

Resume Sandwich Web Application

By Joshua Hutcheson

Spring 2019

# Introduction

One of the most time-consuming aspects of job hunting is developing a resume. Resumes display the key aspects of an individual’s experience with the goal of showing them to be well-rounded to employers. Every position requires a specific set of skills, which means each time a person applies to a new position, they have to take the time to update their resume to fit it. The process becomes very tedious, and if not using the right tools, can lead to formatting mistakes and inconsistencies within format. These are not matters that someone looking for a position should have to deal with, so this project focuses on finding ways to lift that burden. I, Joshua Hutcheson, met with Dr. Wilson on Thursdays 11:15am and communicated through email about guidelines of the application.

## Purpose

The purpose of this project is broken down into several goals, which are applied to the scope:

1. There are several factors that contribute to making the resume building process difficult and this project would like to uncover the general main areas of difficulty through user studies.
2. Based on requirements, this project will focus on building a functional application that has the soul focus on walking through building a resume.
3. The application built will have to provide the ability to work through different “blocks” of information rather than just a one per topic constraint. This will allow users to put in multiple experiences under one category. In order to do this, a framework will need to be developed to carry out this process with ease.
4. When finished, the application needs to have the capability to export the file in to a useable format, such as a pdf. This will require a specific set of tools, potentially from an open source project to implement.

## Requirements

Based on the purpose above, the requirements are set to change based on the requirements gathering process (see below). For now, we will need to define the following:

1. Types of user studies applicable to the project
   1. Interviews – these can help break down basic issues with resume building and form the first and abstract level goals for the application’s design.
   2. Card Sorting – this type of user study will be modified to fit the purpose of replicating a version of the “modular” resume building process. The term “modular” will be used throughout the project as the way in which the experiences are added.
   3. Focus Group – a group of 3 will discuss their thoughts on a resume template and figure out what are the pros and cons of it based on prior professional experience.
   4. Prototype walkthrough – this will allow users to work with a version of the application with accomplishing tasks and trying to see how well they can through the layout.
2. The application will need to pick a particular design pattern based on the technologies decided to use for the project. After careful research, the frame work the project will be using is a variant of Model-View-Controller using Angular 7 for the front-end and creating a fully functional API using Java Spring Boot for the backend. The layout of the dataflow will need to be created to help guide the setup of the application and an entity diagram should be developed to uncover what each module should contain.
3. A resume template has already been created for the project based on university approved advisors and the creator of this project. The template has been in development over two years and has helped numerous students through their job applications on Hire-A-Niner. An outlined copy of this template can be seen in the Appendix Source A.
4. The theme of the application will be based on a “sandwich” to put emphasis on the modular building aspect of the project. Sandwiches contain a variety of elements to make it a whole, just like a resume. The key is simplicity, so the entire application will be using a single page to navigate through the process.

## Scope

This is the project scope which lists out the main objectives of the project. Anything stated beyond what is listed below is beyond the scope of the project:

This project will conduct user studies involving interviews, card sorting, a focus group and prototype walkthrough to understand the needs of the project. A resume template will be used based off of what was already developed and critiqued by professionals. The application will feature a dataflow diagram and an UML entity diagram to capture the design plan of the software. A back-end API will be created to capture the modules of the software and organize it hierarchy in the resume using Spring Boot. A front-end interface will assemble the user experience on a single page application using Angular 7.

# Requirements Gathering

## User Studies

User studies are an effective way of gaining information from everyday users to find out what their perception and behavior around a particular topic is. For this project, we will be testing users understanding of the resume building process both from a knowledge and experience standpoint. From this information, we will draw our conclusions and implement a next steps for further defining our requirements. Note that only suggestions based within the scope stated above will be implemented. Below details the results of the study, but much further details on these studies can be found in the User Studies document.

### Interviews

#### Analysis

After looking over the interviews, there were patterns in the answers that people gave. People thought the hardest part of building a resume was starting it or following the right kind of template. A way of getting help building resumes using some kind of walkthrough showed up as a general response and have it suggest what skills to use. Generally, they said that the resume is never done or that it is done when it fits with what they are applying for.

