James Baron

Intermediate presentation 1

**Introduction**

* Hi everyone, I hope everyone is staying healthy and is doing well
* Before I get into the specifics, I would like to go over what I will be covering in this presentation.
* First, I will discuss the background of my project. This will include how I came up with the idea and all about the final product.
* Next, I will discuss the progress that I have made up until this point. I will be showing the entity relationship model of my database, my web service script, and the Login screen User interface of the IOS application.
* And finally, I will discuss what will be expected to be completed by the next milestone.

**Brief Background- BuildIt**

* I am building an IOS application that is designed for construction companies to manage their employees more efficiently.
* I got this idea from working summers for my father at his construction company. While working for my father, I noticed how unorganized and inefficient work was being conducted on jobsites.
* These problems stemmed from all communications and decisions having to be made by my father or head of company.
* This would cause employees to sit around and do nothing until told what to do and mistakes to be made because of miscommunication over the phone.
* BuildIt is going to solve those problems by allowing the owner to create and assign tasks for each employee.

**Progress made**

* So the good news is that I am on track according to my proposal.
* Since I had zero experience in IOS or any app development I spent the first two weeks learning the essentials of XCode and swift.
* At this point I promised a completed database for this presentation.
* I completed the database and some more.
* This is entity relationship model of my database.
* Employees has a one-to-many relationship with task, signifying that many tasks can be assigned to one employee.
* Tasks have a many-to-one relationship with jobsites, indicating that many tasks can be assigned to one jobsite.
* Jobsites has a one-to-many relationship with employees, meaning many employees can be assigned to a jobsite.
* The employee’s entity has 6 fields: first name, last name, username, password, admin, and phone number.
* The First and Last name of the employee are the primary key of the entity. Originally, I chose to use the username as the primary key because that would have to be unique. This was problematic because I wanted employees’ names attached to tasks and jobsites, not their username.
* The username and password are self-explanatory because they will be used to login into each account which will provide them with their personal tasks.
* Admin is a Boolean value that will be used to determine if that user will have access to adding and creating employees. The app will determine this when the user logs in.
* Finally, we have the phone number which will be available to look at by all other employees
* Jobsites- The address is the primary key because every jobsite will have a unique address.
* Jobsites will be a page on the app that will display the address, a picture of the site and the employees that will be working there on that day.
* Finally, we have the task entity. It has 5 fields: Number (unique identifier for the task), Details, Jobsite, First name, and Last name.
* The number field identifies the task. The details contain all the information needed for the employee to carry out the task. There is no real format for the task so it will be up to the owner to determine how much information is provided.
* The jobsite field will link the employee to a certain jobsite. Its important to note that this field is optional because some tasks will not be completed at a specific jobsite. An example of this would be going to the window store and getting a window.
* The first and last name are how the tasks are linked to the employees

**Web service**

* This is the php code that I am using to test connecting between the IOS app and my sql database.
* This is not my final code, just what I am using to test the connection.
* This gives a good idea of how the php code interacts with my database
* This picture shows the employee struct in swift that will contain the information taken from the database.
* I have not yet finished connecting the app and the database but should be coming soon.

**Login screen UI**

* This is login screen that I have been working on. Users will type their username and password in the fields given.
* Once they tap the login button, the swift code I wrote will validate the user and load their personal tasks or will give an error message indicating that their username or password was incorrect.
* This screen shows the code currently running for the login user interface. The code is missing the username and password taken from the database but gives an idea of how I will be validating the credentials.

**Conclusion**

* For my next intermediate update, I want to have all the user interface completed.
* This means that the tasks, employees, jobsites, add employees, and create tasks user interfaces will be complete and able to move about them.
* I do not expect the backend to be complete because I am sure I will run into some complications.