**Software Design Document**

InPress - Group 3

Liu Wenbo (0970709)

Justin Kan (0843763)

Abdul Hadi Muhammad (0951862)

# **1. INTRODUCTION**

The following high level design conforms to the IEEE Standard 1016-1998.

## 1.1 Purpose

This document provides a high level description of the design and implementation of InPress. This document has been written for software engineers, software architects, and technical program managers.

## 1.2 Scope

InPress is a web-based application which allows students in universities and colleges to respond real time to questions posted by their instructor. InPress confines to three main design principles - Separation of Concern (separating the software into distinct sections), Non-Repeatable Code (reducing repetation of information of all kinds) and Simplicity (keeping the code simple and elimintating unnecessary complexities).

## 1.2 Overview

Topics covered in this document include the following:

- System Overview - Brief overview of the product;

- System Architecture - How the product is partitioned, a description/rationale of each partition;

- Data Design - How data is stored, processed, and obtained;

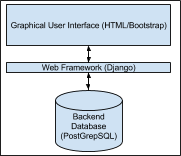
- Component Design - Summary of algorithms used in each partitiation;

- Human Interface Design - Frontend design;

- Requirements Matrix - Linking requirements to design

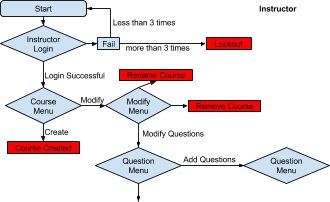
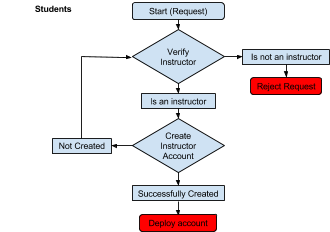
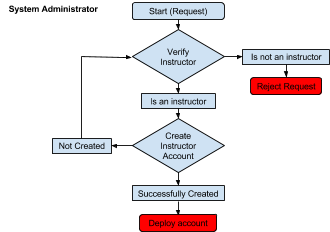
# **2. SYSTEM OVERVIEW**

InPress can be subdivided to two perspectives - the Student, and the Instructor. Instructors are able add/remove courses, enroll students to their respective courses, create/remove assessments, and view/analyze result. Students are able to view/take a particular assessment they are enrolled in. All users interact with the system with user friendly graphical interface. The frontend of InPress is made fully in HTML and Bootstrap (<http://getbootstrap.com/>). Backend is create in PostGrepSQL (<http://www.postgresql.org/>). Communication between the frontend and backend are done via the web framework Django (<https://www.djangoproject.com/>).



**Figure 1**  Basic System Overview

<< PUT FULL DIAGRAM OF INPLACE HERE >>

Data flow among major portions of your system

Program/subprogram structure

Highlights of the user interface characteristics and interaction

Major internal data structures used by the program

*General Functionality:*

*System Adminstrator:*

*Add instructors*

*Instructors:*

*Create / modify courses, questions (MC, short answers)*

*Students:*

*Add courses, answer questions (MC, short answers)*

*Connection to requirements and design*

# 

# 

# 

# 

# 

# 

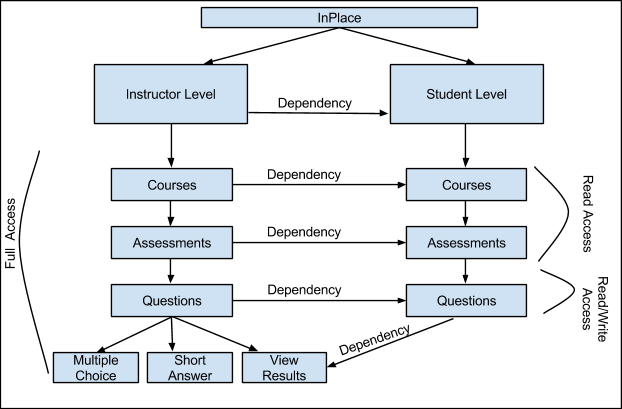
# 

# **3. SYSTEM ARCHITECTURE**

## 3.1 Architectural Design

Our product has two major sublevels - the instructor level and student level. We can further divide the instructor level into the various actions that the instructor has to his/her disposal such as add/remove classes, add/remove assessments, add/remove questions, and view class

results. The student level can be further divided into three areas - view available courses that he/she is enrolled in, view assessments available, complete an available assessment.



**Figure 2** Decomposition of InPlace

## 3.2 Decomposition Description

The division of the instructor and student level is regulated with a login screen. Instructors will be required to login to InPlace with their unique username and password. Students will also require a login, but their login has to be created when the instructor creates the course.

