.2 Team Advisor

**Y/N** Will be able to log into SLICE with the Advisor user name and password

**Y/N** Will be able to view each individual problem

**Y/N** Will be able to view the scoreboard while active

**Y/N** Will be able to receive all publically broadcasted contest clarifications

**Y/N** Will be able to log out of SLICE at any point

### C.3.3 Judge

**Y/N** Will be able to log onto SLICE with a given user name and password

**Y/N** Will be able to view Participant submitted solutions

**Y/N** Will be able to run the submitted solution using the test input provided in SLICE that corresponds to the associated question

**Y/N** Will be able to submit a response to the team stating either "Correct Solution, Compiler Error, Running Time Error, Wrong Answer, OR Time-Limit Exceeded"

**Y/N** Will be able to log out of SLICE at any point

### C.3.4 Scoreboard Judge

**Y/N** Will be able to do everything the Judge can do

**Y/N** Will be able to edit the scoreboard

**Y/N** Will be able to log out of SLICE at any point

### C.3.5 Participant

**Y/N** Will be able to log into SLICE with a given user name and password

**Y/N** Will be able to view the contest scoreboard while active

**Y/N** Will be able to submit problem-solutions

**Y/N** Will be able to submit clarification-requests

**Y/N** Will be able to make a general request

**Y/N** Will be able to associate a problem with request

**Y/N** Will be able to select on of the contest problems

**Y/N** Will be able to select one of a list of programming languages

**Y/N** Will be able to receive individual clarification responses from any Judge

**Y/N** Will be able to receive publically broadcasted messages from any Judge or Administrative User.

**Y/N** Will be able to log out of SLICE at any point

### C.3.6 SLICE

**Y/N** Will be able to compile submitted problems

**Y/N** Will be able to provide an appropriate error message when a wrong username and password is enter

**Y/N** Will be able to handle up to 30 teams

## C.4 Non-Functional Requirements

Non-Functional Requirements are what SLICE is and looks like rather than what it does. These are harder to test but just as important as the Functional Requirements.

**Y/N** The system will be easily maintained

**Y/N** The system will be stable.

**Y/N** The system will be viewable on multiple browsers

**Y/N** The system will run efficiently

**Y/N** The system will be user friendly

## C.5 Exception Handling To Test

SLICE will be designed as a secure system. All users must authenticate to have access to SLICE. Participants may only submit source code of problem solutions. The solutions are compiled by a Judge on a central server under a restricted account environment separate from the contest problems test input and output. There are also time limits set on both compile time and execution time of Participant problem solutions to avoid denial of service attacks on SLICE.

## C.6 Acceptance Test - Acceptance Criteria

The acceptance test is concerned with validating the functional and non-functional requirements of a system. Utilizing this testing process will validate the acceptability of a system. Once a system is declared complete the test plan document and the Unit test logs (described and shown below) are helpful tools to explain to those who were never involved in the project why and how the project is validated. The document validates the integrity of the steps a team took to show that the system meets the requirements and specifications.

The acceptance criteria for the system will be defined by the functional requirements inventory that was explained in further detail earlier in the document. The part of the system that is testable by users is considered the functional requirements. Non-functional requirements are not testable items, such as look or maintainability. By the end of the Acceptance Test some of the requirements may have altered or changed due to complications. Nonetheless, P=NP Solutions will state which of those requirements were met or not met.

SLICE will be tested on Internet Explorer, Mozilla Firefox, Google Chrome and Apple Safari. P=NP Solutions will be doing weekly tests on SLICE whenever a new feature is added. We will then be conducting a smaller test of about ten teams. Upon completing those two rounds of testing, we will then have a larger test of thirty different teams. P=NP will be using these methods of testing to determine if our stated requirements are met.

# Appendix D

# Unit Test

## D.1 Unit Tests

The following is a list of the separate Unit Tests that will be tested from now until the end of our project. Each Unit will either pass or fail. A full system test will be performed once all Unit Tests are passed.