#### Conclusions

Some takeaways from this study is that the app should focus on guiding people through building their resume. The template design should be applicable for different kinds of applications, not strictly along the lines of a particular field. Blocks on resumes should be saved so users can potentially re-use them on other resumes if they need to. Later, a recommendation API can perhaps be implemented to take those values and suggest them to users based on some kind of input suggesting the king of resume they are trying to build, but that is beyond the scope of this project.

### Card Sorting (Testing “Modular” Resume Building)

#### Analysis

Users ranked the order of education, skills, work experience, college engagement and volunteer experience as they saw they were key. Some users who made this format suggested putting college engagement over work experience. They stated this was because they were in college and that employers had told them before that college engagement showed that they went out of their way to do things in college in addition to work and classes. One user suggested adding in a “Projects” module. Employers like to see projects, so showing descriptions of projects and what a person’s role was on the project could be helpful to the employer in uncovering more about an applicants skills.

#### Conclusions

This showed users generally had the same idea on how to put together a resume, with one suggesting an additional module. Based one these results, it would be best to keep the general format flow, but also include a “PROJECTS” option.

### Focus Group

#### Analysis

A group of 3 people began to discuss the resume format, the same as above. They talked with one another and gave feedback. Ultimately, it proved to be the least effective of the user studies as it really only lead to an explanation of the order and answers more based on subjectivity of particular hiring managers rather than industry standards.

#### Conclusions

It appears that these answers were based more on personal opinion of particular hiring managers that the participants had spoken to rather than industry standards. The main conclusion that can be drawn from this is to make the app be able to include and skip tabs in the application.

### Prototype Walkthrough

#### Analysis

The users were prompted with tasks, pre questions and post questions. They gave insightful feedback saying it would have been nice to have a forward and back button rather than a navigation bar so not to be confused on which step they were on, but overall found it to be helpful.

#### Conclusions

Overall, the responses were quite satisfied with the design of the application. It’s theme was enjoyable and was good for the context. The conclusions to draw from the prototype is that it is pleasant to navigate with but would prefer arrows and options to reset fields.

## Overall User Study Conclusions

These helped to define some key aspects of the application’s design and user experience. It was helpful and insightful to learn from what users had to say and how they behaved. Using this information, the application requirements will be further defined. As stated above, a more detail breakdown of these studies can be found under the User Study Design document.

# Design

## Prototyping

### Low Fidelity

*Resume Sandwich Form*

The prototype of the Resume Sandwich app showed the most basic abstract design of the application, which would help to provide an idea of what the actual layout would look like. It did not feature colors or specific logo designs. Further development of the application changed the buns which looked more like hamburger bunts to sandwich buns to make it more general. The input of the values were ultimately not used as a sandwich stacking due to difficulty in the implementation stage. It did however match the overall placement of the inputs and the side/top navigation were designed in the format for future developments (Appendix, Source B).

## System Design

The system design consisted of careful studying of diagrams from other application frameworks and breaking down the resume itself. The key was to find the entities needed to populate the fields of the resume and to be able to transfer the data to and froe correctly between the Angular program and the backend Java Spring Boot API.

### UML Diagram

The UML Diagram defined entities and were derived from the resume fields already created. Each module would represent a different bean entity in the Java Spring Boot API. All attributes would be fields within the resume itself. Defining the attributes can be seen in the Appendix Source A. The resume entity would hold a one-to-many relationship with all the other module entities so that there can be many modules to one resume (Appendix, Source C).

### Dataflow Diagram

Next, it was key to define how the data would transfer between the API and the front-end view program. This was done using research from all kinds of different applications the used either Angular or Java Spring Boot. It was pivotal that it remain in the MVC design pattern to keep with consistency and abstraction. This was carefully compiled and was used as an ultimate map for the dataflow of the application. Though the application itself will work as a single page application, the dataflow is where the complications come into the design (Appendix, Source D).

