­AIR Project Transition Document

Prepared by: Consolidated Coders

4 / 17 / 2020

# Project Overview

## Name:

ASUM Information and Referral Project (AIR Project)

## Description:

Jordan Lyons (client) has been working for ASUM for some time, and previously worked in the referral industry where he helped perform benefit referrals the old way, with a phone call from a client, and a filing cabinet full of referral information. It was during this time that Jordan heard of the open source “Open Referral” project, hoping to automate and standardize the service referral procedure for cities across the nation, while also making it more accessible than ever. Jordan worked on the project previously in recent years, but progress on the project was halted in lieu of the team’s other commitments. Now, the mission of ASUM Information and Referral is to connect students experiencing basic needs insecurity with community resources, and to create a model for other service agencies’ resource databases.

The desired impact of the project is to help connect more basic needs insecure UM students and nonstudents alike with services they qualify for and that can help them, thus lowering the margin of people that have needs but do not seek assistance. The secondary desired impact of the project is to become a “one stop shop” for agencies and decision makers to see information about available programs for basic needs assistance.

The means to achieve this goal include creating a database that will eventually contain all pertinent referral information for the city of Missoula, allowing Organizations to create their own accounts to edit or insert this data, and allowing admins (like Jordan) to view and edit this data to ensure its accuracy. Finally, average people will be able to view and interact with the data in a variety of intuitive ways to help narrow down their needs, such as the Needs Assessment Quiz, and a dynamic series of options that narrows the scope of their search to bring them exactly what they need.

## Major Stakeholders and their Roles:

### Jordan Lyons:

Jordan Lyons is the project’s client. As stated above, he has worked in the referral field before, and has even worked with the Open Referral project before this. Much of the project is his brainchild, and his signature and approval was required on all documentation (see ‘Design Documents’). Jordan provided the Airtable database and associated schema that was copied in order to create the database the project currently uses. Jordan (and ASUM) own all rights to the software and will take immediate ownership after the end of the Spring 2020 semester.

### Consolidated Coders:

The Consolidated Coders, consisting of Sean Rice, Kathryn Reese, and Jesse LaFlesch, was the team that implemented the AIR Project as part of their senior capstone class at the University of Montana. The first semester of their senior year was used for planning, documentation, and prototyping while the second semester was entirely implementation, testing, and user interface testing. While they will retain no rights to the project after its handoff, the hands-on experience they received during the planning, implementation, and testing stages of the project is the real value they got, and a fundamental part of their Computer Science education.

### Dr. Yolanda Reimer:

Dr. Reimer was the professor of the class (Advanced Programming I & II) for which the Consolidated Coders implemented the AIR Project. She originally approved the project to be available as an option in the class and assigned the Consolidated Coders to it. Dr. Reimer instructed the team on how various aspects of the software development process should be carried out, maintaining proper client communication, creating an implementation plan, and other key aspects of the class. She also largely oversaw the documentation process and implementation process to ensure adequate progress was being made and kept in regular contact with the client to make sure the team was performing to his standards as well.

### Additional Stakeholders:

For information on additional stakeholders, or more detail on those above, see section 2, *Stakeholders* in the Project Charter document found in the *Documentation* section of this document.

## Status of Requirements:

Due to time constraints, all High and Medium priority requirements were able to be completed during the semester long implementation. However, the low priority requirements were not able to be implemented, and the team instead focused on testing and bug fixing to ensure the delivery of product that serves its intended purpose well, rather than serving a variety of purposes poorly. For more information on what requirements were implemented, by whom, and when, please see the final implementation entry in the ‘Design Documents’ Section.

## Suggested Next Steps:

While the software as it currently stands does function and should be able to function properly for some time to come, it serves almost more as a prototype than as a real project that can be reliably shipped. Firstly, the Airtable database that the project communicates with is still only the free version, and so the number of records that can be stored is limited. Depending on the number of records that will eventually be entered, this may warrant the need for actually purchasing a package from Airtable that would allow more records to be stored, as the team has hit the record limit previously and it would certainly not be good if the maximum records was reached while the site was actually deployed and in use by the general public.

More importantly though than getting Airtable to allow for more records, it is the firm belief of the Consolidated Coders that the Airtable component of the project should be removed. At the time of starting the project, the team was not familiar with Airtable or its API but has since come to realize that Airtable does not offer nearly as much functionality or security as a conventional SQL database. Unfortunately, this would mean ripping out a great deal of the code the team wrote to get the functionality that was needed out of Airtable, but on the bright side an implementation with a conventional database would require much less code than the Airtable equivalent and would be much easier to implement as well. This would give the project the opportunity to have a respectable level of security, as well as fix some issues with the schema and normalization, and finally standardize naming conventions within the database (for the record, the team was not responsible for creating these aspects). As the Airtable API has also been updated during the development of the project, this would also future proof the project and prevent problems down the line caused by changes to Airtable’ s API.