At the instructor level, there are a total of three subsystems. The first of these subsystems are “Courses”. Instructors are able to add and remove classes. While creating a course, the instructor is asked basic questions of the course being offered such as the course name, course code, and a file (which is uploaded) listing all students needing enrollment in the ccourse. Upon submitting this information, InPlace creates the course, and login ids for all students enrolled in the course.

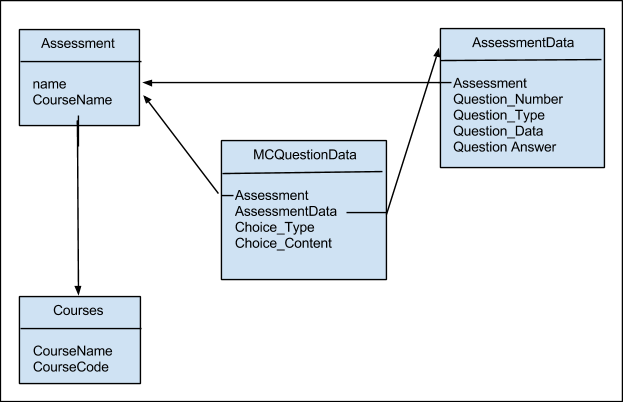
Once the course is created via the “Courses” subsystem, the instructor may chose to interact with the “Assessment” subsystem, which allows the instructor to add or remove assessment for a specific course. For an assessment to be created, there must be at least one course created. “Assessment” is a subsystem of “Courses”. Only the name of the assessment is required for the instructor to create an assessment.

Once an assessment has been created via the “Assessment” subsystem

## 3.3 Design Rationale

# 

# **4. DATA DESIGN**



## Figure 3 Database Schema Design

## 4.1 Data Description

The product will have types of data to use. User profile data, course data, subject/quiz data and data for each question.

## 4.2 Data Dictionary

Because we have User data, Course data, Subject data, and Question data. We need 4 datatables for this product. User datatable contains the login information about each user, includes the userid, username, passwords, usergroup etc. Userid will connect the course datatable with User table. Course table will have course name, courseid, course belonging etc. Subject table contains the courseid it belongs to, the subject name, subjectid etc. Question table has subjectid, questionid, question name, question answer etc.