## Conclusions

The diagrams proved substantially crucial during the implementation process as this was a very complicated setup for transferring data through the application interfaces. With these, the prototype, and the user studies complete, the application was ready to begin the implementation phase.

# Implementation

For this phase, there were several programs installed and used to develop the application. For the IDE, Java EE IDE Eclipse was used to develop and run the spring application. Angular was used by node JS and was started using ng serve. Postman was used to track the responsiveness of the API endpoints.

## Model

### Entity Beans

Entity beans were broken down into each of the required modules and one for the resume itself. The relationship to each module to the resume was many-to-one. Using Java Spring Boot, the key was to make each of these entities hold URI. The common Spring modeling approach refers to entities as “resources” and each resource contains a URI. A collection of resources is called a domain. In our context, the domain is the resume itself and each of the module entities are considered the resources. An example in the context of a “job” (later called workExperience) to “resume” is as follows:

Domain>Resume/Resource>Job/{Add job, Get job, Update job, Delete job}

Similarly, API Endpoint is:

Collection Resource that returns a list of resources (note v1 means version1): /resume/job/v1

Singleton Resource that returns a single resource: /resume/job/v1/{id}

To later implement this into the URI, each entity would contain a constructor with attribute fields as well as getters and setters for each attribute.

#### Entity Requests

Entity Requests are identical to the entity models themselves. They contain getters, setters and a constructor. These will be used to first take information from a user and put them into an entity.

#### Entity Responses

Entity Reponses are identical to the entity models themselves. They contain getters, setters and a constructor. These will be used to first take information from the model entities and export them out to the user.

## Controller

### Java Spring Boot

#### General Setup Goals

Using the same example as above for creating a new job, the REST Controller would have to create a JSON collection in the following way:

Create a new job

POST: /resume/job/v1 also required headers and request body

This POST is in JSON format and must include the ID, start date and end date, company, title, skills

{

“jobID” : 1,

“startDate” : “May 2017”,

“endDate” : “June 2018”,

“company” : “Harris Teeter”,

“title” : “Barista”,

“skillsA” : “Created drinks for customers”,

“skillsB” : “Wrote orders”,

“skillsC” : “Helped managers”

}

#### Converters

Converters take the getters and setters from each request, response and entity and transfer information between each one. It evokes the converter library and the each is loaded into the converter configuration file. Each is context specific based on what the resources call.

#### Resources

As defined earlier, resources are what take the entity values and transfer the information to and froe. They have GET, POST, PUT and DELETE methods implemented in them. They use the converters to take information transferred and place them into the entities themselves or to take information from entities and move them out as GET responses. In our context, GET would be used to display resume information on the page, POST would be used to add new module information, PUT would be to edit the resume (outside the scope of this current project, but available for implementation later) and DELETE to delete modules (also outside the scope for this current project). To these endpoints, Postman was used to perform requests based on the URL needed for each action.

#### Repositories

Repositories were used to guide the process of carrying information to the controller from GET requests. They are defined as interfaces in order to server the purpose of defining functions to pull information. Only the FindById function was implemented into the interface of each entity.

## Views

### Page Design

#### Resume Sandwich Page

The design of the page was heavily based off of the paper prototype and was implemented using basic CSS and custom SVG logo designs. The colors are a simple black and white format to give off a simplistic theme just like the rest of the application. There wasn’t a lot of special features implemented using CSS as to not detract from the focus of the resume forms.

### Angular 7

### Observable Components

Observables are Angular’s way of taking user input and applying it to data. The term for this is called “subscribing”. Each form will have an observable that is subscribed to a corresponding data attribute and fed back to the API as a POST request. This was the key reason that Angular was picked as a framework for the project as it can allow for multiple forms to be submitted in an organized fashion and carefully take in values.

To actually activate the application, the NPM commandline would need to navigate to the following directory, followed by the angular starting command ng serve:

cd Documents\Resume Application\Application\resume-sandwich-application\resume-sandwich-application\src\main\webapp\angular2

### Imported Tools

#### MatStepper

MatStepper is a library made for Angular that allows a user to step through a form or series of forms on a single page. This was selected to keep the single page aspect of the project at the center. Each page would allow the user to fill out a different series of attribute for each module until finally the last one is filled out triggering a refresh displaying the updated resume.

