# LabAssist
# Design Document

*Adam Cantrell*

West Virginia University Institute of Technology

*Benjamin Culkin*

West Virginia University Institute of Technology

Version: 1

## 1. Introduction

### 1.1. Document Overview

This document describes the design of the components that LabAssist uses:

- A web interface used for interacting with the database and performing the outlined functions.
- A database used for storing and querying information.
- A background job used for sending mail notifications.

### 1.2. References

## 2. Software Architecture Overview

LabAssist is built from three main components: A web interface for interacting with that database, a database that stores information and prepares statistics from it, and a background job that sends mail notifications.

The only component that the users need know exist is the web interface. It provides users the ability to access all of the provided features of the software, and it does this by interacting with the database; both sending data to update the database, and running queries to retrieve data.

The database is the central component of the system, by virtue of being the place where all of the data that the system needs is stored and processed.

The background job is a task that is set to run every twenty to thirty minutes. Its job is to retrieve any pending notifications from the database, merge together any messages that can be merged, and then send out the merged notifications via email.

The system is designed to run off of any system running the following:

- PHP 7
- Web Server
- Postgres 9.6

## 3. Software Design Description

### 3.1. Component 1: Web Interface

The job of the web interface is to provide a way for the user to interact with the rest of the system.

#### 3.1.1. Component Interfaces

This component interacts with both the end-user and the database, getting input and output from both.

#### 3.1.2. Component Design Description

The web interface has three main components:

User Mode

User mode is the component that the staff will interact with the most. Students will also interact with it occasionally, mainly to use the question and answer system. User mode is more a container for three subcomponents:

Question & Answer

The question and answer system will allow for students to ask questions, tutors and staff to respond to those questions, and then for students to ask follow-up questions as well.

Reporting

The reporting system provides the usage reports for the labs. These include:

- Lab Utilization Per Section
- Lab Utilization Per Time Period
- Lab Utilization Per Professor

User Management

The user management provides the capability for staff and administrators to manage the users of the system, including who tutors are, who staff are, and who administrators are; among other things.

Kiosk Mode

Kiosk mode is the component that will be the one that students interact with the most. It allows the students and tutors to clock in and out of specific lab sections, and provides the database with the information necessary to provide analytic reports.

Login

The login component is the first component that the user sees and the one that is responsible for allowing the users to use the other components. Upon using the system for the first time, the user will directed to the User Registration sub-component; otherwise, they will be directed to the component that they selected.

User Registration

User Registration is responsible for taking first time users and associating their real name, email and username with their student ID number.

### 3.1.3. Workflows and Algorithms

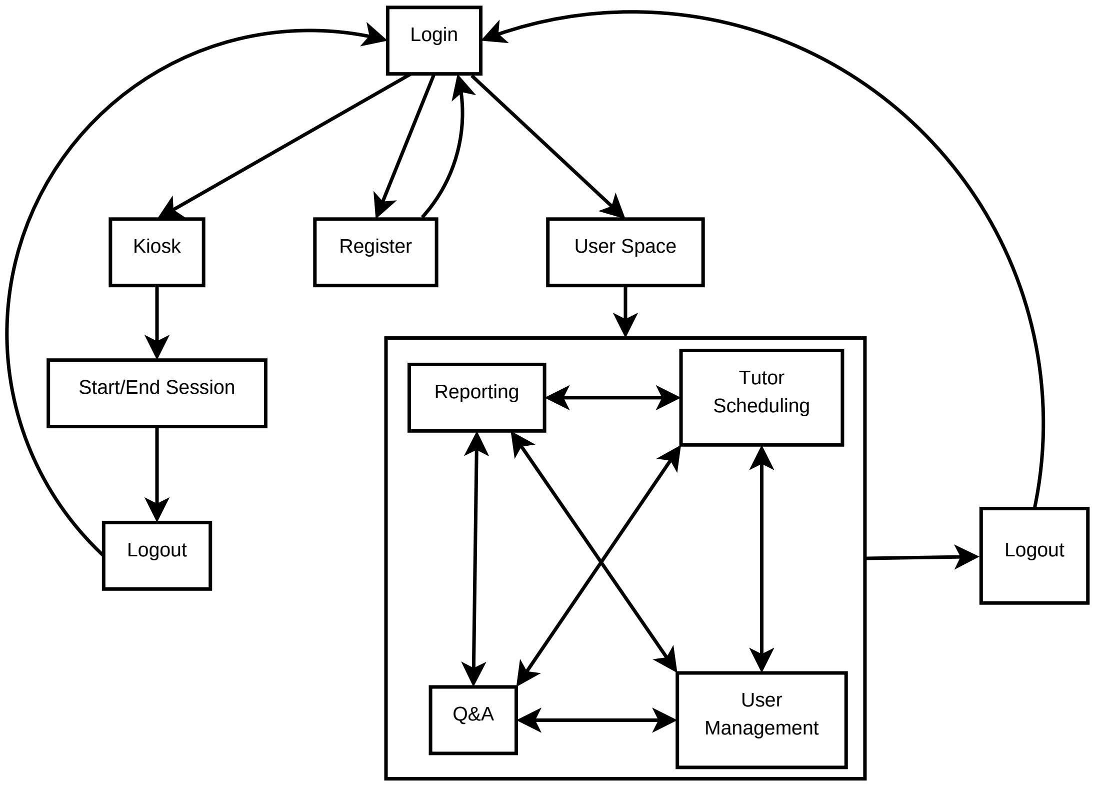

**Fig. 1:**Web Interface Flow Diagram

### 3.2. Component II: Database

The database serves as the place where all of the data that LabAssist uses is stored and analyzed.

