**Feasibility Study and Project Plan**

**Online Judge App**

**Version 4**

**Prepared by**

**Mohammed Abdullah Albukhari**

**Instructor:**Dr. Masoud Sadjadi

**Course:** CIS 4911

**Date:** September 2014

The MIT License (MIT)

Copyright (c) 2014 Florida International University

Permission is hereby granted, free of charge, to any person obtaining a copy  
of this software and associated documentation files (the "Software"), to deal  
in the Software without restriction, including without limitation the rights  
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
copies of the Software, and to permit persons to whom the Software is  
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in  
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,  
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
THE SOFTWARE.

Abstract

To optimize the evaluation process that takes place during the final rally of high-level projects, The Mobile Judge App has been created. Over a period of a few hours, the student will be judged by several judges. The classification process of various projects seems very inefficient, and judges Online App attempt to make the process easier to judge.

The three previous versions of the application have exceptional work performance of judges and the requirements of the students. However, since the application was used by the department of computer science. Based on user experience and current features some new functionalities have been requested. We strive to fulfill this gap by extending the current system, the desired functionality.

The new version will be adding new requested feature into the current system, as well as it will also fix some existing bugs those have been identified during user’s evaluation phase. This system will use the same kind of technologies that has been used before and will also add some new libraries wherever necessary.

Contents

[1. Introduction 5](#_Toc406066063)

[1.1 Problem Definition 5](#_Toc406066064)

[1.2 Background 5](#_Toc406066065)

[1.3 Definitions, Acronyms, and Abbreviations 6](#_Toc406066066)

[1.4 Overview of Document 6](#_Toc406066067)

[2. Feasibility Study 7](#_Toc406066068)

[2.1 Limitations and Constraints of the Current System 7](#_Toc406066069)

[2.2 Function of the New System 7](#_Toc406066070)

[2.3 High-level Definition of User Requirements 8](#_Toc406066071)

[2.4 Alternative Solutions 8](#_Toc406066072)

[2.4.1 Description of Alternatives 8](#_Toc406066073)

[2.4.2 Selection Criteria 9](#_Toc406066074)

[2.4.3 Analysis of Alternatives (refer to Appendix C – Feasibility Matrix) 9](#_Toc406066075)

[2.5 Recommendations 9](#_Toc406066076)

[3. Project Plan 10](#_Toc406066077)

[3.1 Project Organization 10](#_Toc406066078)

[3.1.1 Project Personnel Organization 10](#_Toc406066079)

[3.1.2 Hardware and Software Resources 10](#_Toc406066080)

[3.2 Work breakdown of Tasks, Milestones and Deliverables 12](#_Toc406066081)

[4.0 Proposed System Requirements 14](#_Toc406066082)

[4.1 Functional Requirements 14](#_Toc406066083)

[4.2 Analysis of System Requirements 27](#_Toc406066084)

[4.2.1 Use case model 27](#_Toc406066085)

[30](#_Toc406066086)

[4.2.2. Static model 31](#_Toc406066087)

[4.2.3. Dynamic Model 31](#_Toc406066088)

[5. Application Design 34](#_Toc406066089)

[5.1 Introduction 34](#_Toc406066090)

[5.2 System Components 34](#_Toc406066091)

[5.3 Whole System Integration Design 36](#_Toc406066092)

[5.4 Database Design and Management 37](#_Toc406066093)

[5.5 System Vulnerability and Security 37](#_Toc406066094)

[5.6 New Implementation Update 38](#_Toc406066095)

[4. Appendix 39](#_Toc406066096)

[4.1 Appendix A – Project Schedule (Agile Approach) 39](#_Toc406066097)

# Introduction

## Problem Definition

It is always a difficult task for a professors and Instructor to grade exam or projects. If to the traditional burdens we add a live environment where multiple students have to be graded simultaneously in a timely manner then grading a senior project showcase is at minimum challenging that is why the Mobile Judge App was designed; to ease the judges’ task while keeping the excitement.

The first three versions of the Online Judge App accomplished this task beautifully but once the app has been used, it generates data, information that could be used but there’s no functionality to access it yet.

Version 4 of the Mobile Judge App focuses on the enhancement of the administrative aspects of the app. A new email module will be added to allow the creation and management of email template along with the capability to do bulk notifications that will access the historical data available in the application.

Administrators will now be able to accept or reject multiple grades simultaneously from a single screen and to get live statistics regarding the state of the grading by the judges. The current application platform will be expanded to support multiple roles for the same user and it will provide an interface for that user to switch between his/her roles.

## Background

The previous version of the app used Sencha Touch as the main framework for development of the mobile app. This approach allows targeting multiple mobile platforms with the same code base while losing some functionalities available in the traditional native apps. However the tradeoffs are worth it. The app is data driven for most use cases and the framework provides most of the desired functionalities so for this version we will continue to use **Sencha Touch** as the main development framework.

## Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| Task | A piece of job that serves as a unit of work. |
| App | A mobile application that can be installed on smartphones. |
| Alternative | A possible manner by which a given problem may be resolved. |
| PERT chart | Program Evaluation and Review Technique. A tool that is used to model the tasks and schedules associated with the completion of a project. |
| Feasibility study | Evaluating the best way for resolving a problem. |
| Sencha Touch | High-performance HTML5 mobile application framework |
| iOS | The mobile operating system offered by Apple |
| Android | The mobile operating system offered by Google |
| UML | Unified Modeling Language |
| DBMS | Relational Database Manage System |
| SQL | Structured Query Language |
| JS | Javascript Language |
| CSS | Cascading Style Sheet |

## Overview of Document

This document provides information about the feasibility study carried out before implementing the project, followed by the manner in which the project will be organized.

* Chapter 2 contains all the details associated with the feasibility study undertaken by the group.
* Chapter 3 contains the details regarding the project plan.
* Chapter 4 contains various appendices that aid in the understanding of this document.
* Chapter 5 contains references to the websites that were used in order to complete this document.

# Feasibility Study

## Limitations and Constraints of the Current System

The current version of the Online Judge app accomplishes the objectives of that version scope, but in general lacks functionality that the administrators using the app today would like to have in order to ease their jobs. In version 3 there is no mechanism to send emails to students or judges, only one role is linked to any given account, and administrators have no real-time visibility of the grades during the showcase.

## Function of the New System

* The purpose of the new system, Mobile Judge Version 4, is to extend the previous version and provide the desired functionalities to the app administrators.
* A brand new multiple email sending module will be added that will allow filtering the historical data by students, judges, semesters and state of the judges’ responses and performing bulk email notifications.
* Keep historical record of email send successfully.
* Add or remove specific users from within application with chosen role
* Users will be able have multiple roles and switch amongst them. An administrator will have real-time feedback of the grades during the showcase with filter and graph capabilities.
* Implement a "view as" feature where the admin can view as if he was the student or judge he selects
* Add a new graph/chart which shows the average class question grade. A bar chart which has class average for Question 1, Question 2, etc
* A a new feature which will alert the administrator if there is a grade submitted which is bellow a threshold (example a judge gave a student a 10/50)
* It is also going to have a brand new google authentication system as previous version has some problem with that.

## High-level Definition of User Requirements

* The user requirements for this system are very minimal. The user would be required to have either a smartphone or a tablet in order to use the application or any other device with web access where the user can access the application online.
* Second set of requirements is for each student is judge fairly and equally by using the pre-specified rubric.
* In order to assure privacy each student will only be able to view their scores after being graded, and each judge will be able to see only the student that they are assigned to.
* For security purposes both judges and students will need to register in order to have access to the application and the data available.
* The registration process will be standardized requesting basic information such as first name, last name, username, password and email address. The information gather at registration will be used for authentication when accessing one user’s specific information i.e. grades for a student.
* Fixed student’s login using Gmail (FIU email) authentication system.

## Alternative Solutions

### Description of Alternatives

* In alternative 1 to keep the Sencha Framework as the front-end interface and replace the backend implementation with a new one that will provide the desired functionalities.
* In alternative 2 to keep all existing frameworks and expand on them to integrate the new functionalities to the existing code base.

### Selection Criteria

The feasibility criteria that would be used consist of four categories: operational, technical, economic and schedule.

Operating feasibility includes functionality, meaning to what extent the admin would benefit from the email capabilities and how well the system would work. Technical feasibility includes the technology involved in the development of the functionality and the expertise in such technology required from the developer. Economic feasibility is the cost associated with the project. Lastly, the schedule feasibility is how long the system would take to be designed and implemented.