**P=NP Solutions 8 Unit Test**

- User Login
- Problem Submission
- Contest Setup
- Private Messaging
- Public Messaging
- View Scoreboard
- Edit Scoreboard
- Log out

### D.1.1 Unit Test Cases

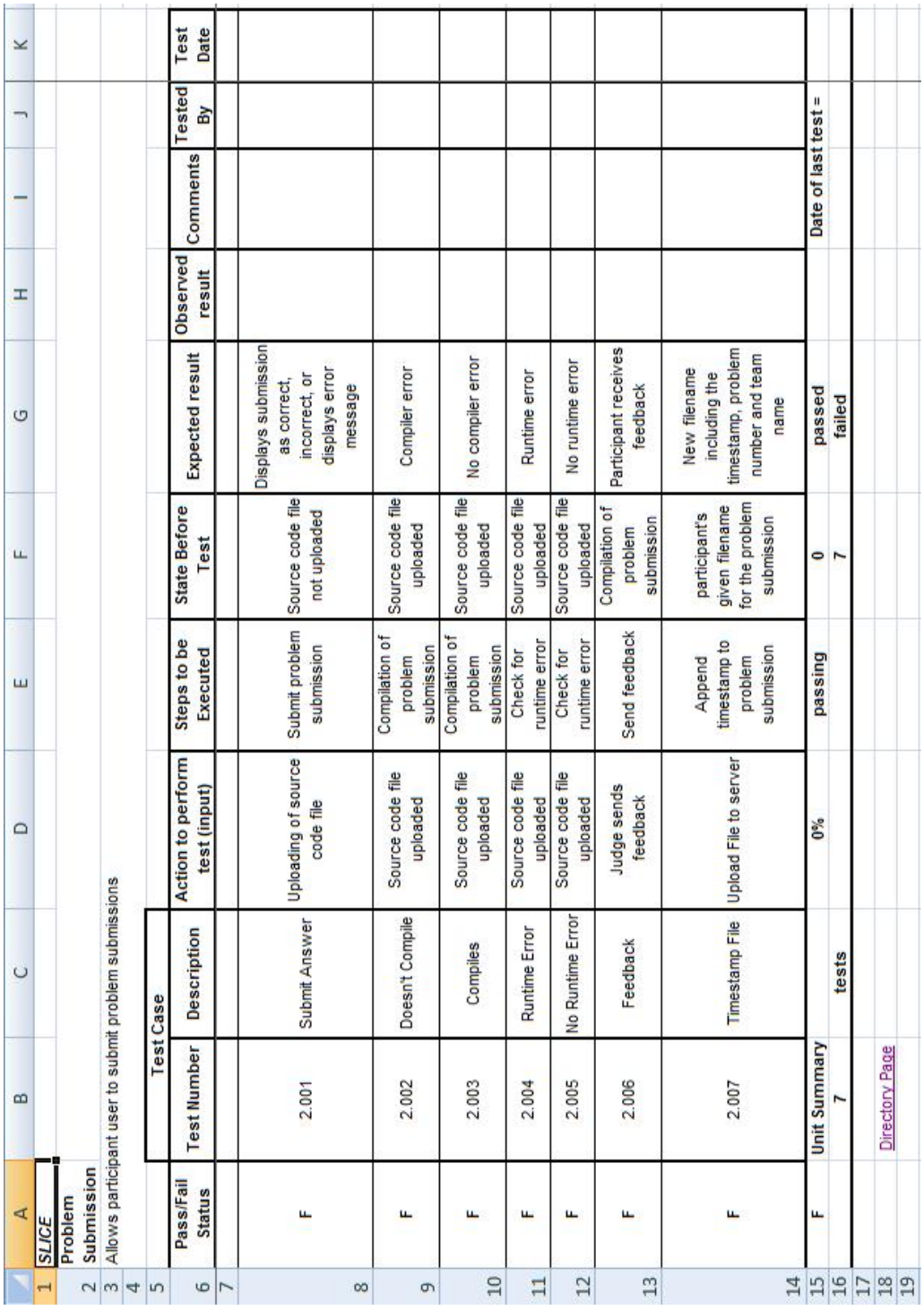
Each Unit Test is divided into test cases. Test Cases include a test number for tracking, along with a description of the purpose for the test case. The excel table also includes the action to be performed or the input, steps to be executed, the state before the test, and the expected resulted. There are also columns that are to be filled in when one is testing SLICE. There is a column to edit to say what happened when testing that specific test case.

