Final Project Report

Eric DiGioacchino - Jay Kerbelis - Claudia Usis - Alan Benitez

Capstone Project [Spring, 2022]

The University of West Florida

4/25/2022

CIS4595 - Capstone Project

Bernd Owsnicki-Klewe

# Table of Contents

[**Table of Contents**](#_heading=h.1ik5s0jbjit4) **2**

[**1 Executive Summary**](#_heading=h.4d8zycmowci2) **3**

[**2 Final requirements and comparison with initial requirements**](#_heading=h.116ug48jv3sr) **4**

[2.1 Architecture](#_heading=h.8wmlfxr9igif) 4

[2.1.0 Tier](#_heading=h.rc5pvh5e5gd5) 4

[2.1.1 Strategy and Tools](#_heading=h.kxp31tps4we2) 4

[2.2 Comparison](#_heading=h.pzejhb1g3tlt) 5

[**3 Final timeline and comparison with the initial timeline**](#_heading=h.r49kww6b65w5) **6**

[**4 Project results compared with expectations**](#_heading=h.20yn2syvmso) **7**

[4.1 Expectations](#_heading=h.lo4kx0w1l06n) 7

[4.2 Final Results](#_heading=h.kty1lrh6wybg) 9

[**5 Software Evaluation**](#_heading=h.c3c7gkdlskrs) **10**

[**6 Work to be done**](#_heading=h.kk187hexhe8) **11**

# 1 Executive Summary

The proposed project is an implementation of a note taking program. It might be a personal thing, but our target users, students and faculties might find this very relevant to gather information. For instance, students could have a hard time to confirm the information they’ve gathered unless they will be letting their teacher check it, so the program helps to verify the accuracy of the notes, and since it is digital, it is much more legible to share it to other users. Users can register by choosing their role – either a student or a teacher, and by using their own choice of username and password. When logged in, users should be able to create, edit, or delete notes. By doing these, it will hopefully allow users to achieve our project goals such as helping to patch sparse notes of users, and having an immense growth with the user’s note taking skills.

# 

# 2 Final requirements and comparison with initial requirements

In this section, we are listing the architectures and models used for the development of our final program.

## 2.1 Architecture

This section will outline the tiers that we have used to make this application possible.

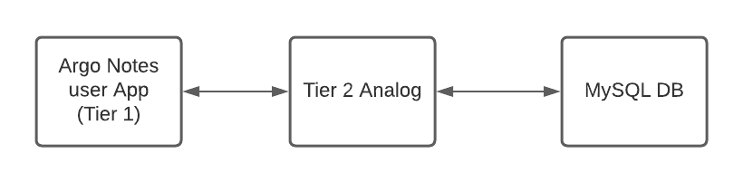
### 2.1.0 Tier

Argo Notes was written using a three-tier architecture as its foundation. For our first tier, we have used Java to create our core application – this includes the ability to to execute SQL operations, GUI, and handling user inputs. We have used JDBC, a *Java Database Connectivity* that allows us to create a connection to our MySQL database. An event-driven server platform is also needed and we’ve decided to use NodeJS since it is easy to set up and to extend. Lastly, we have used MySQL database to store any of our data consisting of User credentials, Role (either a student or a teacher), and the created or edited notes.

### 2.1.1 Strategy and Tools

Our final requirement for testing our Java application is through testing in isolation and we’ve used JUnit5, which has an up-to-date foundation for developers. We’ve also used the same Unit Testing as for our Tier 2, which is the Node JS. The MySQL parts were evaluated through an analysis schema that creates a view to test each table’s cardinalities.

## 2.2 Comparison

Comparing our final requirements with our initial requirements, it is very obvious that we’ve followed our plan thoroughly. We made each of the tier possible and we still have decided to build an interface that connects our core application and our MySQL database through a single server thread instead of using a remote server. Hopefully, by implementing more of our future planned features, we’ll get more users and we can upgrade our server to a remote one, so it could be accessible by many. Below is a figure that we’ve followed from our initial plan.