### Analysis of Alternatives (refer to Appendix C – Feasibility Matrix)

* Alternative 1 is less expensive than alternative 1, but given the close ties between the database schema and the User Interface on the current version, replacing the database schema would require to rewrite a great percentage of the existing functionalities.
* Alternative 2 uses as much of the existing code base as possible. This close relation between the old and new code can bring some complexities, however it allows for an incremental development where the system evolves slowly between stable states.

## Recommendations

The recommendation for this project is alternative 2. It provides the best compromise to get the desired functionalities without affecting or having little risk of affecting the existing ones. By reusing existing code and schemas, the design prioritizes the new functionalities over refactoring existing code.

# Project Plan

## Project Organization

The project has been divided into five different sections: the Email section, the Roles section, the View as section, the Grade Monitor section and the Stats section. Each section will be done one by one by the only developer will be in charge to develop the components needed in each tier to support the new functionality. In this way the one section would be responsible for the database changes, PHP API additions, and for the Sencha views and stores in the client code.

### Project Personnel Organization

There will be one people working on the project. So all the task will be done one by one based on priority.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Position | Periods required | Key Phases |
| Mohammed Abdullah Albukhari | Website/Mobile Developer | 08/25/14 to 12/12/14 | All |

### Hardware and Software Resources

In order to successfully complete this project the proper hardware and software are needed. Developer member will be using his own personal laptop, having one being a PC with windows and another using a MacBook. Each computer will be required to have a copy of Sencha Architect and any database software that supports SQL.

Hardware resources:

* 1 local server
* 1 development server
* 1 integration server
* 1 Android mobile phone
* 1 iOS mobile phone
* 1 Windows phone

Software resources:

* Sencha Architect, Sencha Cmd
* IIS or Apache
* MYSQL
* SourceTree
* Git Server

## Work breakdown of Tasks, Milestones and Deliverables

This semester the group used the agile methodology. This approach is an incremental software development technique that divides the project’s time into sprints where small features have to be developed, tested and showcased before the moving to the next sprint. The chart schedule can be seen in Appendix A.

Sprint 1:

* Requirements Definition
* Developers Roles
* Initial Mockups Design
* Environment Setup

Sprint 2:

* Multiple Email Static View
* Statistics Graph/Chart Static View
* Login Sequence Modification with Gmail
* View as for Admin Static View
* Automatic Grade Check Static View
* Unit Testing

Sprint 3:

* Multiple Email PHP API Development
* Statistics Graph/Chart View
* Statistics PHP API
* Automatic Grade Check PHP API
* Unit Testing

Sprint 4:

* Multiple Email Template Database
* Statistics Judge Graph/Chart
* View as for Admin PHP API
* Feature Integration
* Unit Testing

Sprint 5:

* Multiple Email Feature Final
* View As as Admin Feature Final
* Graph/ Chart Final
* Automatic Grade check Final
* Unit Testing Integration Testing
* Deployment

# 4.0 Proposed System Requirements

This chapter of the document will cover the new version of this project will have. This chapter is divided into two main sections. Section 4.1 covers the functional requirements; in this section the high-level functionalities are described. Section 4.2 covers the analysis of the system requirements; in this section use case models, static models and dynamic models are used to help with the analysis of the system requirements.

## 4.1 Functional Requirements

***FR-1:***

-The system should allow only administrators to assign judges to students. This needs to be in a random fashion with no duplication of judges.

*Usability:* Function should be easy to perform and should provide feedback when done.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance:* The assigning of judges to students should not take longer than 10 seconds.

*Supportability*: This function should be provided for the present version and future versions of all major internet browsers. It should also work on any IOS provided.

***FR-2:***

-The system should allow only the admin to add questions.

*Usability:* Function should be easy to perform.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period. .

*Performance:* Adding questions to the database should not taking longer than 15 seconds.

*Supportability*: This function should be provided for the present version and future versions of all major internet browsers. It should also work on any IOS provided.

***FR-3:***

-The system should allow only the admin to set the rules for distribution of judges.

*Usability:* Function should be easy to perform.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance:* Setting a judge limit should not taking longer than 5 seconds.

*Supportability*: This function should be provided for the present version and future versions of all major internet browsers. It should also work on any IOS provided.

***FR-4:***

-The system should allow students to view their scores/grades.

*Usability:* The function should be easy to do and clearly shown on their profile page.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance:* Viewing a score should not take longer than 5 seconds.

*Supportability*: This function should be provided for the present version and future versions of all major internet browsers. It should also work on any IOS provided.

***FR-5:***

-The system should allow the judges to view and score only the students that they are assigned to judge.

*Usability:* The function should be easy and intuitive.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance:* The ability to view and submit a score should not take longer than 5 seconds.

*Supportability*: This function should be provided for the present version and future versions of all major internet browsers. It should also work on any IOS provided.

***FR-6:***

-The system should allow a student to register. This registration should include an email and password.

*Usability:* The function should be quick, easy and intuitive.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance:* The ability to submit a registration should not take longer than 5 seconds.

*Supportability*: This function should be provided for the present version and future versions of all major internet browsers. It should also work on any IOS provided.

The functional requirements numbers from FR-7 to FR-12 are new functionalities that will be implemented for Version 2 of this project. These functionalities were not included in the previous version.

***FR-7:***

-The system should allow the admin to send invitations to judges.

*Usability:* The function should be easy and intuitive.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance:* The ability to send an invitation should not take longer than 5 seconds.

*Supportability*: This function should be provided for the present version and future versions of all major internet browsers. It should also work on any IOS provided.\

***FR-8:***

-The system should allow the admin to set the location of the student.

*Usability:* The function should be easy and intuitive.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance:* The ability to submit a student location should not take longer than 5 seconds.

*Supportability*: This function should be provided for the present version and future versions of all major internet browsers. It should also work on any IOS provided.

***FR-9:***

-The system should allow a judge to identify students that can’t be graded by him/her.

*Usability:* The function should be easy and intuitive.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance:* The ability to identify a student should not take longer than 5 seconds.

*Supportability*: This function should be provided for the present version and future versions of all major internet browsers. It should also work on any IOS provided.

***FR-10:***

-The system should keep track of the participation of judges in previous semesters.

*Usability:* The function is automated.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance:* The ability to show history for a judge should not take longer than 15 seconds.

*Supportability*: This function should be provided for the present version and future versions of all major internet browsers. It should also work on any IOS provided.

***FR-11:***

-The system should allow the administrator add/delete a judge at any time after assignment is done.

*Usability:* The function should be easy and intuitive.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance:* The ability to add/delete a judge should not take longer than 5 seconds.

*Supportability*: This function should be provided for the present version and future versions of all major internet browsers. It should also work on any IOS provided.

***FR-12:***

-The system should allow the admin to accept or reject each judge evaluation.

*Usability:* The function should be easy and intuitive.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance:* The ability to accept/reject an evaluation should not take longer than 5 seconds.

*Supportability*: This function should be provided for the present version and future versions of all major internet browsers. It should also work on any IOS provided.

***FR-13:***

-The system should allow the user to change between his/her roles.

*Usability:* The function should be easy and intuitive.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance:* The ability to change roles should not take longer than 5 seconds.

*Supportability*: This function should be provided for the present version and future versions of all major internet browsers. It should also work on any IOS provided.

***FR-14:***

-The system should allow the admin to add or remove roles for all users including himself/herself.

*Usability:* The function should be easy and intuitive.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance:* The ability to add/remove roles should not take longer than 5 seconds. Except when sending email in which case it should not take longer than 10 seconds.

*Supportability*: This function should be provided for the present version and future versions of all major internet browsers. It should also work on any IOS provided.

***FR-15:***

-The system should allow the admin to change his/her default role

*Usability:* The function should be easy and intuitive.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance:* The ability to change a default role should not take longer than 5 seconds.

*Supportability*: This function should be provided for the present version and future versions of all major internet browsers. It should also work on any IOS provided.

***FR-16:***

-The system should add the appropriate role when new users are added to the system

*Usability:* The function should be automated

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance:* The adding of appropriate roles should not take longer than 15 seconds for a mass import and 5 seconds for single instance.

*Supportability*: This function should be provided for the present version and future versions of all major internet browsers. It should also work on any IOS provided.

***FR-17:***

-The system should allow a user to login

*Usability:* The function should be easy and intuitive. The logging in should be automated.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance:* The ability to login should not take longer than 5 seconds.

*Supportability*: This function should be provided for the present version and future versions of all major internet browsers. It should also work on any IOS provided.

***FR-18:***

-The System should allow administrator to view average grades in a list as they are being submitted.

*Usability:* The function should be easy and intuitive.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance:* The ability to view average grades in a list should not take longer than 5 seconds.

*Supportability*: This function should be provided for the present version and future versions of all major internet browsers. It should also work on any IOS provided.