Below are screen shots of the excel file of our 8 different Unit Tests with their Test Cases.

Figure D.1: Directory of the Unit Tests

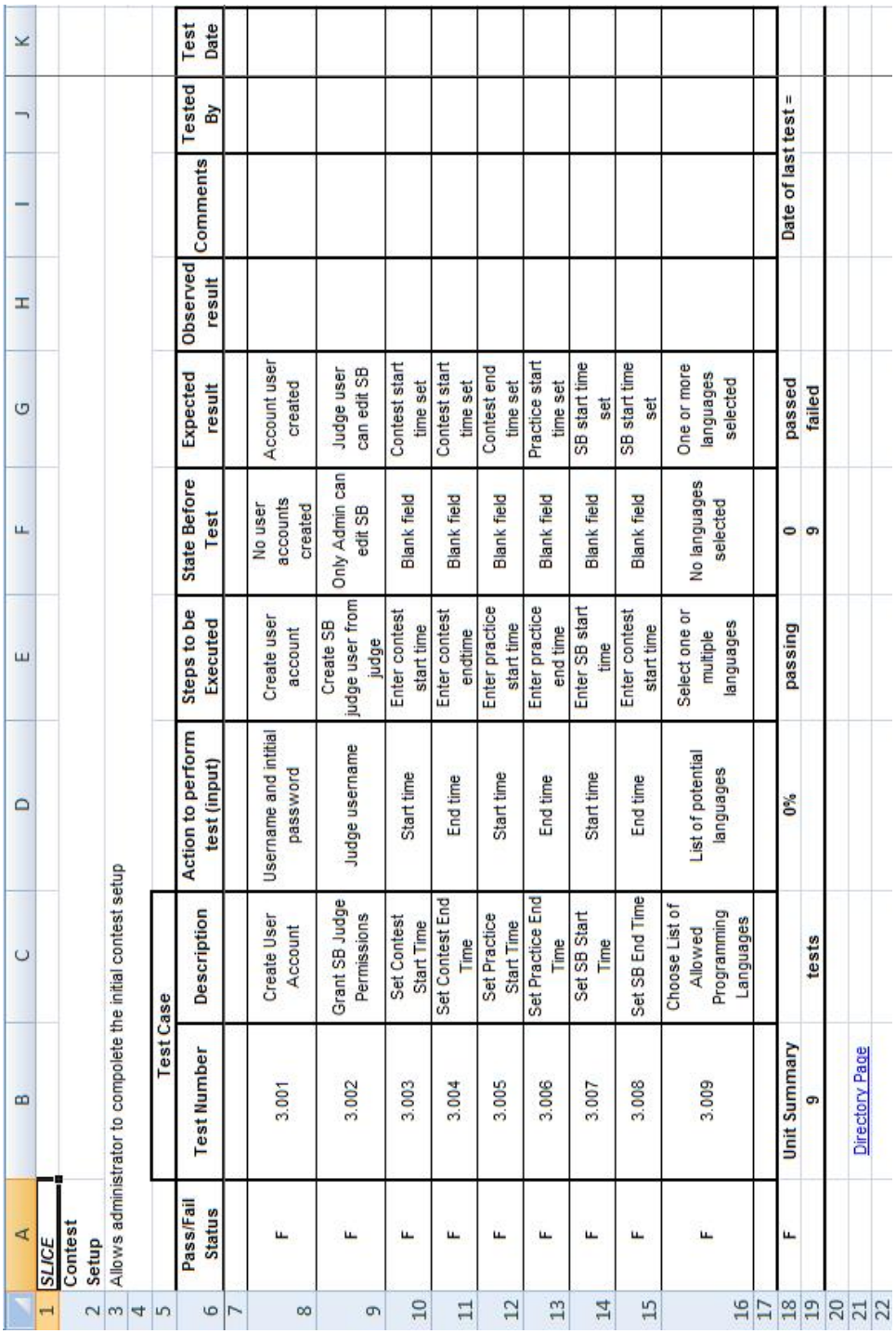
| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | ***System Test - Test Results*** | | | | | | |
| 2 | | | | | | | |
| 3 | **Team Name** | | **P=NP Solutions** | | | | |
| 4 | **Project Name** | | **S.L.I.C.E.** | | | | |
| 5 | **Client Name** | | **Dr. Darren Lim** | | | | |
| 6 | | | | | | | |
| 7 | **Unit Test Directory** | | | | | | |
| 8 | | | | | | | |
| 9 | **Pass/Fail Status** | | **Unit Number** | **Unit Test Name** | **Date Last Tested** | **Comments or description** | **Integrated with these units** |
| 10 | | | | | | | |
| 11 | P | F | 1 | User Login | | | All |
| 12 | P | F | 2 | Problem Submission | | | |
| 13 | P | F | 3 | Contest Setup | | | |
| 14 | P | F | 4 | Private Messaging | | | |
| 15 | P | F | 5 | Public Messaging | | | |
| 16 | P | F | 6 | View Scoreboard | | | |
| 17 | P | F | 7 | Edit Scoreboard | | | |
| 18 | P | F | 8 | Log Out | | | |
| 19 | | | | | | | |

