Prison Garden Application

Design Document

Team Number:	may1713
Client:	Julie Stevens
Advisor:	Doug Jacobsen
Leader:	Juan Venegas
Webmaster:	Andrew Dailey
Webmaster:	Jacob Stimes
Communications:	Matt Koenig
Concept Holder:	ChiaJun Tai
Concept Holder:	Jacob Anhorn
Team Email:	may1713@itstate.edu
Team Website:	https://may1713.sd.ece.iastate.edu:657
Revised:	1 November, 2016

Contents

1 Introduction

1.1 Project statement

The Prison Garden Production app will be a tool for inmates to use in prisons across Iowa for educational and rehabitual purposes. It will provide inmates with skills they can use to re-incorporate themselves into society, while also giving them a sense of accomplishment. This app will meet prison security standards regarding internet access by restricting wifi. A connection through a secure server will be made for the faculty to track inmate progress.

1.2 Purpose

The purpose of the Prison Garden Production app is to educate and rehabilitate inmates by allowing them to more efficiently track the development of their gardens within the prison. Society will benefit from this by seeing a smaller rate of reincarceration. Inmates will leave prison with a new set of skills and the confidence to use those skills in the working world.

1.3 Goals

Through this project we hope to provide a product that is affordable and feasible to implement in Iowa's prison facilities. We hope to achieve creation of a secure server that will allow communication between inmates and ISU gardening experts that adheres to prison security standards.

2 Deliverables

These tie in with the goals. What deliverables are necessary to meet the goals outlined in the introduction?

This project has two primary deliverables:

- Android Application - Contains gardening instructions, progress tracking, and exam system. Reports content back to a central database and supports multiple users.
- Backend Web Server - This will handle connections between the Android application and our database. Data to be stored and access includes user credentials and lesson plans.

Additionally, there will likely be supplementary deliverables created to accompany the two mentioned above:

- User Guide - Instructions for using the application from the perspective of a normal user (inmate). These will need to be easy to read and as non-technical as possible.

- Administrator Guide - Instructions for managing the application from the perspective of a project administrator: likely a member of Julie's team. This content should explain how to perform the necessary modification of gardening content. There should exist a non-technical interface for easily adding new "lessons" to the application.
- Feature Compliance Document - This is a document that explicitly states the various prison restrictions (for the application) and the steps we took to ensure they are satisfied. As an example, if the only networking allowed is a service-level connection to the central database, then this document would describe how we disabled all other network capabilities.

3 Design

So far, our team has been brainstorming design ideas. Since our project is primarily a software one, prototypes and concepts can be mocked up with little overhead.

The Prison Garden Application will be designed to fit the Android platform. This was chosen for a few reasons. Firstly, it is an open source platform that was designed with development in mind, allowing for effective and intuitive application building. Second, it is an operating system that is supported by most tablets. This will allow us more options when it comes to deciding what specific hardware will run our application.

3.1 System specifications

Must abide by security standards set by the Iowa Prisons, most important of which is limiting access to wifi.

There shall be a secure server to handle communication to the database, where we will store the application's data.

3.1.1 Non-functional

The tablet needed for the app shall be very durable in terms of it possibly being dropped or even tampered with. The tablet will be given to inmates and needs to be sturdy so that it can not be modified on a hardware level.

The OS on the tablet shall have its security options modified. This is so that the the inmates will not be able to tamper with the app or tablet on a software level.

The application shall have its content organized in a user-friendly manner. The app will have a high volume of content to educate users on gardening. The UI needs to be organized into sections so that way the user does not get lost when trying to navigate the application.

The application shall be mobile friendly: specifically with tablets. The application will be very interactive and tablets are the current platform we will be developing on.

App shall support multiple user profiles. The app will need to able to handle different users signing into the app and be able to sync their progress back to a database.

3.1.2 Functional

The app shall prompt for credentials when it is opened. The app has to support multiple users and will need keep track of individual progress for each.

The app shall allow users to view lessons and instructions for furthering their progress within the prison's gardening program.

The app shall prompt the user for small objective quizzes after each lesson/educational section. After the user goes through the desired lesson/educational section, the app will prompt if the user is ready to take a quiz, and display the quiz when ready.

The app shall assist the user in the form of navigational tools and tips. When the user logs in, the app will give small helpful tips on how to navigate through the app or how to use some of the features.

The app shall prompt the user if they are ready to log out. When the user clicks on the logout button, the app will ask if the user is ready to log out.

The OS shall allow for persistent storage and the taking of photos, but only once the user has acquired elevated privileges.

The OS shall be able to sync the course progress of its users back to a central database in the prison. This process will start once the tablet is docked and will ensure all tablets contain the same content.

