HANDYMAN ORGANIZER
TEAM 9

OVERVIEW

1. Members
Dominic Pitts (d867p256@ku.edu)
Liam Hughes (JohnLiamHughes@ku.edu)
Tim Elgart (telgart@ku.edu)
Travis Augustine (augustine@ku.edu)
Andrei Elliott (a.d.elliott@ku.edu)
Jeromy Tsai (j001t765@ku.edu)
Jesse Yang (j574y923@ku.edu)

2. Meeting Times
Team Scrum Meeting – Monday, Wednesday, Friday 11 am
Meetings with Professor: 9/26, 10/17, 11/28 11am

3. Project Description
Handyman Organizer’s main role will be to improve the daily routine of members with managerial responsibilities within a company. This includes but is not limited to tracking daily activities through notes and the ability to save and track job history. This job history will allow the user to access all information tied to the item such as location, date / time, materials used, hours billed, etc.

I feel a great opportunity exists here due to the lack of options available to this category of workers. Though there are a few applications that cover a couple of these functionalities, they are bloated with many other tools that only certain professions use. Thus the average user has 10 times the functionality on the application than needed. This causes confusion and frustration for a group of workers who generally don’t spend much time on technology anyways.

Handyman Organizer will provide a basic user interface that is easily navigable while providing the necessary tools that an average foreman or small business construction owner needs. The simplicity and use of this application will allow it to thrive within this scope of users.
4. Milestones

- Project Initial Design
  - Outline of webpage design
    - 11/7/2016
  - Database items needed to be stored
  - Estimated Completion
    - 11/7/2016
- Database Structure
  - All variables covered
  - Easily expandable for further additions to project
  - Estimated Completion
    - 1/21/2017
- Backend Access
  - Ability to save and retrieve data
    - 11/27/2016
  - Communicate with both Database and Front-End
    - 2/11/2017
- Front-End Functionality
  - Ability to create / save Note
    - Location
    - Time
    - Details
    - Estimated Completion
      - 2/25/2017
  - Ability to create / save Job
    - Location
    - Time
    - Details
    - Materials Used
      - Name
      - Cost
    - Hours Billed
    - Notes Tied to Job
    - Estimated Completion
      - 2/28/2017
- Documentation
  - Due one week after implementation deadline.
5. Project Budget

<table>
<thead>
<tr>
<th>Item</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Laptop / Cellphone / Tablet</td>
<td>Already Owned ($0)</td>
</tr>
<tr>
<td>Webstorm</td>
<td>Free for a year ($0)</td>
</tr>
<tr>
<td>Database(AWS)</td>
<td>$0-$40</td>
</tr>
<tr>
<td>Hourly Pay $0/hr</td>
<td>$0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>$0-40</strong></td>
</tr>
</tbody>
</table>

6. Work Plan

<table>
<thead>
<tr>
<th>Team Member</th>
<th>Team Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dominic Pitts</td>
<td>Team Lead, Front-End, Back-End, Database</td>
</tr>
<tr>
<td>Liam Hughes</td>
<td>Database and Back-End Data</td>
</tr>
<tr>
<td>Tim Elvart</td>
<td>Front-End and Functionality, Back-End</td>
</tr>
<tr>
<td>Travis Augustine</td>
<td>Front-End and Functionality</td>
</tr>
<tr>
<td>Andrei Elliott</td>
<td>Back-End Data</td>
</tr>
<tr>
<td>Jeromy Tsai</td>
<td>Front-End and Functionality, Back-End</td>
</tr>
<tr>
<td>Jesse Yang</td>
<td>Back-End Data</td>
</tr>
</tbody>
</table>

7. GitHub

Presentation Layer

- [https://github.com/jjh88/HandyMan_Presentation_Layer](https://github.com/jjh88/HandyMan_Presentation_Layer)

Business Layer

- [https://github.com/jjh88/HandyMan_Business_Layer](https://github.com/jjh88/HandyMan_Business_Layer)
8. Preliminary Project Design

Handyman is being implemented by attacking two fronts at once. To make this program be as scalable as possible, we are implanting a Front-End and Back-End separately. This attack allows us to design a Back-End that can be accessed via many different Front-Ends such as Web, Android, iOS, etc. What becomes important with scalability is creating our Back-End in a way that items can be pushed and pulled easily and quickly. As well as the flexibility to create new columns in the table to store new information that wasn’t previously stored. A diagram to show the connections and thinking behind this implementation can be seen in Figure 1.

Figure 1.