#### Html2canvas

This was used in aiding with exporting the formatted values from html to pdf. It is under the MIT license and is free to public use.

#### JsPDF

This was used to make the actual export of the resume to a PDF readable format. It was calculated to allow for multiple pages. The only issue that it ran into was the file size, which could be due to a variety of reasons. It is under the MIT license and is free for public use.

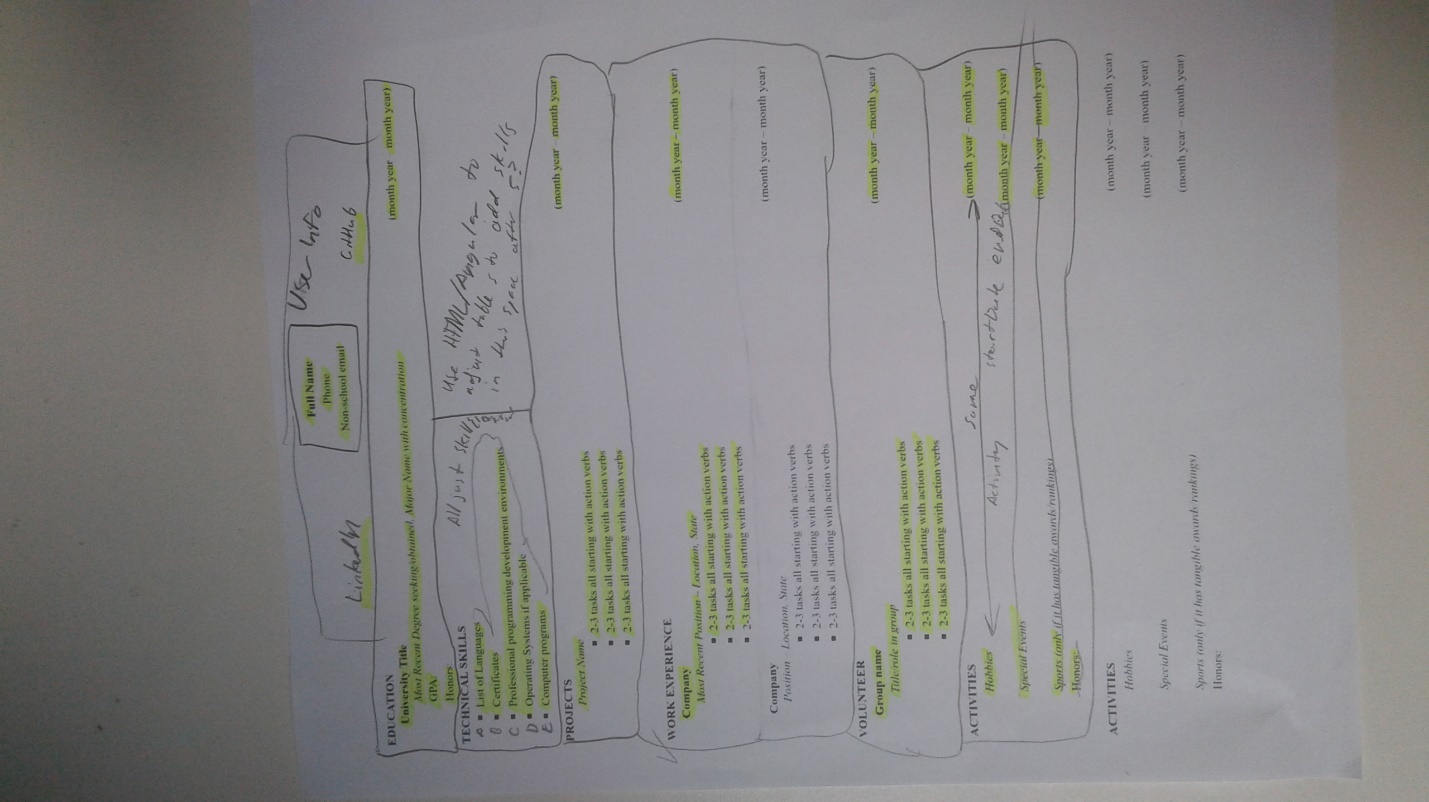
# Conclusion

## Lessons Learned and Next Steps

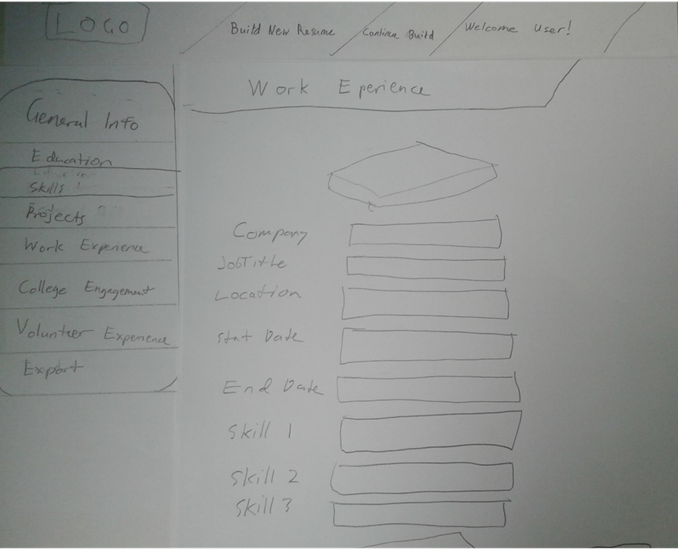
The overall implementation was very difficult and unfortunately not as much time was planned than what was actually needed for the project so a lot of potential features had to be put on hold. Issues arose quite often with sending data between the two applications. Ultimately, the design was created and the application functions with it’s basic premise, which was all the scope was asking for anyway. When the project is picked up, the next steps would be to add more flexibility in the application, finding ways to make exporting the pdf not as large, and perhaps developing the look and feel of the application more.