***FR-19:***

-The System should allow administrator to view average in a graph as they are being submitted.

*Usability:* The function should be easy and intuitive.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance:* The ability to view average grades in a graph should not take longer than 5 seconds.

*Supportability*: This function should be provided for the present version and future versions of all major internet browsers. It should also work on any IOS provided.

***FR-20:***

-The System should allow administrator to change the grade selection for the list and graph based on students or projects.

*Usability:* The function should be easy and intuitive.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance:* The ability to change a listing method for the list and graph should not take longer than 5 seconds.

*Supportability*: This function should be provided for the present version and future versions of all major internet browsers. It should also work on any IOS provided.

***FR-21:***

-The System should allow the administrator to select an element in the list or graph and view details of selected element.

*Usability:* The function should be easy and intuitive.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance:* The ability select an element from the list or graph should not take longer than 5 seconds.

*Supportability*: This function should be provided for the present version and future versions of all major internet browsers. It should also work on any IOS provided.

***FR-22:***

-The System should allow administrator to accept or decline grades submitted by judges within the same list.

*Usability:* The function should be easy and intuitive.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance:* The ability to accept or decline grades should not take longer than 5 seconds.

*Supportability*: This function should be provided for the present version and future versions of all major internet browsers. It should also work on any IOS provided.

***FR-23:***

-The System should allow judges to view a controlled version of the grades submitted by all judges.

*Usability:* The function should be easy and intuitive.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance:* The ability to view controlled version of the grades should not take longer than 5 seconds.

*Supportability*: This function should be provided for the present version and future versions of all major internet browsers. It should also work on any IOS provided.

***FR-24:***

-The System should allow administrator to change the refresh rate of the stats page.

*Usability:* The function should be easy and intuitive.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance:* The ability to change auto-update rate should not take longer than 5 seconds.

*Supportability*: This function should be provided for the present version and future versions of all major internet browsers. It should also work on any IOS provided.

***FR-25:***

-The system should allow admin to send emails to current students.

*Usability*: The system should be easy and intuitive to use and provide feedback when it is finished.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance*: Sending the email should not take more than 10 seconds.

*Supportability*: This function should be work for any browser.

***FR-26:***

-The system should allow admin to send email to students from past semesters

*Usability*: The system should be easy and provide access to previous semesters to use and provide feedback when it is finished.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance*: Sending the email should not take more than 10 seconds.

*Supportability*: This function should be work for any browser.

***FR-27:***

-The system should allow admin to send emails to current judges

*Usability*: The system should be easy to use and provide feedback when it is finished.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance*: Sending the email should not take more than 10 seconds.

*Supportability*: This function should be work for any browser.

***FR-28:***

-The system should allow admin to send emails to pending judges.

*Usability*: The system should be easy to use and provide feedback when it is finished.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance*: Sending the email should not take more than 10 seconds.

*Supportability*: This function should be work for any browser.

***FR-29:***

-The system should allow admin to send email to declined judges

*Usability*: The system should be easy to use and provide feedback when it is finished.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance*: Sending the email should not take more than 10 seconds.

*Supportability*: This function should be work for any browser.

***FR-30:***

-The system should allow admin to send emails to judges from past semesters

*Usability*: The system should be easy to use and provide feedback when it is finished.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance*: Sending the email should not take more than 10 seconds.

*Supportability*: This function should be work for any browser.

***FR-31:***

-The system should allow admin to send emails to pending judges from past semesters

*Usability*: The system should be easy to use and provide feedback when it is finished.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance*: Sending the email should not take more than 10 seconds.

*Supportability*: This function should be work for any browser.

***FR-32:***

-The system should allow admin to send email to declined judges from past semesters

*Usability*: The system should be easy to use and provide feedback when it is finished.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance*: Sending the email should not take more than 10 seconds.

*Supportability*: This function should be work for any browser.

***FR-33:***

-The system should allow admin to create email templates5

*Usability*: The system should be easy to use and provide feedback when it is finished.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance*: Saving a template should not take more than 3 seconds.

*Supportability*: This function should be work for any browser.

***FR-34:***

-The system should allow admin to insert fields in the emails template that would be populated at send time.

*Usability*: The system should be easy to use and provide feedback when it is finished.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance*: Inserting a field should not take more than 1 second.

*Supportability*: This function should be work for any browser.

***FR-35:***

-The system should allow admin to edit any existing email template

*Usability*: The system should be easy to use and provide feedback when it is finished.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance*: Saving an edited template.

*Supportability*: This function should be work for any browser.

***FR-36(New):***

-The system should provide the admin with a filter that will only allow to send email to judges

*Usability*: The system should be easy to use and provide feedback when it is finished.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

*Performance*: Filter should return result in less than a second.

*Supportability*: This function should be work for any browser.

***FR-37(New):***

-The system should provide the admin a way to send email automatically to many judges

*Usability*: The system should be easy to use and provide feedback when it is finished.

*Reliability:* The probability of failure is 1% and 3% over a 48hrs period.

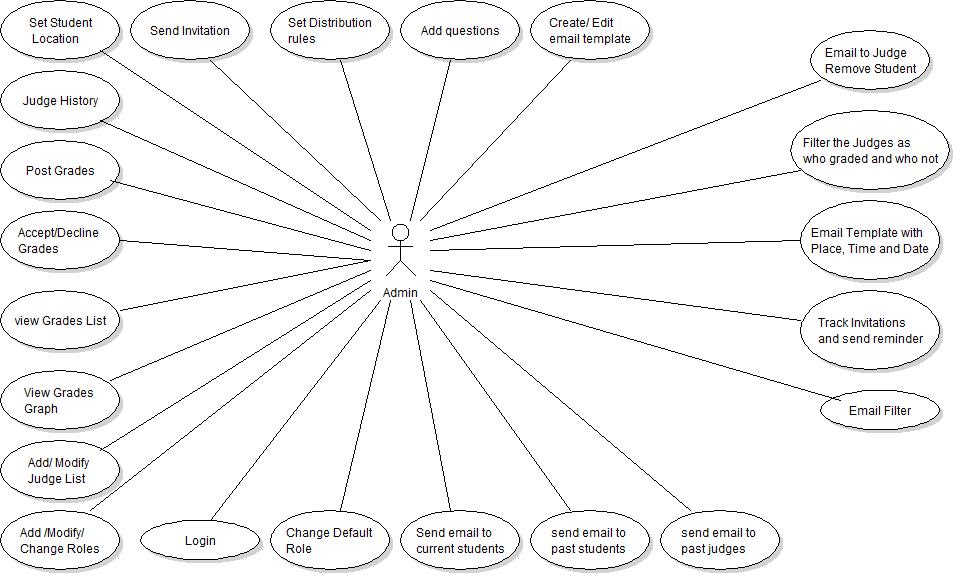
*Performance*: Filter should return result in less than a second.

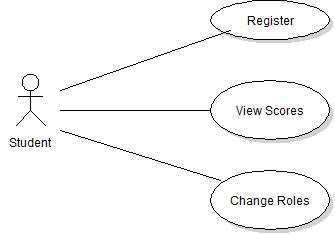
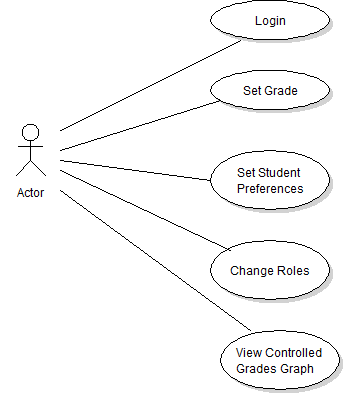
*Supportability*: This function should be work for any browser.