Figure D.2: Problem Submission



| SLICE Problem Submission | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Allows participant user to submit problem submissions | | | | | | | | | | |
| | Test Case | | | | | | | | | |
| Pass/Fail Status | Test Number | Description | Action to perform test (input) | Steps to be Executed | State Before Test | Expected result | Observed result | Comments | Tested By | Test Date |
| F | 2.001 | Submit Answer | Uploading of source code file | Submit problem submission | Source code file not uploaded | Displays submission as correct, incorrect, or displays error message | | | | |
| F | 2.002 | Doesn't Compile | Source code file uploaded | Compilation of problem submission | Source code file uploaded | Compiler error | | | | |
| F | 2.003 | Compiles | Source code file uploaded | Compilation of problem submission | Source code file uploaded | No compiler error | | | | |
| F | 2.004 | Runtime Error | Source code file uploaded | Check for runtime error | Source code file uploaded | Runtime error | | | | |
| F | 2.005 | No Runtime Error | Source code file uploaded | Check for runtime error | Source code file uploaded | No runtime error | | | | |
| F | 2.006 | Feedback | Judge sends feedback | Send feedback | Compilation of problem submission | Participant receives feedback | | | | |
| F | 2.007 | Timestamp File | Upload File to server | Append timestamp to problem submission | participant's given filename for the problem submission | New filename including the timestamp, problem number and team name | | | | |
| F | Unit Summary | | 0% | passing | 0 | passed | | Date of last test = | | |
| | 7 | tests | | | 7 | failed | | | | |
| | Directory Page | | | | | | | | | |