# Appendix

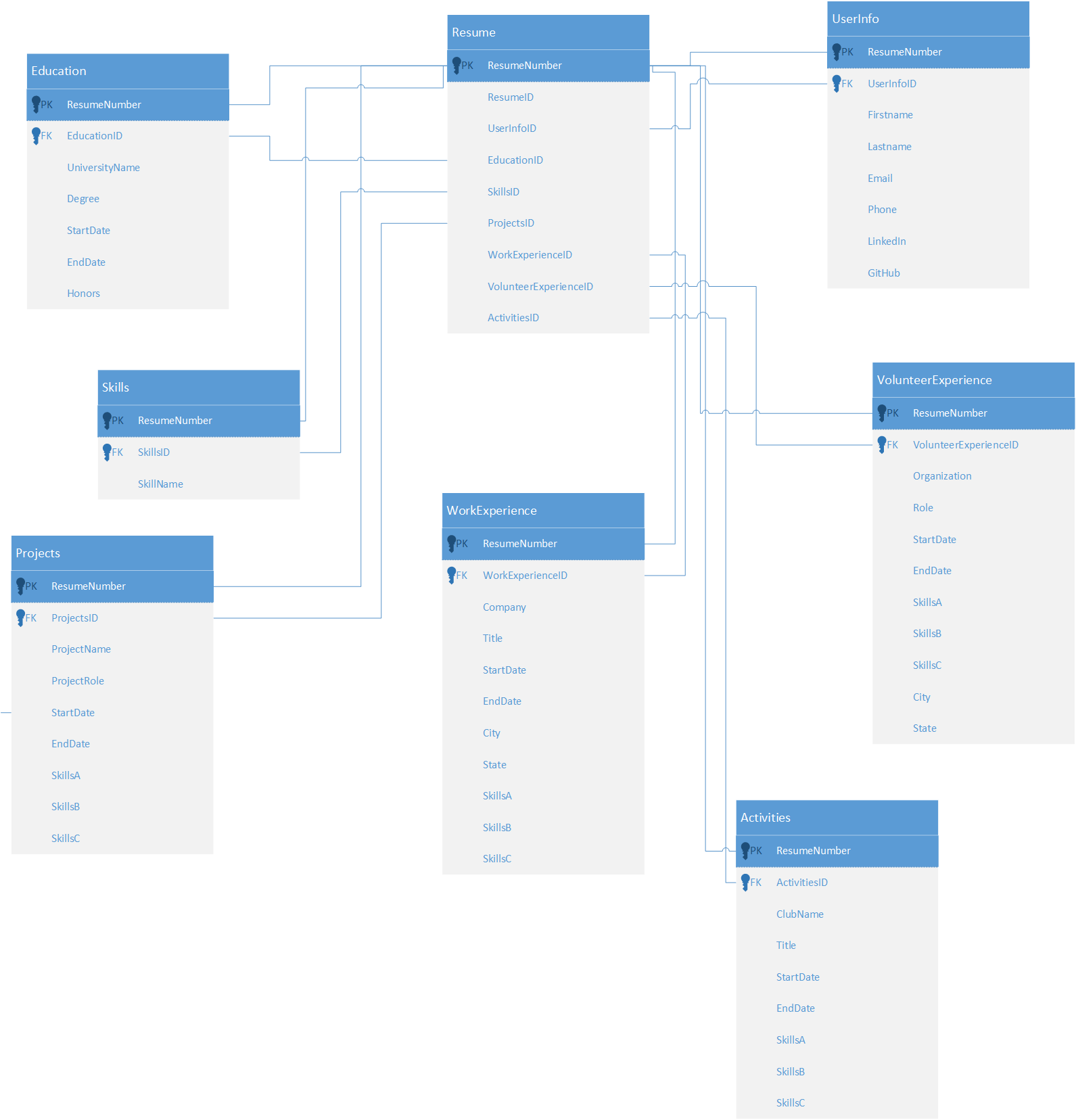
Source A – Resume Template