## 4.2 Analysis of System Requirements

### 4.2.1 Use case model

A use case by definition is a list of steps expressing the interactions between the system and the people identified as actors that use the system. It clearly communicates what the system will need in order to meet the requirements. The use case model shows all use cases and their interacting users in a high level view. In the system that is to be developed expresses a solution for the need to judge students on their senior projects. The model shown in Appendix B is a use case Diagram and it shows all functions that belong to the actors in order to reach the requirements placed on the system. For this project we have a total of three actors and a total of 11 use cases that will be implemented. The use cases are divided among the actors as followed:





Actor1: Admin

* Add Questions
* Set Distribution Rules
* Send Invitation
* Set Student Location
* Judge History
* Post Grades
* Accept Grades
* Decline Grades
* View Grades List
* View Grades Graph
* Modify Judge List
* Change Roles
* Login
* Add Roles
* Remove Roles
* Change Default Role
* Send email to current students
* Send email to past students
* Send email to current judges
* Send email to past judges
* Create email template
* Edit email template
* Send Email to multiple judges

Actor2: Judge

* Set Grade
* Set Student Preferences
* Change Roles
* View Controlled Grades Graph

### 

### 

Actor3: Student

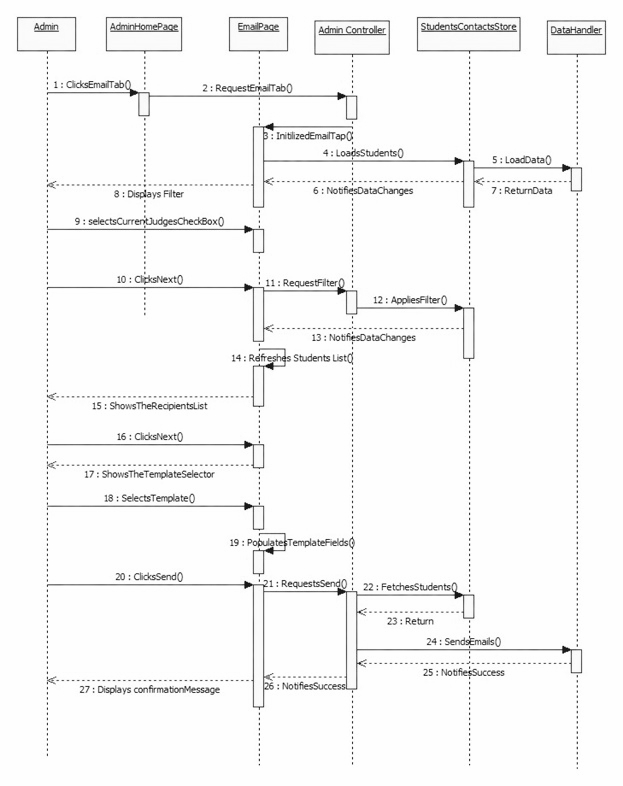
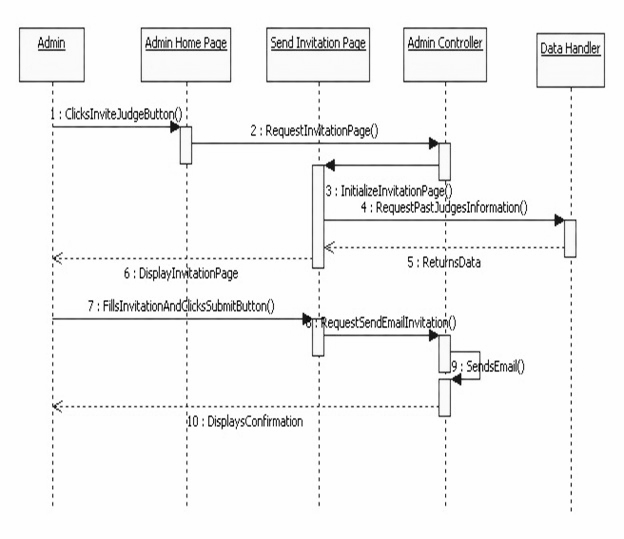
* Register
* View Scores
* Change Roles

### 4.2.2. Static model

A static model expresses the system and does not account for time or the sequence of events. This kind of model includes class diagram and object diagram. A class diagram displays the structure of system by showing the classes, the attributes, methods, and the relationship between these classes. While an object diagram focuses on a particular set of objects with their attributes and the links between them. These diagrams are shown in Appendix. In this project we will only have three main classes, an Admin class, a judge class and a student class. Please refer to Appendix for more details on the class and object diagrams for this project.

### 4.2.3. Dynamic Model

The dynamic model however does take into account the element of time. This kind of model includes sequence diagrams and state machines. A sequence diagram shows object and class interactions in a sequence of events arranged in a time line that displays functionality. Another type of dynamic model is a state machine, which displays an object of class and the tracing of this object throughout the system. These types of diagrams display transitioning of events for an object in the system and allow the developers and programmers to view how the users should transition given their actions. These types of diagrams are displayed in Appendix.

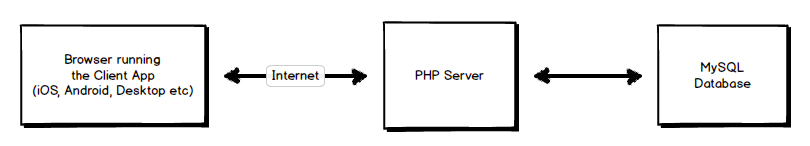


# 5. Application Design

In this chapter we are covering our system and subsystem design. This chapter is made out of the following sections. Section 5.1: Overview, this section provides a high-level description of the system design architecture. Section 5.2: Subsystem Decomposition, this section will provide a detailed description for each of the major subsystems. Section 5.3: Hardware and Software Mapping, in this section we will cover how the subsystems are mapped to the hardware and software. Section 5.4: Persistent Data Management, this section will identify the data that needs to be stored and the structure of the data. Section 5.5: Security and Privacy, this section describes the user authentication processes, encryption of data and all other security parameters being implemented.

## 5.1 Introduction

our design because it allows for a thin client, easier scalability and a more secured database. However, our system design also has an aspect of client server architecture. This is because both our business logic and our database are running on the same server. The client subsystem is an HTML5 application made with the Sencha Touch framework. A PHP server composes the server subsystem that provides all the data that is being handled through the client application. This server provides a way to connect via the Internet to the database. The second subsystem is where all the information is being stored. A MySQL database takes this role. Below is a basic illustration showing how the client server architecture works and also identifying the different tiers of the 3 tier architecture.



Tier 3: Database Tier

Tier 2: Business Logic Tier

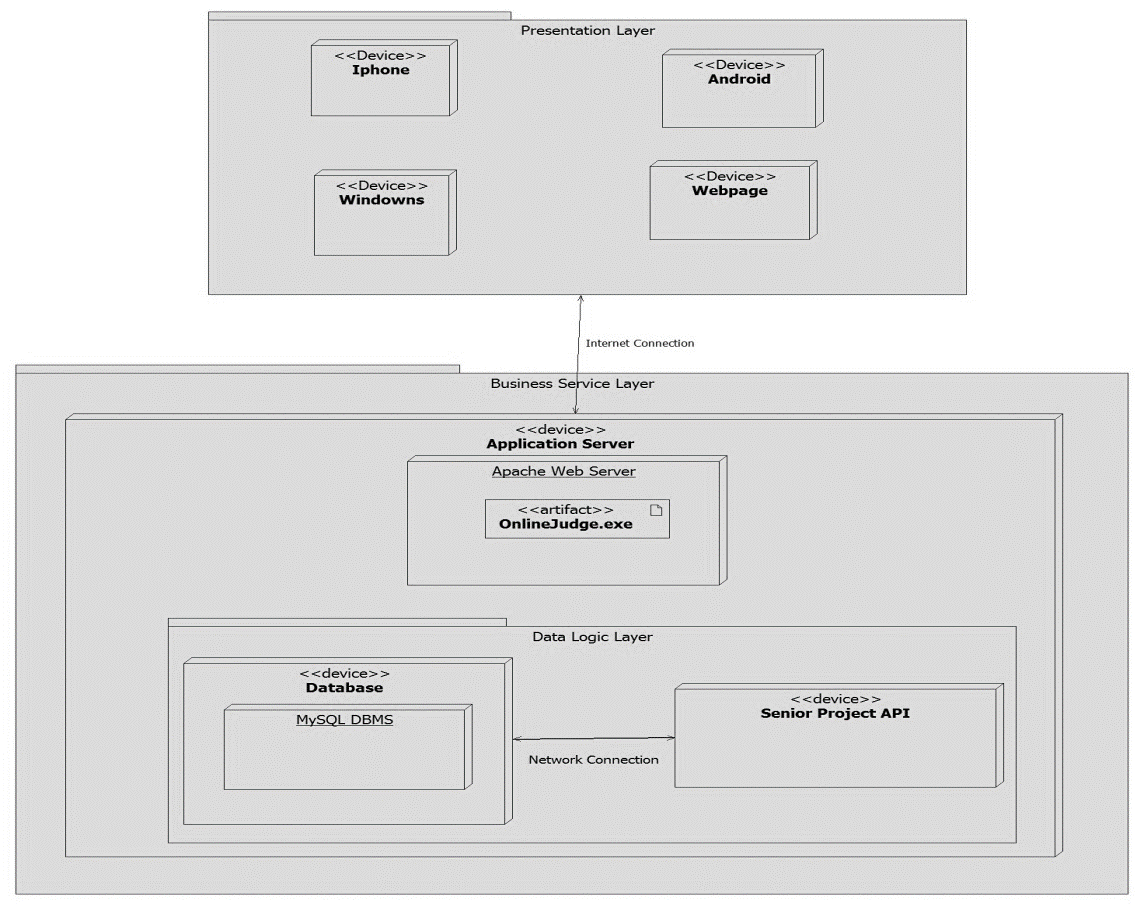
Tier 1: Client Tier

## 5.2 System Components

For this version we have identified 3 subsystems. The first one is roles subsystem. This subsystem will manage what interface is delivered to the user based on the role(s) the user has assigned. This subsystem will provide functionality for a user to switch between his/her roles. It also allows the administrator to manage the roles assigned to each user. The second subsystem is the Live Statistics. This subsystem focuses on the showcase event by providing a real-time feed of the grades submitted by judges. It also provides graphing functionalities for the administrator to capture the status of the whole class during the event. The third and final subsystem is the emails subsystem. This subsystem will allow the administrator to email students and judges while providing advanced filtering capabilities. It also allows to create and manage multiple email templates to be used to generate the emails’ bodies.