The Frontend is equivalent to the Presentation Layer. Backend is comprised of the business layer and Data Layer. All filtering of information should be done in the Business Layer so that adding Android or iOS app will be just a matter of displaying the information. AJAX allows calls to the API to be made and will wait for the response. JERSEY is a JAX-RS implementation that will allow us to make a RESTful web service. Every method that can be called by the Presentation Layer will have a URL. Making a call to the URL/parameter will allow us to call that method using the parameter included. Setting the system up like this will allow us to work on various portions at the same time and then get the required parameters once the methods are completed. Some things to be implemented in Business Layer are:

1. Retrieval and Storage of Notes, Locations, and Job History by User.
2. Retrieval and Storage of Notes, and Job History by Location.
3. User Creation and Verification
4. User Login
5. All SQL scripts will be here

Presentation Layer:
1. Control views for the user. (Possibly use Handlebars or something similar)
2. Make calls to the Business Layer
3. Parse responses and display to user.
4. Have proper fields for all information.
5. Display graphics and logo for HMO

Data Layer
1. Users and Locations should have unique IDs
2. Locations, Notes, and Jobs should all have USER_ID column
3. Possibly add Company ID that is also associated with user so multiple members of same company can access same data.
4. Notes and Jobs should have LOCATIONS_ID column

This is not a complete list of what will be done. Adding in companies inventory data adds a large amount of functionality.

With the point of this project being quick and easy. We need fast and efficient code and a very simple UI. The User Interface is arguably the most important aspect of a project, with speed and efficiency being a close second. The reason the UI is important, especially for this project, is because our user base is everyday workers who either don’t use electronic and web apps every day or don’t have the time or patience to constantly be messing around with their device while at work. Due to this outlook, the idea behind this UI is bigger is better. We want our User experience to be fast and easy. This evolves a few selections of buttons that will be used constantly, as well as making them larger and clearly named allowing one to quickly locate the action they want to perform in a timely manner.

To obtain these traits, we will deploy a Java rest web server that will perform all connections with our database. This will allow us to use the front end for purely display purposes and increase performance. As can be seen in Figure 1 above, there is a separation of responsibilities. The front end will send requests via jQuery ajax and
access functions on Java server using the Jersey implementation of JAX-RS. The information will be returned in JSON format and be parsed in the front end. Inventory data is shown on the figure but that is a stretch goal that we are not sure if we can implement yet.

Now, bringing us back around to our UI, a simple UI outline can be seen in Figure 2. The final product will look very different than Figure 2 while also keeping the general structure and theme of usability. The concept of this setup is having the most used functionality easily accessible. This can be seen on our Home tab, the two large buttons that place the top two functions used in your face, also providing a google maps view of all old Jobs and Notes that contain saved locations.

Figure 2.

The tabs allow for the none everyday functionality, such as editing your profile and checking job history. These functions, while important, aren’t expected to be used as frequent as the functionality placed on the Home Tab. The tab’s job is to allow the user the ability to edit and easily control what notes and jobs are saved and stored, giving the user control of their own application, though keeping the base functionality quick and easy to use.

Figure 3 below shows the starting point of our database. The current design is based on having a company be the creator of both users who would be its employees and jobs that its employees will then take on. After creating a user, that individual would be responsible for creating their own password. After that they can select a job from their company’s job list. The company jobs list will have unique ids and when entering an address, the address will match with the location table to retrieve the location id if it exists and it will create a new one if it does not. Multiple users can select the same job from the company job list and all will be stored under the second job table. Once a company declares a job finished, the job status will be switch to F. When a user takes on a job the job status will be P for in progress. Job status will default to N for not started. As the location table is, multiple locations can be made with the same id so that users can filter by locations they have worked on. This may be better implemented using a company based location table or if locations themselves aren’t as important because users are already going to be able to get their job history which will contain location ids. Notes will be stored by jobs that the user is working on with user, company, job and location ids or at any time but will not be populated by a job or location id. Later, the notes will be editable to attach them to a job at which point those fields will be updated. The diagram shows info populated from other tables in text above the arrow.
A billing system is also planned. To get proper amounts, a materials table and a bills table for the company will be created. The interactions can be seen in Figure 4 below. The diagram shows actions straight from front end to backend for easier visualizing but the requests and responses will always pass through the Java web service. Once a company marks a job as finished, the Java service will collect the hours worked by employees, combine that with their hourly wages, and sum that with the material costs to get a final total. Then this value will be stored in the company’s bills table along with a default payment status of false.

Figure 4.
9. Ethical and Intellectual Property Issues

Intellectual Property: Trademark Infringement

Trademark Infringement is defined to be the unauthorized use of a trademark. When designing the logo and name for our product, we must take into consideration the possibility that another company has previously used an identical or highly similar logo. We must design a logo that minimizes confusion and deception between our product and previously trademarked products.