Figure D.3: contest Setup



SLICE

Contest Setup

Allows administrator to compolete the initial contest setup

| | Test Case | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Pass/Fail Status | Test Number | Description | Action to perform test (input) | Steps to be Executed | State Before Test | Expected result | Observed result | Comments | Tested By | Test Date |
| F | 3.001 | Create User Account | Username and intitial password | Create user account | No user accounts created | Account user created | | | | |
| F | 3.002 | Grant SB Judge Permissions | Judge username | Create SB judge user from judge | Only Admin can edit SB | Judge user can edit SB | | | | |
| F | 3.003 | Set Contest Start Time | Start time | Enter contest start time | Blank field | Contest start time set | | | | |
| F | 3.004 | Set Contest End Time | End time | Enter contest endtime | Blank field | Contest start time set | | | | |
| F | 3.005 | Set Practice Start Time | Start time | Enter practice start time | Blank field | Contest end time set | | | | |
| F | 3.006 | Set Practice End Time | End time | Enter practice end time | Blank field | Practice start time set | | | | |
| F | 3.007 | Set SB Start Time | Start time | Enter SB start time | Blank field | SB start time set | | | | |
| F | 3.008 | Set SB End Time | End time | Enter contest start time | Blank field | SB start time set | | | | |
| F | 3.009 | Choose List of Allowed Programming Languages | List of potential languages | Select one or multiple languages | No languages selected | One or more languages selected | | | | |
| | | | | | | | | | | |
| F | Unit Summary | | 0% | passing | 0 | passed | | Date of last test = | | |
| | 9 | tests | | | 9 | failed | | | | |