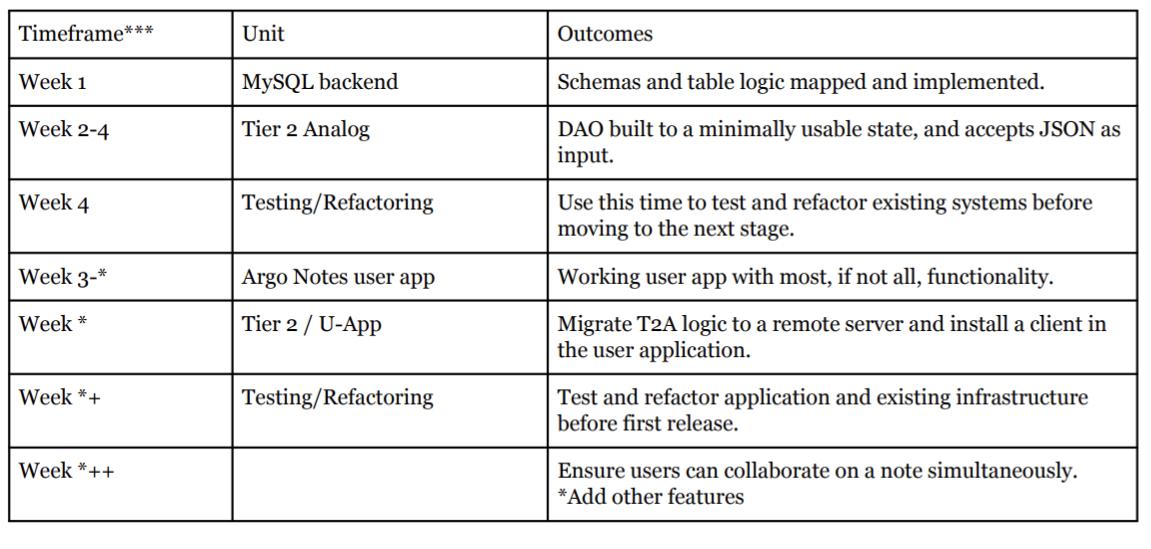
For testing, we have also decided to use the same methods since the chosen testing units are the simplest yet the most efficient to test each of our tiers.

# 3 Final timeline and comparison with the initial timeline

Our final timeline is pretty similar to our initial timeline. The testing and refactoring part for each of the tiers took us the longest to complete. We kept going back and forth implementing the MySQL Database and Argonotes Application since we have coded our functionalities one at a time to make it easier for testing. As soon as we have implemented all the functionalities, we tested the application again as a whole.

| **Timeframe** | **Unit** | **Outcomes** |
| --- | --- | --- |
| **Week 1** | **MySQL backend** | **Schemas and table logic mapped and implemented** |
| **Week 2-4** | **Server/ Tier 2 Analog** | **DAO built to a minimally usable state, and accepts JSON as input** |
| **Week 4** | **Refractor Server** | **Use this time to test and refractor existing systems before moving to the next stage** |
| **Week 4-5** | **Argo Notes User app** | **Working user app with most functionality** |
| **Week 5-6** | **Tier 2/ U-App** | **Migrate T2A logic to a remote server and install a client in the user application** |
| **Week 6-7** | **Testing/ Refactoring** | **Test and refractor application and existing infrastructure** |
| **Week 7-9** | **Client Refactoring** | **Test interaction to API** |
| **Week 10-12** | **Merge Security Updates/ Test/Refactoring** | **Test to make sure the server- client server is functioning properly** |
| **Week 12-16** | **Delivery** | **Ensures all of the functionalities are working how it is planned** |