#### 3.2.1. Component Interfaces

This component interfaces with both the web interface and the mailer. It gets information from the web interface, and then sends information out to both the web interface and the mailer.

#### 3.2.2. Component Design Description

The database is stored as a collection of SQL tables split into the following categories:

Users
: This group of tables is concerned with storing users and what roles they have.

Class Usage
: This group of tables is concerned with storing data about classes and when people are using the lab.

Question & Answer
: This group of tables is concerned with storing the data necessary for the question and answer system to work.

Scheduling
: This group of tables is concerned with storing the data necessary for the tutor scheduling feature to work.

#### 3.2.3. Workflows and Algorithms

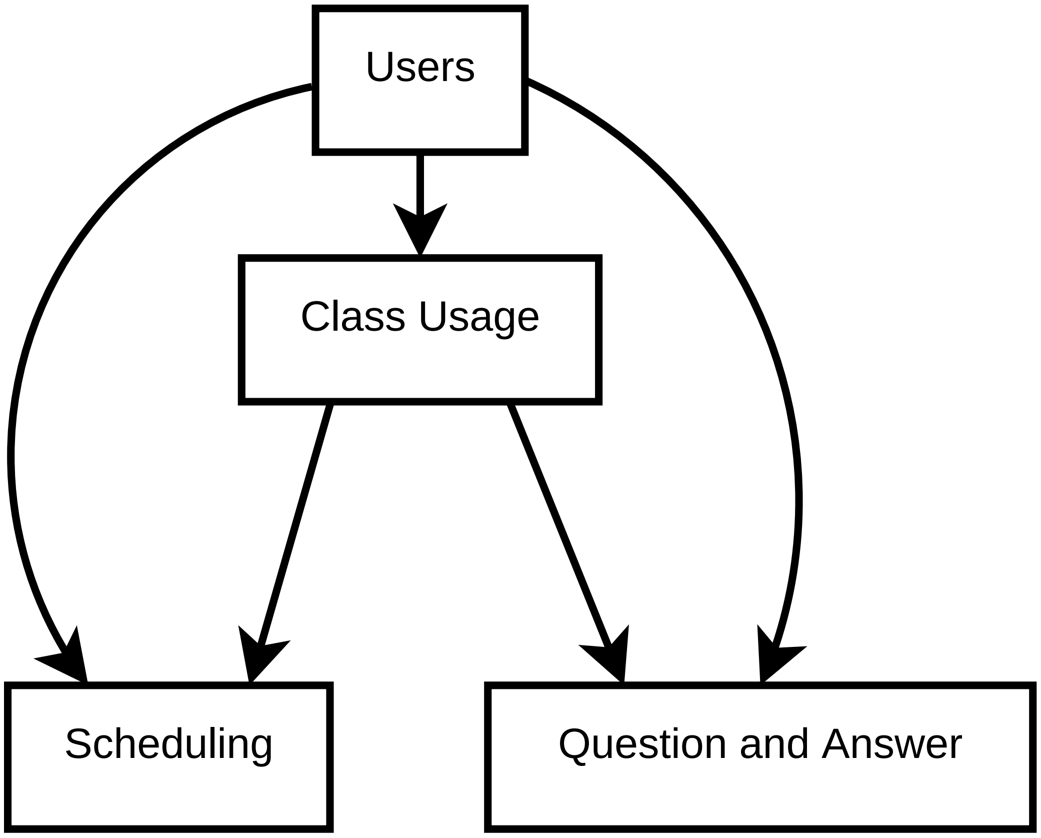

**Fig. 2:** Table Category Dependancies

For a more detailed description of the database, see the attached database schema.

### 3.3. Component III: Background Mailer

The job of the background mailer is, as previously noted, to connect to the database, retrieve messages, merge them and then send out the merged messages.

#### 3.3.1. Component Interfaces

This component interfaces with both the database and an external SMTP server, using the database to retrieve messages to send, and the SMTP server to send the messages to users they are supposed to go tgo.

#### 3.3.2. Component Design Description

The mailer is designed as a java application split into three main parts: Grabber, Batcher and Sender. The roles of the components are as follows

Grabber

The grabber component functions to retrieve messages from the database and convert them into Message objects.

Batcher

The batcher component functions to take a bunch of Message objects, and then combine compatible messages to cut down on the total number of mail messages sent.

Sender

The sender takes Message objects and converts them into

SMTP messages to send them.

#### 3.3.3. Workflows and Algorithms

To allow for easier editing of message types and to simplify the addition of more message types, the mailer uses a text-based template for each message type with placeholders that will be filled in by the body variables from the message.

To cut down on the amount of notifications sent, the mailer uses a merging algorithm consisting of performing the following steps on every message:

1. Get the stored messsage of the correct type for the recipients of the current message, or create one if there is no stored message for the recipients of the current message.
2. Merge the body variables of the current message into the stored message. The way this happens is specific to the type of the message, but in general it is done by appending the two variables together, possibly with a seperator.

## Table of Contents