## 5.3 Whole System Integration Design

Our hardware and software mapping is illustrated below by our deployment diagram.



## 5.4 Database Design and Management

For this version of the Mobile Judge Application, very little changes made to the database Schema. Only a table was added to store the email template information and 2 fields were added to support the roles functionality.

We were able to provide all this version’s functionalities by querying the existing schema.

## database.png5.5 System Vulnerability and Security

All data is password protected in the Mobile Judge App. Passwords will be encrypted while being stored in the database using a hash algorithm so there won’t be a way to get them back in case of breach. Judges will only have access to their assignments. Students will only have access their own data, and grade. Only the admin will have access to complete results. Authentication of each student user is done by checking against the Senior Project Website project API. Authentication of the admin and judges will be done within the database for the existence of the users email and password in the Users table.

## 5.6 New Implementation Update

The email subsystem main functionality is set on the filter and send multiple emails from previously available judges in the system multiple at a time. As the users interacts with the filters by enabling checkboxes and selecting items in the lists, flags are enabled for each interaction. At the time of displaying the recipients list the setJudgeFilters() functions build another function that is then passed to the store for judges and students respectively. This function will be using the flags previously enabled and state each record in the store to determine if that record has to be included in the filter. When the user deselects either judge from the list in the recipient lists step, the sent flag for that record is disabled. Finally on the send step the subsystem fetches the filtered data for students and judges and builds the email bodies by replacing the fields on the email template with the record values. Also any record with the sent flag disabled is ignored. A simple loop iterates over the built emails and calls the PHP API to send the email.

The Live Stats subsystem makes extensive use of the Sencha Touch’s built in data binding features to populate the list, rendered graph, and apply the corresponding grouping. The popup that appear when a user selects an element in the list or graph call on the the itemtap() function. This function will map the element chosen with the corresponding record from the store and process the record for the adequate information to display in the popup.

# 6. System Validation

This chapter coves our system validation, meaning the testing process of the system. This system is divided into two sections. Section 7.1 Subsystem Tests, covering testing done for each subsystem individual prior to integration. Section 7.2 System Tests, covering all the system tests conducted after integrating the roles, emails, and stats subsystems.

## 6.1 Subsystem Tests

**Roles** conducted by Fernando Dos Santos

|  |  |
| --- | --- |
| **Subsystem** | **Test Description** |
| Change Roles | This tests that all the users can change between their roles. This will only validate that the changing of roles is correct and that the only roles users can change to are those available to them. |
| Add/Remove/Default Roles | This tests that roles cane be added, removed, or set as default by the admin only. This should all be done from the application, not database. |
| Set Roles when User Added | This tests that the appropriate roles are added when new users are imported or added such as new judges or students. |

**Emails** conducted by Andy Norcisa:

|  |  |
| --- | --- |
| **Subsystem** | **Test Description** |
| Filtering | This tests that all the recipients are correctly chosen from the filters that are set. It also tests that specific recipients can later be removed or added. |
| Templates | This tests that new templates can be created or edited and that they can be previewed. |
| Sending Emails | This tests that emails are sent to the appropriate recipients with the chosen template. |

**Live Stats** conducted by Eric Jo:

|  |  |
| --- | --- |
| **Subsystem** | **Test Description** |
| Admin Viewing Lists | This tests that the admin can accurately see the grade data in real-time at the appropriate intervals for both students and projects in a list view. |
| Admin Viewing Graph | This tests that the admin can accurately see the raw and accepted grade data in real-time at the appropriate intervals for both students and projects in a graph view. |
| Judge Viewing Graph | This tests that a judge can accurately see the submitted grades for their students in real-time at the predefined interval with the right threshold of students. |

## 6.2 System Tests

***Change Roles from admin to judge/student***

**Valid-Input-Test** –

*PreCond:* admin has judge/student role and clicks the right button.

*Result:* Admin presented with judge or student screen. -- PASS

**Invalid-Input-Test**–

*PreCond:* admin does not have more roles.

*Result:* Admin does not see more role buttons. – PASS

***Change Roles from judge to student/admin***

**Valid-Input-Test** –

*PreCond:* judge has admin/student role and clicks the right button.

*Result:* Judge presented with admin or student screen. -- PASS

**Invalid-Input-Test**–

*PreCond:* judge does not have more roles.

*Result:* Judge does not see more role buttons. – PASS

***Change Roles from student to judge/admin***

**Valid-Input-Test** –

*PreCond:* student has admin/judge role and clicks the right button.

*Result:* Student presented with admin or judge screen. -- PASS

**Invalid-Input-Test**–

*PreCond:* student does not have more roles.

*Result:* Student does not see more role buttons. – PASS

***Admin change default role to judge/student***

**Valid-Input-Test** –

*PreCond:* admin in settings page and has judge/student roles. Admin selects default role.

*Result:* Default role changed. Next login will show default role page -- PASS

**Invalid-Input-Test**–

*PreCond:* admin does not have more roles.

*Result:* admin does not see more default role options. – PASS

***Add Roles to admin/self***

**Valid-Input-Test** –

*PreCond:* admin in settings page and clicks roles button. Admin adds desired role

*Result:* Role is added, admin can switch to that role via roles button.. -- PASS

**Invalid-Input-Test**–

*PreCond:* admin tries to add a role he/she already has.

*Result:* Role is removed when unchecked and confirm. – PASS

***Remove Roles from admin/self***

**Valid-Input-Test** –

*PreCond:* admin in settings page and clicks roles button. Admin removes desired role

*Result:* Role is remove, admin does not see role in roles button popup.. -- PASS

**Invalid-Input-Test**–

*PreCond:* admin tries to remove a role he/she doesn’t have.

*Result:* Role is added when checked and confirm. – PASS

***Add Roles to judge***

**Valid-Input-Test** –

*PreCond:* admin in people page->judge tab. Admin clicks judge. Admin adds desired role

*Result:* Role is added to judge. Judge can switch between new role(s). -- PASS

**Invalid-Input-Test**–

*PreCond:* admin tries to add a role the judge already has.

*Result:* Role is removed when unchecked and confirm. – PASS

***Remove Roles from judge***

**Valid-Input-Test** –

*PreCond:* admin in people page->judge tab. Admin clicks judge. Admin removes desired role

*Result:* Role is removed from judge. Judge cannot see the role removed in roles popup. -- PASS

**Invalid-Input-Test**–

*PreCond:* admin tries to remove a role the judge doesn’t have.

*Result:* Role is added when checked and confirm. – PASS

***Add Roles to student***

**Valid-Input-Test** –

*PreCond:* admin in desired student page. Admin clicks roles button and adds admin role

*Result:* Role is added to student. Student can switch between new role(s). – PASS

**Valid-Input-Test** –

*PreCond:* admin in desired student page. Admin clicks roles button and adds judge role

*Result:* Email is sent to student. -- PASS

**Invalid-Input-Test**–

*PreCond:* admin tries to add a role the student already has.

*Result:* Role is removed when unchecked and confirm. – PASS

***Remove Roles from student***

**Valid-Input-Test** –

*PreCond:* admin in desired student page. Admin clicks roles button and removes the desired role.

*Result:* Role is removed from student and cannot see the role removed in roles popup. -- PASS

**Invalid-Input-Test**–

*PreCond:* admin tries to remove a role the student doesn’t have.

*Result:* Role is added when checked and confirm. – PASS

***Reset App***

**Valid-Input-Test** –

*PreCond:* admin in settings page. Admin selects term and clicks save. Admin clicks reset app.

*Result:* All users removed except admin, and put in history table under specified term. -- PASS

***Import students***

**Valid-Input-Test** –

*PreCond:* admin is on people page->students. The Senior Project Website is correctly configured and the URL is correct. Admin clicks the Load button.