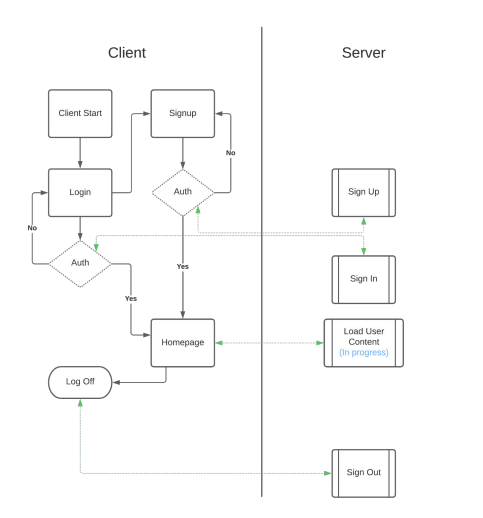
**Please see the table below for our initial timeline:**

****

# 4 Project results compared with expectations

## 4.1 Expectations

The success we experienced was a result of good leadership and a great planning phase. From our first presentation, the expectation of our product was to have a functional and stable database, a stable tier 2 analog, and a Java client with a functional and lightweight UI. Additionally, we planned to deliver Installation Instructions and Usage Instructions. Our project workflow diagram, from presentation two, shows how we wanted our application to function.

****

In presentation two, there were multiple known security vulnerabilities that we needed to address. The most critical vulnerabilities discovered involved authentication. Those being; SQL Injection and Rainbow table attacks. At that time, SQL Injection attacks would result in our application crashing.

## 4.2 Final Results

The final application uses a lightweight graphical user interface (GUI), a Tier 2 NodeJS server, and a functional and stable SQL Database. The only major adaptation we made to the original plan was to convert the Tier 2 Analog to an actual Tier 2 NodeJS server. Our initial workflow diagram matches our final product almost exactly. Additional functionality that is not included in the workflow diagram above is the ability to create, save, edit, and delete notes.

We have provided ‘Deployment/Installation Instructions’ to assist the user with downloading and installing required components, configuring the NodeJS Server and SQL Database, and then launching the application. Additionally, we have included a ‘User Manual’ to demonstrate how to use the application properly. Lastly, we included an ‘Admin Manual’ to explain how to address scaling and highlight important features.

To address the SQL Injection vulnerability, we implemented use of the ‘sqlstring’ javascript library. We formatted our SQL queries to use parameterized inputs in order to escape special characters and prevent users from manipulating our login and sign up queries. To address our vulnerability with rainbow table attacks, we implemented a salting/hashing algorithm to store our passwords. A salt and hash are performed twice (once on the client and once on the server) before being stored in the database.

# 5 Software Evaluation

After the installation of all the tools needed for running the program, the software works as expected. Some things that we had planned to include in the release are now missing from the project due to different circumstances and difficulties that we encounter as we were doing the program. However, the program is capable of connecting with the database and use a server thread, and we were able to accomplish the three-tier architecture that we planned in the beginning of the release.

All the features that we implemented such as creating an account, log in, and log out works as expected, as well as the note taking itself. We were able to use the three tier architecture to create a new note, edit, delete, load another note and save it, so that is a great thing that we accomplished when we set our goal before starting the program.

There are a couple of features that could’ve been added in order to enhance the software and will make the program even better, which is working with clustered notes, having the students collaborating on notes, which could be adaptable with our project for a future release.

# 6 Work to be done

* Upgrade GUI
  + Professional looking design
  + ‘Show Password’ and ‘Reset Password’ functionality
* Upgraded note taking interface
  + Fonts, Text Sizing, Highlighter, Text formatting options
* Real time collaboration between multiple users
* User Classing
  + User Clustering on Class
  + Teacher Homepage and Student Homepages
* Clustering
  + Teacher as cluster host
  + Students as population of cluster
  + Teacher can manage their clusters
  + Students can create clusters of other students
* Known Security Vulnerabilities
  + Convert from http to https
  + More vulnerability scanning to detect all possible vulnerabilities
  + Accounts created through the web client will only be able to login through the web client. However, this is a non-issue because we never intended to use a web client for our application.