There are two primary factors in determining whether a trademark is infringing upon the intellectual property claimed by another company. The first factor in question is called the “likelihood-of-confusion” determination, and is a primary factor when determining if a design is infringing upon a previously claimed trademark. A secondary factor in this determination is called trademark “dilution.” Essentially, trademark dilution occurs when a similar, but not identical trademark is used. This similarity would weaken the strength of another company’s trademark.

Ideally, a completely unique trademark should be created, then compared to trademarks of similar products to ensure that no confusion can be made between the two products. An alternative to creating a unique trademark would be to use a copyright-free logo generator, since these trademarks would, by design, not be infringing upon any intellectual property.

Source: https://www.uspto.gov/page/about-trademark-infringement

IP Issue

A goal of HandyMan Organizer is to create an open source map containing the electrical and piping within a region. To do this when users add a new spot, that location will be available to anyone else using the app. Some of the information stored that relates to a location such as types of inventory used as well as notes about that location may be IP of the company that stored the information. We will have to discuss with users what information they are willing to share and maybe provide an opt-out system to make not as much information available to other users. If we make these features company or user specific, then the app will suffer for them and others as maps will not be as populated with useful information.

Ethics: Ensure Secure Authorization

We will be creating a service that users will need to make accounts for and log in to. We must ensure that our login process is secure, so that malicious entities can’t impersonate our users, or steal their login info and get into accounts our users have with other services. Since none of us are security experts, we will ensure we are being as secure as possible by using vetted, third-party, open source libraries to correctly hash and sort users’ passwords to store in our database. This needs to be fast, efficient, and secure to allow quick and easy usability while also giving the customer peace of mind that their data isn’t being stored insecurely.

Ensure adequate testing, debugging, and review of software and related documents

Handyman Organizer will be made for use by small construction company owners. Since business owners may be relying on Handyman Organizer in the future, it is important to make sure the product is stable, and unlikely to cause problems for the business owners. It would be an unfortunate incident if a company were to lose all of its job locations, notes, and other data stored in the application due to a bug or specific state in the code we as developers did not see or fix due to inadequate software testing and debugging.

Ethics: Maintain Data Integrity

Effective information distribution and maintenance plays a large role in successful management. Our goal is to create a system where most of a company’s daily routines can be outlined in notes and their job histories can be stored. Clients must be able to easily and reliably access the information contained on these notes from our backend.
and databases. In order to attain this goal, proper maintenance for the data stored by clients must be satisfied and the ability to promptly relay this data back to them must also be achieved.

Accuracy of data is the most important quality for information stored into our databases. Data representation within company management may largely impact their decisions. Many operations on our databases could potentially compromise the security of this data which we need to prepare for. Major responsibility lies amongst those who have direct access to our databases such as our teammates. When coding the backend, we must ensure that the information processed from the user comes in to the databases unaltered. We must ensure that no other processes can modify or corrupt this data when they are not supposed to such as programming errors and security breaches. Data encryption, backup, and validation are all methods we will employ to ensure the data we access is consistent.

Ability to transmit this data between user and database is equally important. Efficient use of our service depends on the information which must be correctly returned and its reliable transmission. User input must be valid and checks should be made to ensure this. Checks must be done consistently to ensure integrity in our databases. The backend must efficiently retrieve data from the database and output back to the user. Thus, our databases must also have effective upkeep. Additionally, certain information should be disclosed only to certain parties that are granted access.

Confidentiality should be preserved during clients' usage of our service. No company should have unauthorized access to another's data. Username and password credentials will be used as validation to determine what data is accessible to the user. Specifically, items such as location, date, materials, hours, and billing should remain confidential to that user. The client should always feel their information is secure in our databases.

Source: https://www.veracode.com/blog/2012/05/what-is-data-integrity

2.01 Act Within Areas of Competence

Being students that have only been briefly exposed to computing languages and webpage development, it is important not to try to overstep our bounds as it is important not to create an application that may cause problems with the customer’s computer. With the languages that are being used, we shouldn’t have to worry about garbage collection as Java already does that. This will help with memory management, but being new to JavaScript and AngularJS we need to be careful in the sense that we want to write code that is both fast and efficient. This involves avoiding overly complex code that can become unpredictable as well as using inefficient methods in regards to retrieving and sending data.

10. Change Log

Though it wasn’t specified above. The change made recently is that the web page being created will be a single web page that uses AngularJS to change the views via tabs instead of buttons that redirect to a different page.