*Result:* New students are imported into database and given all their appropriate roles and attributes. -- PASS

**Invalid-Input-Test**–

*PreCond:* Senior project website url is wrong.

*Result:* No students are added. – PASS

**Invalid-Input-Test**–

*PreCond:* Senior project website has incorrect data or function is wrong

*Result:* No students are added or students are added with incorrect data. – PASS

***Student accepts judge invite and registers***

**Valid-Input-Test** –

*PreCond:* student received invite and accepted. Student registers

*Result:* Judge Role is added to student. Judge (prev student) is automatically assigned students to judge/grade. -- PASS

**Invalid-Input-Test**–

*PreCond:* student declines invite.

*Result:* Judge Role is not added. Invite listed as declined. – PASS

**Invalid-Input-Test**–

*PreCond:* student registers with invalid data.

*Result:* student prompted to correct data. Judge role not added until data is valid. – PASS

***New judges accepts invite and registers***

**Valid-Input-Test** –

*PreCond:* user received invite and accepted. User registers

*Result:* New account is created for the user. The judge is assigned students, an id, and given judge role. -- PASS

**Invalid-Input-Test**–

*PreCond:* user declines invite.

*Result:* Invite listed as declined. – PASS

**Invalid-Input-Test**–

*PreCond:* user registers with invalid data.

*Result:* user prompted to correct data. Account not created until data is valid. – PASS

***Admin/Judge login***

**Valid-Input-Test** –

*PreCond:* user inputs valid credentials.

*Result:* System determines default role and loads the corresponding home page. -- PASS

**Invalid-Input-Test**–

*PreCond:* user inputs invalid credentials.

*Result:* System displays invalid prompt. User not logged in. – PASS

***Student login***

**Valid-Input-Test** –

*PreCond:* student inputs valid credentials. Student accepts google conditions on redirect page.

*Result:* System redirects student to google site. Student is displayed with student home page after they have accepted the google terms. -- PASS

**Invalid-Input-Test**–

*PreCond:* student inputs invalid credentials.

*Result:* System displays invalid prompt. User not logged in. – PASS

***Filter and add emails***

**Valid-Input-Test** –

*PreCond:* admin clicks all under students and navigates to next page.

*Result:* student list populated with all students. – PASS

**Valid-Input-Test** –

*PreCond:* admin clicks current under students and navigates to next page.

*Result:* student list populated with the current students. – PASS

**Valid-Input-Test** –

*PreCond:* admin clicks past under students, selects Fall 2013, and navigates to next page.

*Result:* student list populated with past students from Fall 2013 semester. – PASS

**Valid-Input-Test** –

*PreCond:* admin clicks all under judges, selects invited for both and navigates to next page.

*Result:* judge list populated with all judges. – PASS

**Valid-Input-Test** –

*PreCond:* admin clicks current under judges, selects pending and navigates to next page.

*Result:* judge list populated with the current judges whose invite is pending. – PASS

**Valid-Input-Test** –

*PreCond:* admin clicks past under judges, selects declined for Fall 2013, and navigates to next page.

*Result:* judge list populated with past judges from Fall 2013 semester that declined an invite. – PASS

**Valid-Input-Test** –

*PreCond:* admin adds a valid email to the extra emails field and navigates to next page.

*Result:* extra emails list populated with the email from previous page and name if email is in database. – PASS

**Valid-Input-Test** –

*PreCond:* admin adds an invalid email to the extra emails field and navigates to next page.

*Result:* extra emails list populated with the email from previous page and name if email is in database. However the email will be sent to no one and delivery will fail. – PASS

***Create, edit, and select templates***

**Valid-Input-Test** –

*PreCond:* admin clicks new template, adds valid data to the template fields, and clicks save.

*Result:* A new template is created with the data given and is ready for use. – PASS

**Valid-Input-Test** –

*PreCond:* admin clicks an existing template, clicks edit, modifies existing data with valid data, and clicks save.

**Valid-Input-Test** –

*PreCond:* admin clicks an existing template.

*Result:* The fields are populated with the template’s data. – PASS

**Invalid-Input-Test**–

*PreCond:* admin clicks new template, adds invalid data to the template fields, and clicks save.

*Result:* A new template is created with the data given and is ready for use. However, any invalid html data will not be rendered correctly. -- PASS

**Invalid-Input-Test**–

*PreCond:* admin clicks an existing template, clicks edit, modifies existing data with invalid data, and clicks save.

*Result:* The template is updated with the data given and is ready for use. However, any invalid html data will not be rendered correctly. – PASS

***Send email***

**Valid-Input-Test** –

*PreCond:* admin is sending to valid email addresses and clicks send.

*Result:* Emails get sent to previously selected email addresses with the template data. -- PASS

**Invalid-Input-Test**–

*PreCond:* admin is sending to invalid email addresses and clicks send.

*Result:* System attempts to send emails but invalid recipients fail to send. Valid email addresses are still sent. -- PASS

***Change live stats refresh rate***

**Valid-Input-Test** –

*PreCond:* admin clicks + or – button for refresh rate in settings button, then navigates to stats tab

*Result:* Admin will see the stats update at the given interval on the stats page -- PASS

**Invalid-Input-Test**–

*PreCond:* admin is sending to invalid email addresses and clicks send.

*Result:* System attempts to send emails but invalid recipients fail to send. Valid email addresses are still sent. – PASS

***View live stats list and info for students***

**Valid-Input-Test** –

*PreCond:* admin clicks on stats tab while students exist. Admin ensures he/she is in student view on top right button.

*Result:* Admin can see the students’ stats list – PASS

**Valid-Input-Test** –

*PreCond:* admin clicks and holds a student name on the list. Admin ensures he/she is in student view on top right button.

*Result:* Admin can see that student’s detailed info -- PASS

**Invalid-Input-Test**–

*PreCond:* admin clicks on stats tab while students do not exist.

*Result:* Admin sees an empty list – PASS

***View live stats list and info for projects***

**Valid-Input-Test** –

*PreCond:* admin clicks on stats tab while projects exist. Admin ensures he/she is in project view on top right button.

*Result:* Admin can see the projects’ stats list – PASS

**Valid-Input-Test** –

*PreCond:* admin clicks and holds a project name on the list. Admin ensures he/she is in project view on top right button.

*Result:* Admin can see that project’s detailed info -- PASS

**Invalid-Input-Test**–

*PreCond:* admin clicks on stats tab while projects do not exist.

*Result:* Admin sees an empty list – PASS

***View live stats graph and info for students***

**Valid-Input-Test** –

*PreCond:* admin clicks on stats tab and clicks graph button while grades and students exist. Admin ensures he/she is in student view on top right button.

*Result:* Admin can see the students’ stats graph for raw and approved grades – PASS

**Valid-Input-Test** –

*PreCond:* admin clicks and holds a student bar on the list. Admin ensures he/she is in student view on top right button.

*Result:* Admin can see that student’s detailed info -- PASS

**Invalid-Input-Test**–

*PreCond:* admin clicks on stats tab and clicks graph button while students or grades do not exist.

*Result:* Admin sees an empty graph – PASS

***View live stats graph and info for projects***

**Valid-Input-Test** –

*PreCond:* admin clicks on stats tab and clicks graph button while grades and projects exist. Admin ensures he/she is in project view on top right button.

*Result:* Admin can see the projects’ stats graph for raw and approved grades – PASS

**Valid-Input-Test** –

*PreCond:* admin clicks and holds a project bar on the list. Admin ensures he/she is in project view on top right button.

*Result:* Admin can see that project‘s detailed info -- PASS

**Invalid-Input-Test**–

*PreCond:* admin clicks on stats tab and clicks graph button while project or grades do not exist.

*Result:* Admin sees an empty graph – PASS

***Judge view live stats graph and info for students***

**Valid-Input-Test** –

*PreCond:* judge clicks on stats button while more than three grades exist.

*Result:* Judge can see the students’ stats graph for submitted grades – PASS

**Valid-Input-Test** –

*PreCond:* judge clicks and holds a student’s bar on the list.

*Result:* Judge can see that students’s detailed info -- PASS

**Invalid-Input-Test**–

*PreCond:* judge clicks on stats button while less than four grades exist.

*Result:* Judge sees an empty graph – PASS

**Valid-Input-Test** –

*PreCond:* Admin Sends email to many judges.