There are also some UI aspects that could use improvement. The Consolidated Coders did the best that we could, but at the end of the day we are somewhat inexperienced with CSS and styling. While some pages and components look good, others are not so appealing and, especially on pages like the Edit Organization Page where space for components needs to be allocated dynamically, the layout can become convoluted. This is especially true when a user changes their window size, or on mobile. Mobile friendliness, while originally a high priority nonfunctional requirement, was never fully implemented and needs to be added still.

Next is the matter of the server. The site is hosted for free on Heroku, which can be slow and or unreliable at times. We would recommend that a paid Heroku server is acquired (and a new, less convoluted URL along with it) or in-house hosting by the UM or ASUM be set up to ensure a better user experience and more server reliability.

Finally, once all of the above is completed, we believe the site would be fully ready for normal use by the public. At this point, ASUM or other owners of the project could begin advertising and recommending its use, or alternatively begin implementing some of the lower priority functionality (see Requirements Specification and Implementation in ‘Documents’), or both.

## Contact Information:

Please do not contact any member of the team with solicitations, message, information, or questions not directly related to the implementation of the AIR Project.

* Sean Rice:
  + [mrseanrice@gmail.com](mailto:mrseanrice@gmail.com)
  + [sean.rice@umontana.edu](mailto:sean.rice@umontana.edu)
* Kathryn Reese:
  + [kathryn.reese@umontana.edu](mailto:kathryn.reese@umontana.edu)
* Jesse LaFlesch
  + [jesse.laflesch@umontana.edu](mailto:jesse.laflesch@umontana.edu)
  + jesse.laflesch@gmail.com

# Technical Information

## Source Code Repo:

The source code for the project is hosted on GitHub. It can be found publicly for clone or download at: <https://github.com/ASUMRenterCenter/air> . ASUM already has access to this repository.

## Additional Assets:

In addition to the source code, there is also the associated Airtable database. In the interest of security, the associated account and password will not be listed here, but the owning account (Kathryn Reese) has given Jordan Lyon’s account ownership access of the database.

The other potentially interesting assets are the original design, testing, and implementation documents, which will be included in the ‘Design Documents’ portion of this document.

## Hosting Information:

Heroku is the current host for our application. <https://www.heroku.com/>. The current web URL for our site is <https://air-proj.herokuapp.com/> and only Kathryn, Jesse, and Sean have access to alter anything. We recommend starting a new account on Heroku so that this can be hosted without the 3 of us. Heroku requires access to the github repository, so when creating a new account, just note that ASUM (Jordan Lyons) will have to accept Heroku’s repository access request.

When Heroku starts, it executes certain commands that we have given it. The first thing it does is start up a local host on Heroku, then it starts up React. We have found that communicating through local host to the database is considered more secure through using POST to transfer data.

## Deploying Information:

If you want to learn how Heroku deploys this, then you’ll have to learn what npm start is doing. Heroku is deploying this the same way as we do. Inside of the package.json file, it lists the steps that are taken based on the command it’s given. There is also a package.json file inside of the client folder. There are multiple steps that need to be taken in order to be able to initially use this application. Here are the steps that must be taken:

1. We each have been using something different, but the fact is that it’s all being done through a linux environment. Learn how to use Windows Subsystem for Linux (WSL) or know how to use Ubuntu and setup a Git repo (or use the current one) with testing branches for each programmer.
2. Install NodeJS and NPM (Google search how to install each, if using ubuntu or WSL search for how to install on ubuntu)
3. With Ubuntu or WSL, navigate to the root directory of this application. Inside the root directory, there should be the following files/folders: client, node\_modules, package.json, package-lock.json, .env, .gitignore, README.md, server.js. If node\_modules and package-lock.json aren’t there, that’s ok. They will be soon.
4. From the root, type: “npm install”
5. Note that stuff happens. If stuff doesn’t happen, or there’s an error, try using “sudo npm install”
6. Then do “cd client” and proceed to do “sudo npm install” once more.
7. From there, do “cd ..” and then “npm start”
8. Stuff should happen and you should see a local host start. Keep this terminal open and open a new terminal. Navigate to the root and then do “cd client” and “npm start”
9. If everything works as intended, a web browser taking you to local host should open. Your development environment is now setup and ready to be used.

Things to note about starting up local host: you must run “npm start” in the root folder **AND** “npm start” inside of the client folder or it won’t work. In the server.js file, within the root folder, is the file that runs the server along with POST requests and how it treats them.

# Technology Stack and Hardware Dependencies:

## React:

The entire site is built using React, and the functionality is heavily dependent upon the React library. It is safe to say that if the React library and its associated functionality were removed from the project, there would be almost nothing left in terms of functionality and code.