The OS shall restrict app access to only the gardening application created by our team.

The OS shall have a special, restricted profile to be used by the inmates. This profile will disallow the changing of settings and store no persistent data.

3.2 PROPOSED DESIGN/METHOD

- Modified Android Operating System
    - Using Android build options we will disallow camera usage, the installation and usage of other apps, and modification of advanced settings
    - Restrict network access to communicate with our server only
    - Restrict the saving and loading of content to/from external storage
- Gardening App
    - App should have educational content separated into 7 sections
    - Sections should have educational articles, informations, pictures, etc.
    - Sections should be accompanied by quizzes to track progress
    - Will be developed in Android Studio and have MVC
- Web Server
    - The app should connect to the web server through a restricted WIFI
    - Handle user authentication

    - Basic CRUD operations on users' coursework progress
- Database
    - We would like to store as much as possible in the database in order to prevent the memory capacity of the tablets from causing issues.
    - Users: Type, progress, name, password.
    - Lessons, articles, quizzes.
- Administration tool
    - Juan needs to email Julie & share a Google Doc on what she wants or her ideas on the tool

3.3 DESIGN ANALYSIS

Application Screen Ideas:

1. Make the screens for each of the lessons dynamic & interactive and have them display content
    a. We discussed this with Julie and her team and came to the conclusion that it is possible to have the screens for each of the lesson dynamic, and very interactive, but that it would require a large amount of time, and the team is not sure if it can be completed within the allotted time span.
2. Make the screens for each of the lessons content only with moderate interactivity
    a. This is more along what can be very achievable. Each of the screens for the lessons would only have to display text and pictures given by Julie and team, and the screens can some some interactability such as highlighting text and drawing on pictures. The only problem with this that if each of the lessons contains too much text or data it can create a fairly large application and take up too much storage. One way that could relieve this storage problem is if we have the the screens stored in either the server or database and have the app retrieve them when needed.

Server-side Ideas:

1. Direct JDBC connection to database
    a. This idea was quickly dropped due to likely connection issues. As we've read, mobile devices tend to have a fairly high tolerance for spotty networking (dropping packets, etc). Therefore, attempting a JDBC connection would likely cause more issues than its convenience would outweigh.
2. Java server (using sockets) as a mediator between app and database
    a. The idea of using sockets was dropped for a similar reason as JDBC. Unless we wanted to set the sockets to operate on UDP transporting, they'd suffer from the same connection issues. However, if we were more interested in UDP sockets, we might as well use HTTP requests.
3. C++ server with sockets
    a. Would likely be faster than the Java version, but would again suffer from the same connection issues.
4. Small REST API configured via nginx (web server)

    a. This idea seems viable and would also be the most minimal solution. However, with its simplicity comes a lack of depth of control. It wouldn't be able to explicitly handle authentication. Furthermore, its configuration would reside entirely in nginx's nginx.conf file, which doesn't use a programming language.
5. NodeJS REST API
    a. This idea would probably one of the easiest ones to set up. Also, it'd probably bring a decent set of customizability to the table. Though performance might not be as high as others (due to being JavaScript), this is a strong contender.
6. Erlang REST API
    a. Would definitely meet performance requirements
    b. Not many people really know Erlang - difficult to maintain
7. NoSQL database with built-in web frontend (CouchDB)
    a. Depending on how Julie's team seeks to store / serialize their lesson format, a NoSQL database may almost be required. While other databases generally have fields for JSON / XML objects, something like CouchDB would be more aligned with our goal, right out of the box.
8. AMP stack with "phprestsql"
    a. The AMP stack (apache, mysql, php) is a web standard in terms of usability. It has both the flexibility and performance to solve most web-related solutions. Adding phprestsql (a PHP library for interacting with SQL databases) into the mix, this idea would most likely fit all of our needs.

4 Testing/Development

4.1 Interface specifications

In terms of hardware, our team has two primary domains of concern:

- Choice of tablet device
- Existing systems / networks at the prison

The first is a choice that our team is actively discussing, both internally as well as with Julie's team. Though cost is not too much of an issue, there are certain criteria that need to be considered in regards to the safety and durability of the device itself.

As for the existing computer systems in the prison, there are still a few questions that need answered. Julie herself is unsure, but she has a contact link to an IT employee of the prison who will have a decent amount of insight. A list of questions is being gathered and some of the most impactful ones are shown below.

- Are there any existing servers that could support the addition of our backend web server?
    - What operating system does it run? (Hopefully Linux)

    - Are there any caveats to its network connectivity?
- What specific restrictions have been placed on the network, specifically wireless?
    - Could a mobile device connect back to the aforementioned web server?