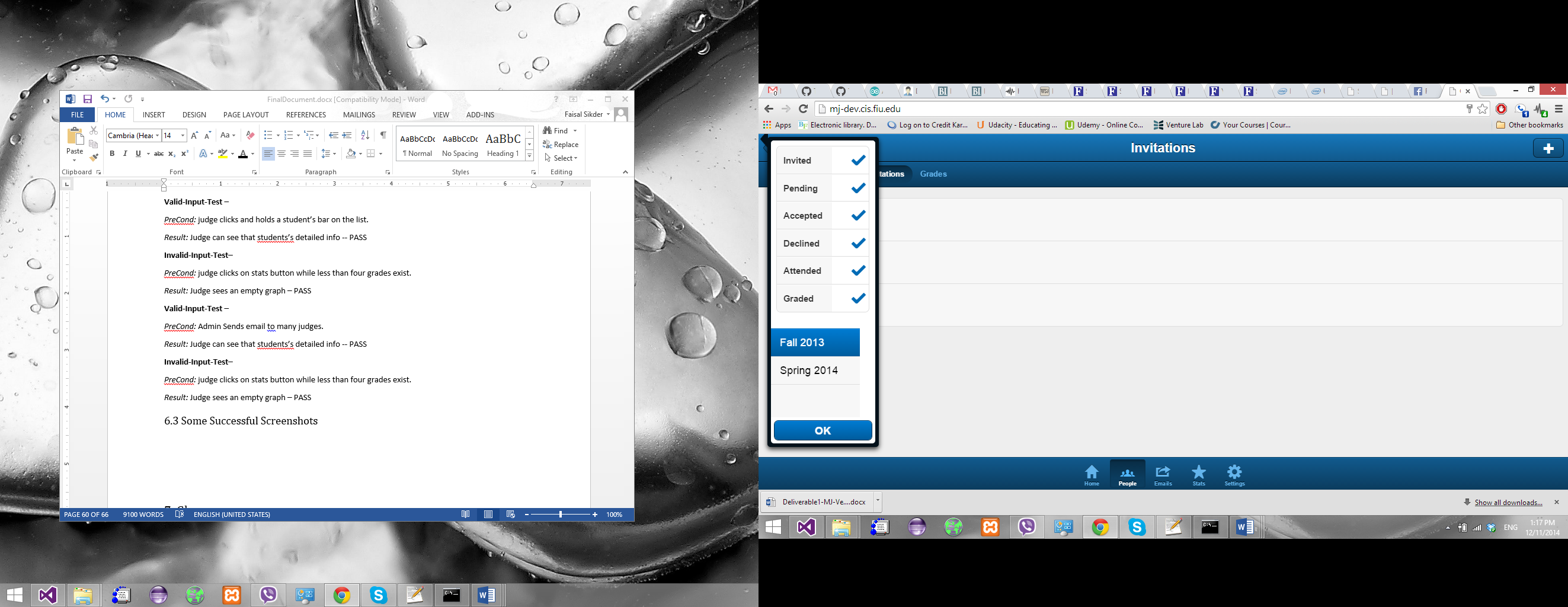
*Result:* Judge can see that students’s detailed info -- PASS

**Invalid-Input-Test**–

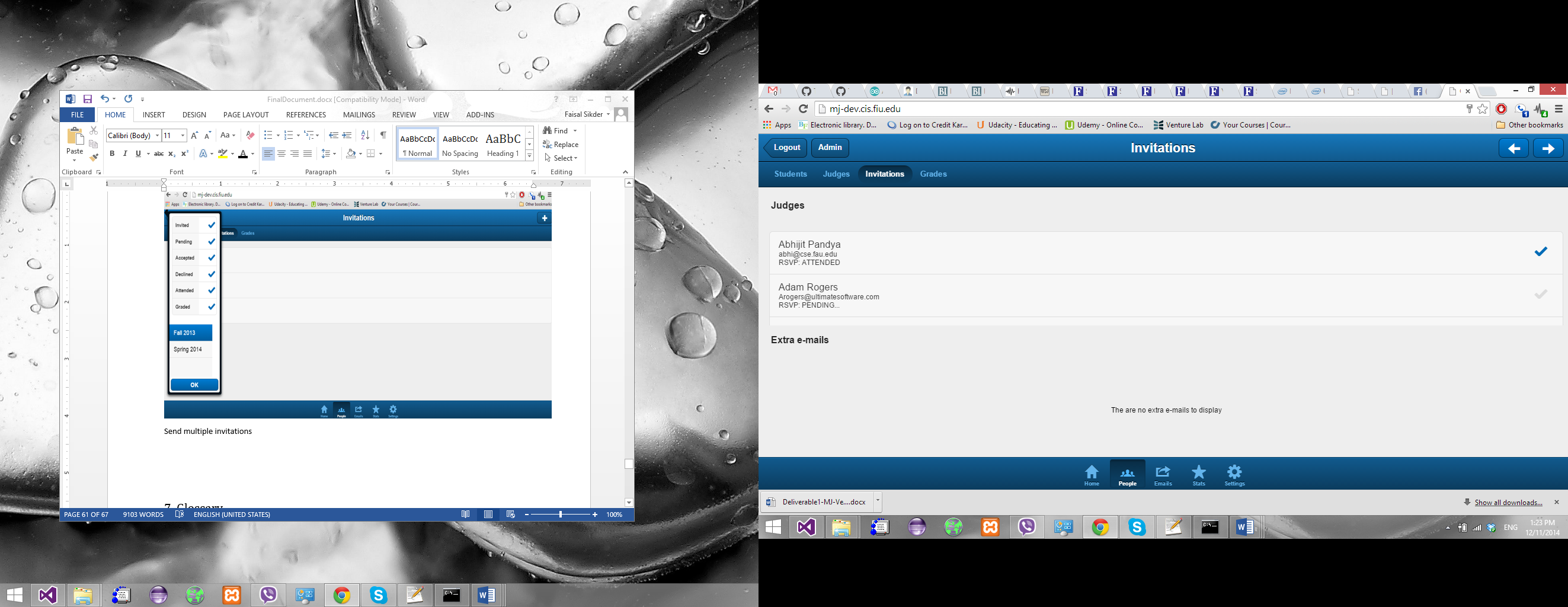
*PreCond:* judge clicks on stats button while less than four grades exist.