## Bootstrap:

Aspects of the project utilize React-Bootstrap and Bootstrap. Removing these components would largely affect the style of the site, as well as some of the functionality. Many of the dropdown menus and buttons were created using Bootstrap, and while the buttons may still function without it, the dropdowns and the functionality within them may not.

## NPM:

The package manager that the team used for this project was the Node Package Manager, or NPM. While we do recommend the NPM for the project, we have tested and had success accessing and editing the project using other package managers – though this is not guaranteed for all package managers. Please note that some packages installed by NPM are not compatible for YARN or other package managers.

## Airtable:

As mentioned above, the database used for the project was an Airtable base. Essentially all worthwhile functionality of the project, such as the Organization Accounts, information editing, the Needs Assessment Survey, etc., needs to communicate with the Airtable API multiple times to function properly. While the routing of pages would still work without it, most worthwhile aspects of the site would not function without the Airtable API unless completely reimplemented with an SQL database.

## Heroku:

As the application currently stands, it is being hosted on Heroku. The site it still functional with a local host without real hosting being needed and could probably be easily hosted with another service. The team would actually recommend this, or at least a paid Heroku account and a different URL.

# Design Documents:

The design, testing, prototyping, and other documents pertaining to the project can be found in the same zipped folder in which this document will be stored. In the event that you are reading this in a context where the original documents cannot be found and need them, contact one of the team members or Dr. Reimer to request copies.

# Technical Debt:

For the uninitiated, technical debt is a concept in design that reflects the implied costs of reworking software that was initially created with the cheapest, fastest, or easiest means rather than the best. Just like real debt, technical debt can gain interest over time, as the more aspects of the project are built upon these less than optimal pieces, the more difficult and costly the rework will become.

The technical debt of the project at this point is difficult estimate, largely in part because the team does not know the average price for most of these items, and additionally, how much time the needed rework would take a team that is actually experienced in these technologies (as opposed to learning as they go along).

Though we do not know the prices, the team can still provide a “shopping list” of what will need to be acquired or done to get the project up professional standards. The list is as follows:

* Development Team that can
  + Rework Airtable interface code into SQL database interface code
  + Set up SQL database
  + Improve CSS and implement mobile friendliness
* Paid Hosting Service

As stated, the team is too new in the field to know the number of people such a team would have, the cost per person, or the number of hours to complete a task. While we could give estimates, they would literally be guesses and we feel it is better to be honest about not knowing than to make false promises that could be detrimental to the future of the project.

# Security Considerations:

One of the main issues with Airtable is that it is blatantly not secure. Even on Airtable’s website, they admit that while it is reasonably secure, it should not be used to store login information. This is unfortunate because login information for Organization accounts does need to be stored in the database. This, as well as the setup of the Airtable API within the code poses serious security risks. The best-case scenario would be someone hacks into an Organization account and creates a fake event with some silly name. However, in the worst case, they could delete all data for that organization, and if the credentials were admin, they could delete nearly everything from the entire database. Even worse, if the organization follows the poor security practice of using the same or similar passwords for multiple platforms, that hacker could then potentially gain access to other accounts as well.

Furthermore, passwords and other account information are currently stored as plain text in the database. Obviously this is a concern on its own, but it is somewhat mitigated by the nature of the site – no personal or sensitive information is ever stored for any type of user. A hacker doesn’t stand to learn anything by hacking that they couldn’t already see publicly on the site, only upload false information or delete information.

Overall, the use of the Airtable database constitutes 100% of the team’s current security concerns for the project, and the reworking of the code to function with an SQL database, while also being necessary in our opinion for the long term viability of the project, would also remediate these concerns.

Regardless, the team would recommend regular backups of the database to ensure that in case of catastrophic failure or deletion by hacker, not all the time spent entering the data will be for nothing. As far as monitoring for false information inserted by a hacker, the team has no ideas given the current Airtable API and capabilities how to effectively watch for such information, short of manually inspecting the contents of the database for their validity.

# Considerations for a Professional Environment:

## Ownership of Data and Accounts:

The largest and most important piece of data in the project, the source code, will be the sole property of Jordan Lyons and ASUM following the end of the Spring 2020 Semester. The data within the database will of course be public knowledge and not subject to ownership, while the schema and taxonomy of the database are already open source. While the team will hand off ownership access to the database the project is currently using to Jordan, the actual personal Airtable accounts of the team will remain in the possession of the team. Jordan, who already has access to the GitHub repository for the project, can restrict the team member’s ability to push to the repository at his discretion.

## Transition Timeline:

Over the course of this project, Jordan has always had full access to every aspect save for ownership access of the database being used. Once ownership access has been granted, which will occur well before the end of the Spring 2020 semester, Jordan will have full control over the project. At that point, he can remove the Consolidated Coders’ access to any and all aspects of the project at his discretion, taking ownership and as much control as he likes. The Consolidated Coders signed away their rights to the project in its entirety at the start of the year and will make no attempt to access the project if their access is revoked.