Directory Page

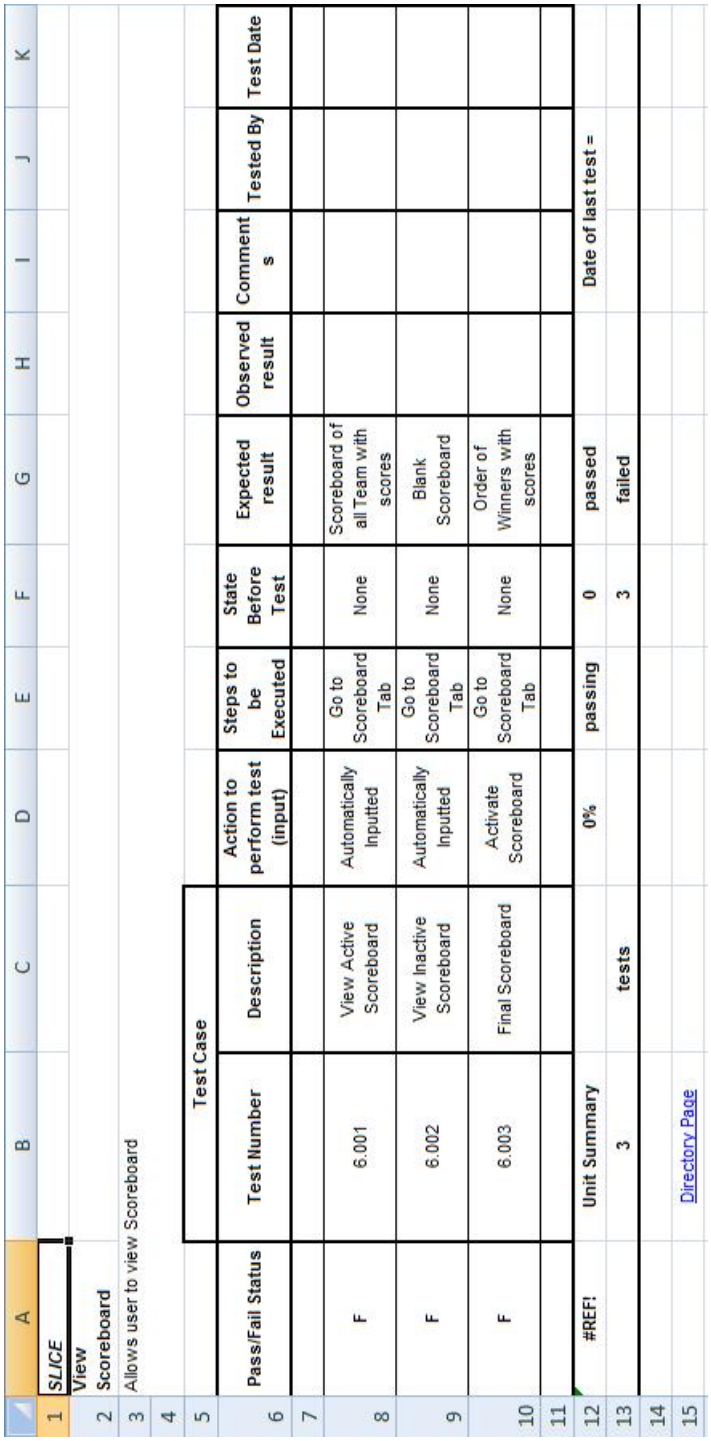
Figure D.4: Private Messaging

| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | SLICE | | | | | | | | | | |
| 2 | Private Messages | | | | | | | | | | |
| 3 | Allows user to send and receive private messages | | | | | | | | | | |
| 4 | | | | | | | | | | | |
| 5 | | Test Case | | | | | | | | | |
| 6 | Pass/Fail Status | Test Number | Description | Action to perform test (input) | Steps to be Executed | State Before Test | Expected result | Observed result | Comments | Tested By | Test Date |
| 7 | | | | | | | | | | | |
| 8 | F | 4.001 | Send Private Message (Fail) | Private message text | Send message text | No private message sent | Error message: "Delivery of message failed" | | | | |
| 9 | F | 4.002 | Send Private Message (Success) | Private message text | Send message text | No private message sent | Message: "Message sent" | | | | |
| 10 | F | 4.003 | Receive Private Message | Private message text | Send message text | No private message received | Reception of private message | | | | |
| 11 | | | | | | | | | | | |
| 12 | F | Unit Summary | | 0% | passing | 0 | passed | | Date of last test = | | |
| 13 | | 3 | tests | | | 3 | failed | | | | |
| 14 | | | | | | | | | | | |
| 15 | | Directory Page | | | | | | | | | |
| 16 | | | | | | | | | | | |

Figure D.5: Public Messaging

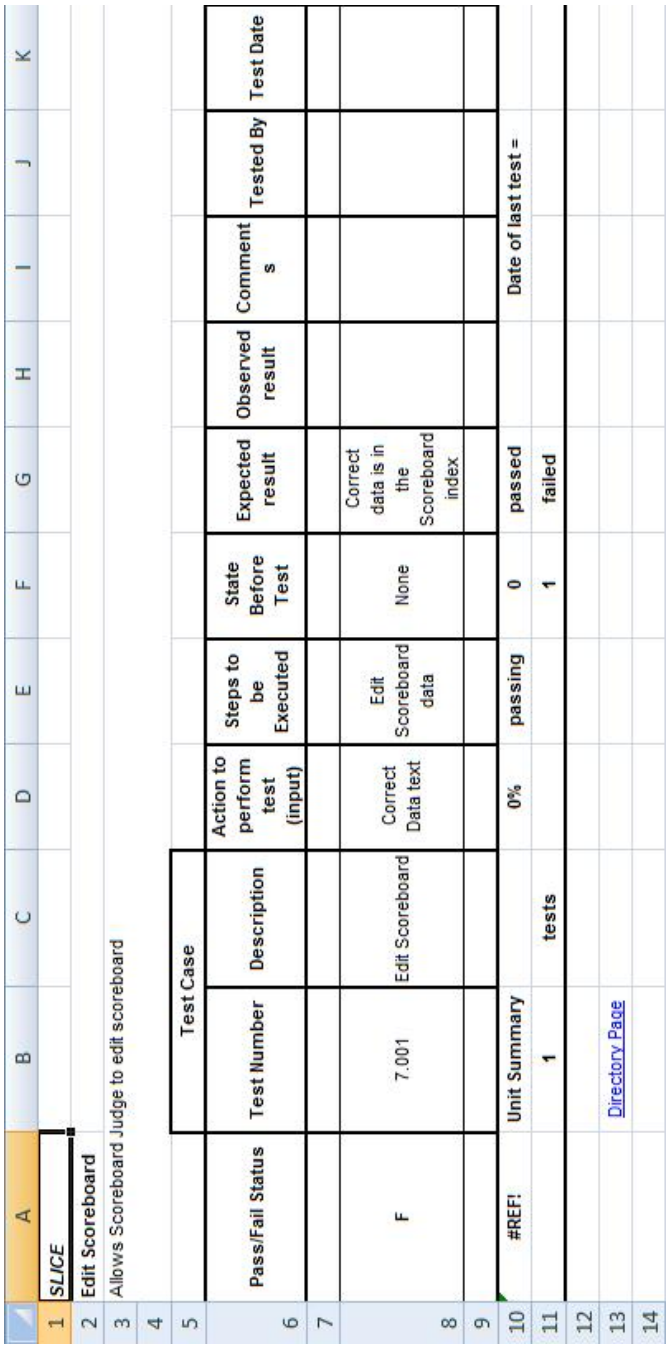
| SLICE | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Public Messaging | | | | | | | | | | |
| Allows user to send and receive public messages | | | | | | | | | | |
| | Test Case | | | | | | | | | |
| Pass/Fail Status | Test Number | Description | Action to perform test (input) | Steps to be Executed | State Before Test | Expected result | Observed result | Comments | Tested By | Test Date |
| F | 5.001 | Failed to Send Public Message | Public message text | Attempt to send public message | No public message sent | Error message: "Delivery of message failed" | | | | |
| F | 5.002 | Send Public Message (Success) | Public message text | Send public message | No public message sent | Message: "Message sent" | | | | |
| F | 5.003 | Receive Public Message | Public message text | Send message text | No public message received | Reception of publicmessage | | | | |
| #REF! | Unit Summary | | 0% | passing | 0 | passed | | Date of last test = | | |
| | 3 | tests | | | 3 | failed | | | | |
| | Directory Page | | | | | | | | | |