*Result:* Judge sees an empty graph – PASS

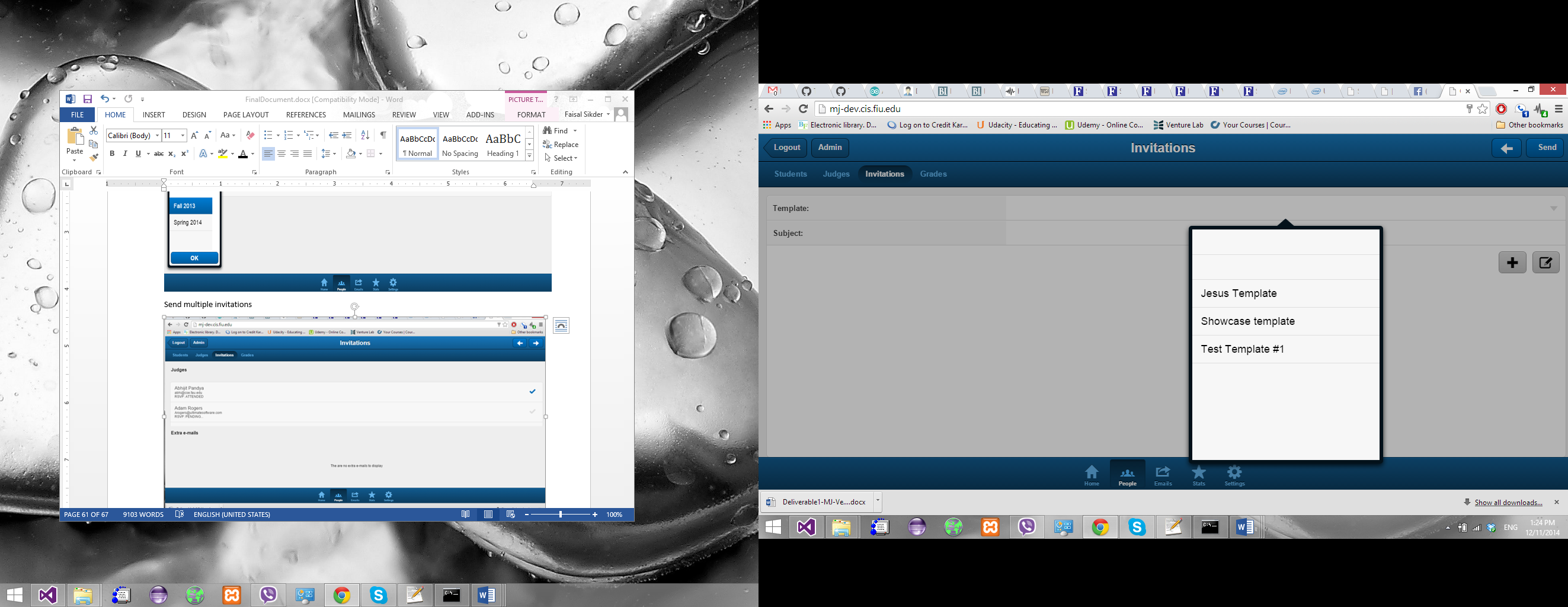
## 6.3 Some Successful Screenshots



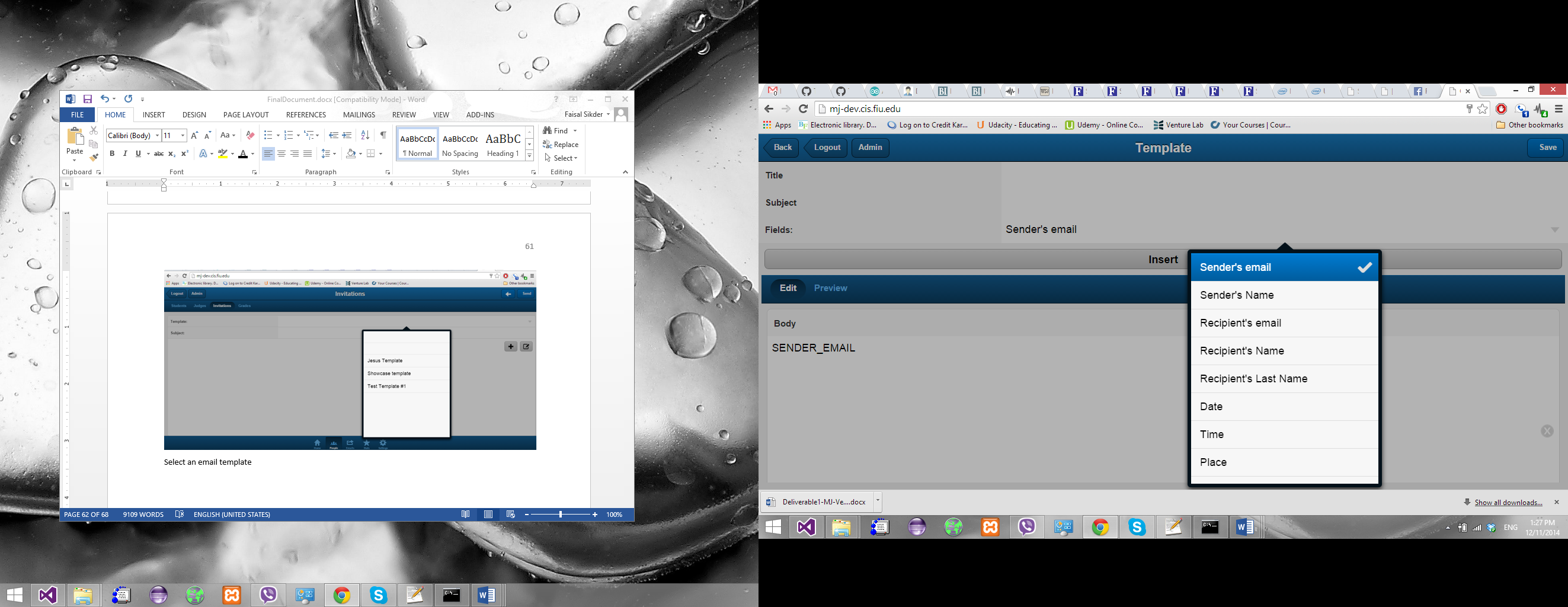
Send multiple invitations



Select judges



Select an email template



Select fields for email template.

# 7. Glossary

**Android** A mobile operating system offered by Google.

**App** A mobile application that can be installed on smartphones and tablets.

**Windows Mob** A mobile operating system offered by Microsoft.

**HTML** Stands for HyperText Markup Language. It is used while creating web pages.

**iOS** A mobile operating system offered by Apple

**JavaScript** A programming language mainly used while creating websites.

**Task** A piece of job that serves as a unit of work.

**UML** Stands for Unified Modeling Language. It is a standardized language that is used to model various things within the field of software engineering.

**Senhca Touch** A framework to create mobile apps that is multiple platform and depends on the user’s browser rather than their OS.

**phpMyAdmin** A system, that is implemented by a server, that allows the server manager to directly handle the MySQL database from their browser.

**MySQL** An open source database system that is used to store information about users and their relation to other data.

# Appendix

## 4.1 Appendix A – Project Schedule (Agile Approach)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Task Decription | **Duration** | **Start Date** | **Finish Date** |
| Sprint 1: |  | 4 weeks |  |  |
|  | ·         Requirements Definition |  | 9/2/2014 | 9/28/2014 |
|  | ·         Developers Roles |  | 9/2/2014 | 9/28/2014 |
|  | ·         Initial Mockups Design |  | 9/2/2014 | 9/28/2014 |
|  | ·         Environment Setup |  | 9/2/2014 | 9/28/2014 |
| Sprint 2: |  | 2 weeks |  |  |
|  | ·         Multiple Email Static View |  | 9/29/2014 | 10/12/2014 |
|  | ·         Statistics Graph/Chart Static View |  | 9/29/2014 | 10/12/2014 |
|  | ·         Login Sequence Modification with Gmail |  | 9/29/2014 | 10/12/2014 |
|  | ·         View as for Admin Static View |  | 9/29/2014 | 10/12/2014 |
|  | ·         Automatic Grade Check Static View |  | 9/29/2014 | 10/12/2014 |
|  | ·         Unit Testing |  | 9/29/2014 | 10/12/2014 |
| Sprint 3: |  | 2 weeks |  |  |
|  | ·         Multiple Email PHP API Development |  | 10/13/2014 | 10/27/2014 |
|  | ·         Statistics Graph/Chart View |  | 10/13/2014 | 10/27/2014 |
|  | ·         Statistics PHP API |  | 10/13/2014 | 10/27/2014 |
|  | ·         Automatic Grade Check PHP API |  | 10/13/2014 | 10/27/2014 |
|  | ·         Unit Testing |  | 10/13/2014 | 10/27/2014 |
| Sprint 4: |  | 3 weeks |  |  |
|  | ·         Multiple Email Template Database |  | 10/28/2014 | 11/19/2014 |
|  | ·         Statistics Judge Graph/Chart |  | 10/28/2014 | 11/19/2014 |
|  | ·         View as for Admin PHP API |  | 10/28/2014 | 11/19/2014 |
|  | ·         Feature Integration |  | 10/28/2014 | 11/19/2014 |
|  | ·         Unit Testing |  | 10/28/2014 | 11/19/2014 |
| Sprint 5: |  | 2 weeks |  |  |
|  | ·         Multiple Email Feature Final |  | 11/20/2014 | 12/1/2014 |
|  | ·         View As as Admin Feature Final |  | 11/20/2014 | 12/1/2014 |
|  | ·         Graph/ Chart Final |  | 11/20/2014 | 12/1/2014 |
|  | ·         Automatic Grade check Final |  | 11/20/2014 | 12/1/2014 |
|  | ·         Unit Testing Integration Testing |  | 11/20/2014 | 12/1/2014 |
|  | ·         Deployment |  | 11/20/2014 | 12/1/2014 |

4.2 Appendix B – Feasibility Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| Feasibility Criteria | Weight | Alternative 1 | Alternative 2 |
| Operational Feasibility | 30% |  |  |
| Functionality: To what degree does the candidate solution benefit? |  | Fully supports required  Functionality. | Fully supports required  Functionality. |
|  |  | Score: 100 | Score: 100 |
| Technical Feasibility | 30% |  |  |
| Technology: An assessment of the maturity, availability, ability to acquire, and desirability of computer technology needed to support this candidate. |  | All smartphones platforms would be supported | All smartphones platforms would be supported |
| Expertise: An assessment of the technical expertise needed to develop, operate, and maintain the candidate system. |  | Extensive knowledge on database design and medium level of Sencha Touch is required | Medium level of expertise in database and Sencha Touch are required |
|  |  | Score: 75 | Score: 90 |
| Economic Feasibility | 30% |  |  |
| Cost to Develop |  | Database Hosting: Self  Maintained/School's  Resources - 0 | Database Hosting: Self Maintained/ School's  Resources - 0 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| Payback |  | Same as 1. | Same as 1. |
| Net Present Value |  | NA | NA |
|  |  | Score: 100 | Score: 100 |
| Schedule Feasibility |  |  |  |
| Assessment of how long the solution will take to design and implement. | 10% | 4 -5 months | 3 to 4. |
|  |  | Score: 85 | Score: 90 |
| Ranking |  | 83.5 | 93 |