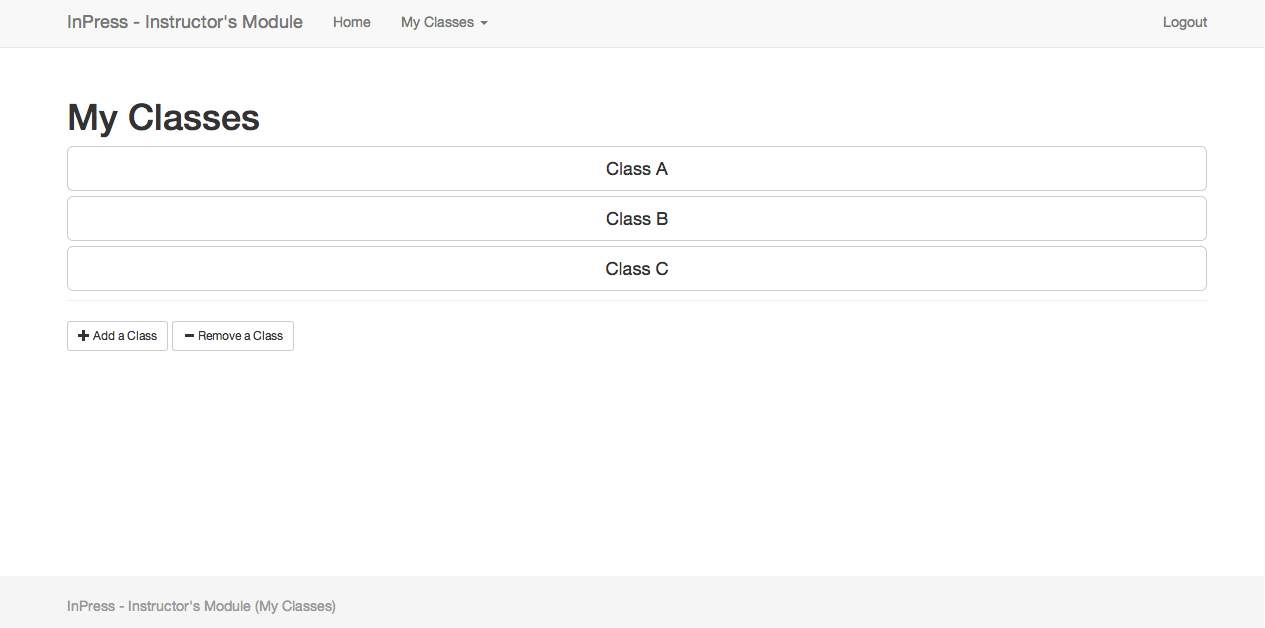
# 

# **5. COMPONENT DESIGN**

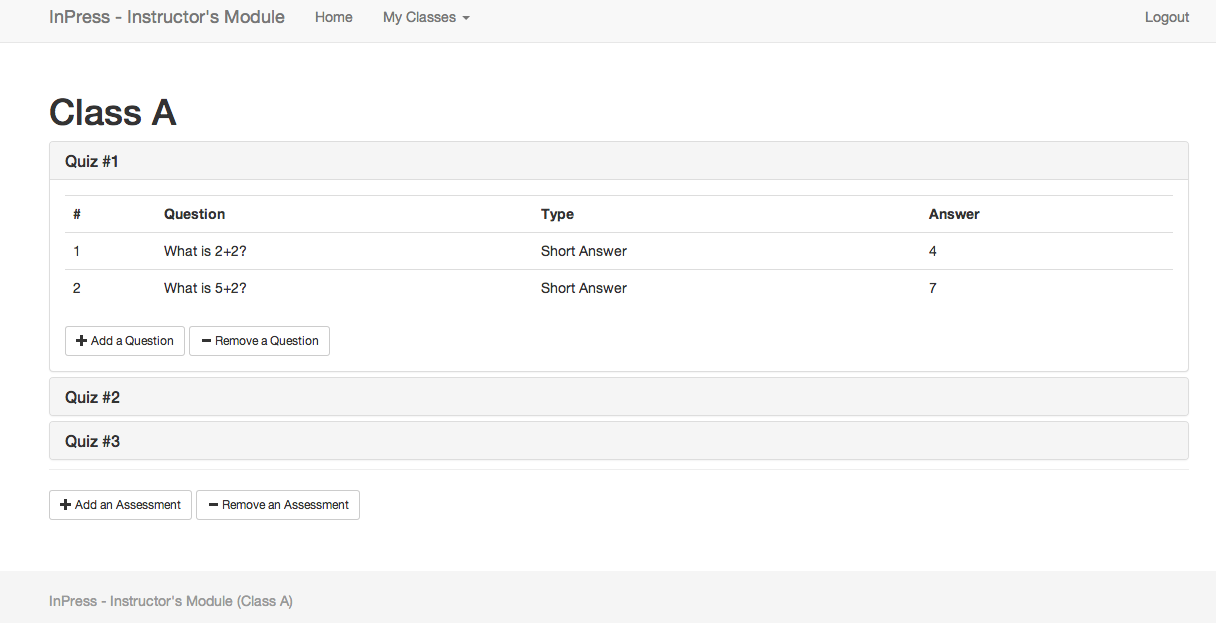
# **6. HUMAN INTERFACE DESIGN**

## 6.1 Overview of User Interface

This product is based on web pages, operating by both remote device and computer. Therefore, the software should capable for both small screen and normal screen interface. We are using Bootstrap provided interface design function that can auto-adjust the same page for different devices.



**Figure 4** Instructor Module: My Classes



**Figure 5** Instructor Module: Class View

<< Add mobile screenshots >>

## 6.2 Screen Images

The whole site will keep in the same theme, we try to keep it simple, and considered the design style from ios 7 and windows 8. We followed the Simple Stupid principle to make each page contains less words and less figures. This could make user’s life easier to find out how to use and increase the system performance.

## 6.3 Screen Objects and Actions

For revision 0, we have title and subtitles for each page different from the normal contents. different question type requries different answer inputs. For Multiple Choice question, we have lists to let user choose, for Short Answer and Number question, we have text boxes in different length. One page only has one purpose.

# 7. REQUIREMENTS MATRIX

- An overview of key algorithms.

- Analysis of Data (Multiple Choice answers, Short Answers)

One algorithm is compare the option user inputs with answer for MC type questions. Another one algorithm is compare the keywords user inputs with Short Answer keywords requirement, by using the customized keywords amount and matching percentage acceptance, system will decide the answer is correct or not.

- Login Algorithms (Security protocols)

By trace the username in the database first then compare the password. For correct check, username should exists and password should exactly match.

- Security protocols (use https)

- Database operations (read, write)

Database should correctly write and read the data from/to database for user to

- Detect multiple users (Restrict users to login only once)

This algorithm is log every user’s creteria and all the requesting data will based on this user’s creteria, only when this user logout, system will restrict user information untill next user login.

- Interface descriptions.

-

- Database structure descriptions.

- Overview of major tools to be used.

- Communication protocols specification.

- Module or class decomposition.

- Each major screen of the UI should be identified

- Purpose described.

- Not exact layout.

- Relational database structure.



**One document template**

<http://www.ceng.metu.edu.tr/_media/course/ceng490/sdd_template.pdf?rev>

For Reference for above template (which adapts to the IEEE template):

<http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=05167255>

Detailed Example:

<http://www.if.uidaho.edu/~hunthl/CS480/sdd_gl.pdf>