Figure D.6: View Scoreboard



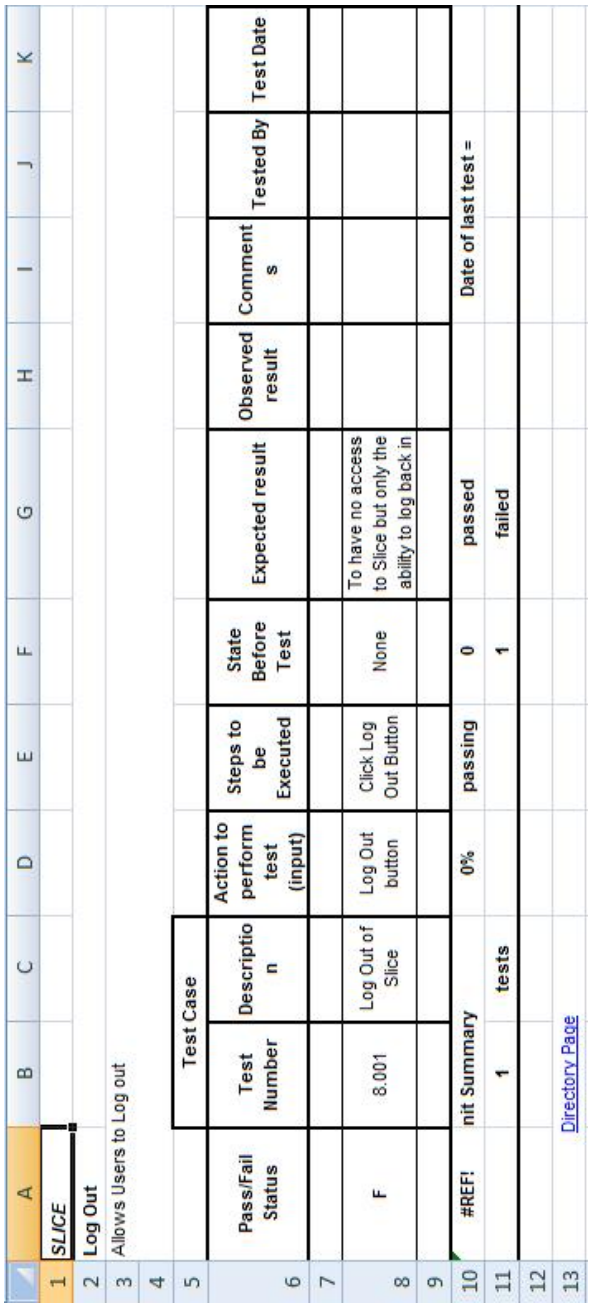
| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | SLICE | | | | | | | | | | |
| 2 | View Scoreboard | | | | | | | | | | |
| 3 | Allows user to view Scoreboard | | | | | | | | | | |
| 4 | | | | | | | | | | | |
| 5 | | Test Case | | | | | | | | | |
| 6 | Pass/Fail Status | Test Number | Description | Action to perform test (input) | Steps to be Executed | State Before Test | Expected result | Observed result | Comments | Tested By | Test Date |
| 7 | | | | | | | | | | | |
| 8 | F | 6.001 | View Active Scoreboard | Automatically Inputted | Go to Scoreboard Tab | None | Scoreboard of all Team with scores | | | | |
| 9 | F | 6.002 | View Inactive Scoreboard | Automatically Inputted | Go to Scoreboard Tab | None | Blank Scoreboard | | | | |
| 10 | F | 6.003 | Final Scoreboard | Activate Scoreboard | Go to Scoreboard Tab | None | Order of Winners with scores | | | | |
| 11 | | | | | | | | | | | |
| 12 | #REF! | Unit Summary | | 0% | passing | 0 | passed | | Date of last test = | | |
| 13 | | 3 | tests | | | 3 | failed | | | | |
| 14 | | | | | | | | | | | |
| 15 | | Directory Page | | | | | | | | | |