4.2 Hardware/software

Ideally, our project will employ multiple levels of testing: from unit up to acceptance. Starting from the bottom, we can make use of JUnit for unit testing since it comes with the JDK. It may be worth noting that the application will be written in Java.

As we develop the main components of our system, it should be easy to concurrently write unit and module-level test cases to verify functionality has been added properly. For example, once we develop the client code for taking a certain quiz, we can test that correct and incorrect answers are handled correctly by the app. Similarly, as we add various request handling utilities to our server, we can verify with test requests that the expected responses are properly returned.

Above this comes system testing: the testing of the application as a whole. We, along with Julie's team, will devise specific use cases for testing the required functionality of the application. Examples of this include logging in, selecting lessons, and taking quizzes.

It is also important for us to perform stress testing during the development phase to ensure that our solution will perform adequately when deployed. For example, we must test that our system can handle storing data for each expected user of the system, and that data can be synchronized between the server and client efficiently.

Lastly, integration testing will be performed between our application and the prison. This includes setting up the prison's server and testing tablet connections, as well as discussing the application's usability with inmates and prison staffing.

4.3 Process

Application Screen Ideas:

The application screens can be white box and black box tested. We would first conduct a white box test by doing some small unit testing and then some integration testing to ensure that the screens can be integrated to the overall system without any problems. After testing the source code, we would move on to black box testing. Those on the programming team who are not working on the Android app would be the first to test the functionality of the screens then by Julie and her team to see if users can find any problems with the functionality.

Server-side Ideas:

Each of the backend server ideas was tested locally and evaluated based on the following criteria:

1. Customizability
2. Performance
3. Maintainability

A fourth section was also considered for evaluation: Ease of Setup. This field would have encompassed how much work / effort was needed to get from a fresh system to a fully operating server. This criteria ended up getting merged into the "Maintainability" category. Though the actual setup work would be scripted, getting a running example of some of the programs mentioned was daunting.

For example, CouchDB seemed like a top option when it was first investigated. However, installing and even building it was nearly impossible due to unforeseen errors. Installing and running CouchDB with apt (on Ubuntu 16.04) even failed, claiming errors in some Erlang documentation symlinks. Moving forward, we attempted to build it from source. While the build was successful, the server failed to ever start properly.

5 Results

Application Screen Idea Results:

See Appendix B for screenshots of the results.

Server-side Idea Results:

See Appendix A for detailed testing results.

After evaluating each idea based on the criteria mentioned in the previous section, three solutions stood as being more appropriate than the rest. In order from best to worst, here are the top three ideas:

1. REST - AMP (Apache, MySQL, PHP)
2. REST - NodeJS
3. REST - NoSQL Database

Something is clear from these results: a REST API is the ideal solution for handling interactions between the application and server. This arose due to the intermittent connection issues that exist in most mobile devices.

Furthermore, these three all ranked highly in maintainability, which servers to benefit both our team as well as anyone who is tasked with making changes to the system. Since our work with this project ends at the end of the Spring 2017 semester, being able to pass our progress on to others is a circumstance not to be overlooked.

6 Conclusions

The goal of our project is clear: to provide an effective and performant platform on which prison inmates can further their gardening coursework. In order to achieve our goals in the most successful way, we've been continually evaluating our design in search of the best ideas. We want our project be easy to use, easy to customize, easy to maintain, and perform well. Therefore, all four of these criteria are considered when making decisions and taking steps forward towards our goal. Our product will be fully tested on software and hardware before we deliver it to ensure that it fulfills our clients' needs.

Section 5 outlines our progress thus far in terms of ensuring the optimal design routes are taken.

7 References

Topic	Description	URL
CouchDB	NoSQL database with web REST API	http://couchdb.apache.org/
NodeJS	Stand-alone JS interpreter	https://nodejs.org/en/
Erlang	Programming language for real-time systems	https://www.erlang.org/
Nginx	Customizable web server	https://www.nginx.com/resources/wiki/
Sockets	Standard networking protocol	https://en.wikipedia.org/wiki/Network_socket

8 Appendices

Appendix A: Server-side idea test results

Scale: 1 = Bad 2 = Good 3 = Great

Idea	Customizability	Performance	Maintainability	Total
JDBC	2	1	3	6
Java Sockets	2	1	3	6
C++ Sockets	2	1	2	5
REST - nginx	1	3	2	6
REST - NodeJS	3	2	3	8
REST - Erlang	2	3	1	6
REST - NoSQL	2	3	2	7
REST - AMP	3	3	3	9

Appendix B: Application screen idea test results