Figure D.7: Edit Scoreboard



| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | SLICE | | | | | | | | | | |
| 2 | Edit Scoreboard | | | | | | | | | | |
| 3 | Allows Scoreboard Judge to edit scoreboard | | | | | | | | | | |
| 4 | | | | | | | | | | | |
| 5 | | Test Case | | | | | | | | | |
| 6 | Pass/Fail Status | Test Number | Description | Action to perform test (input) | Steps to be Executed | State Before Test | Expected result | Observed result | Comments | Tested By | Test Date |
| 7 | | | | | | | | | | | |
| 8 | F | 7.001 | Edit Scoreboard | Correct Data text | Edit Scoreboard data | None | Correct data is in the Scoreboard index | | | | |
| 9 | | | | | | | | | | | |
| 10 | #REF! | Unit Summary | | 0% | passing | 0 | passed | | Date of last test = | | |
| 11 | | 1 | tests | | | 1 | failed | | | | |
| 12 | | | | | | | | | | | |
| 13 | | Directory Page | | | | | | | | | |
| 14 | | | | | | | | | | | |

Figure D.8: Log Out



| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | SLICE | | | | | | | | | | |
| 2 | Log Out | | | | | | | | | | |
| 3 | Allows Users to Log out | | | | | | | | | | |
| 4 | | | | | | | | | | | |
| 5 | | Test Case | | | | | | | | | |
| 6 | Pass/Fail Status | Test Number | Description | Action to perform test (input) | Steps to be Executed | State Before Test | Expected result | Observed result | Comments | Tested By | Test Date |
| 7 | | | | | | | | | | | |
| 8 | F | 8.001 | Log Out of Slice | Log Out button | Click Log Out Button | None | To have no access to Slice but only the ability to log back in | | | | |
| 9 | | | | | | | | | | | |
| 10 | #REF! | nit Summary | | 0% | passing | 0 | passed | | Date of last test = | | |
| 11 | | 1 | tests | | | 1 | failed | | | | |
| 12 | | | | | | | | | | | |
| 13 | | Directory Page | | | | | | | | | |