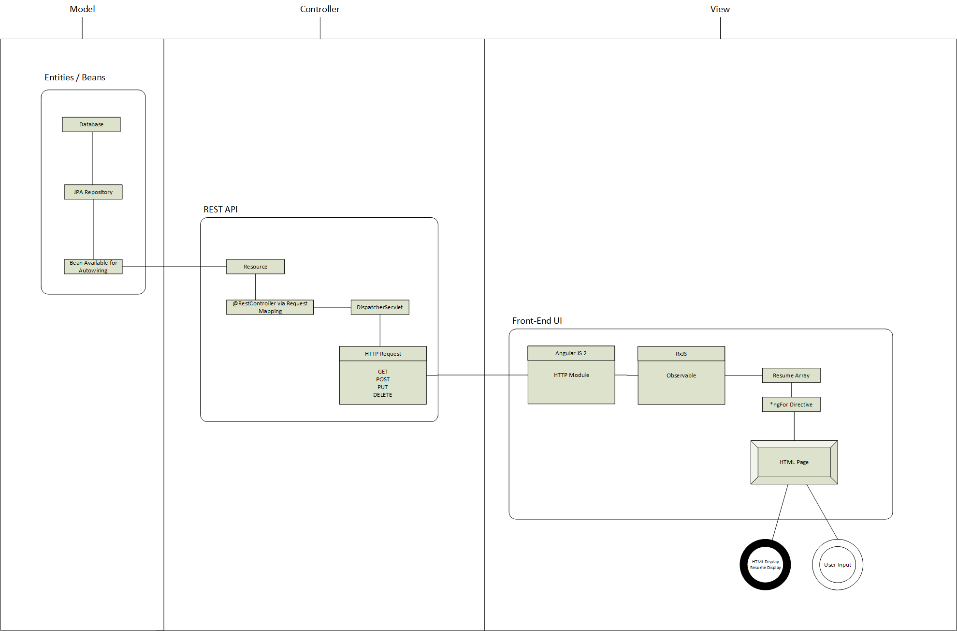
Source B – Low Fidelity Prototype Work Experience Page



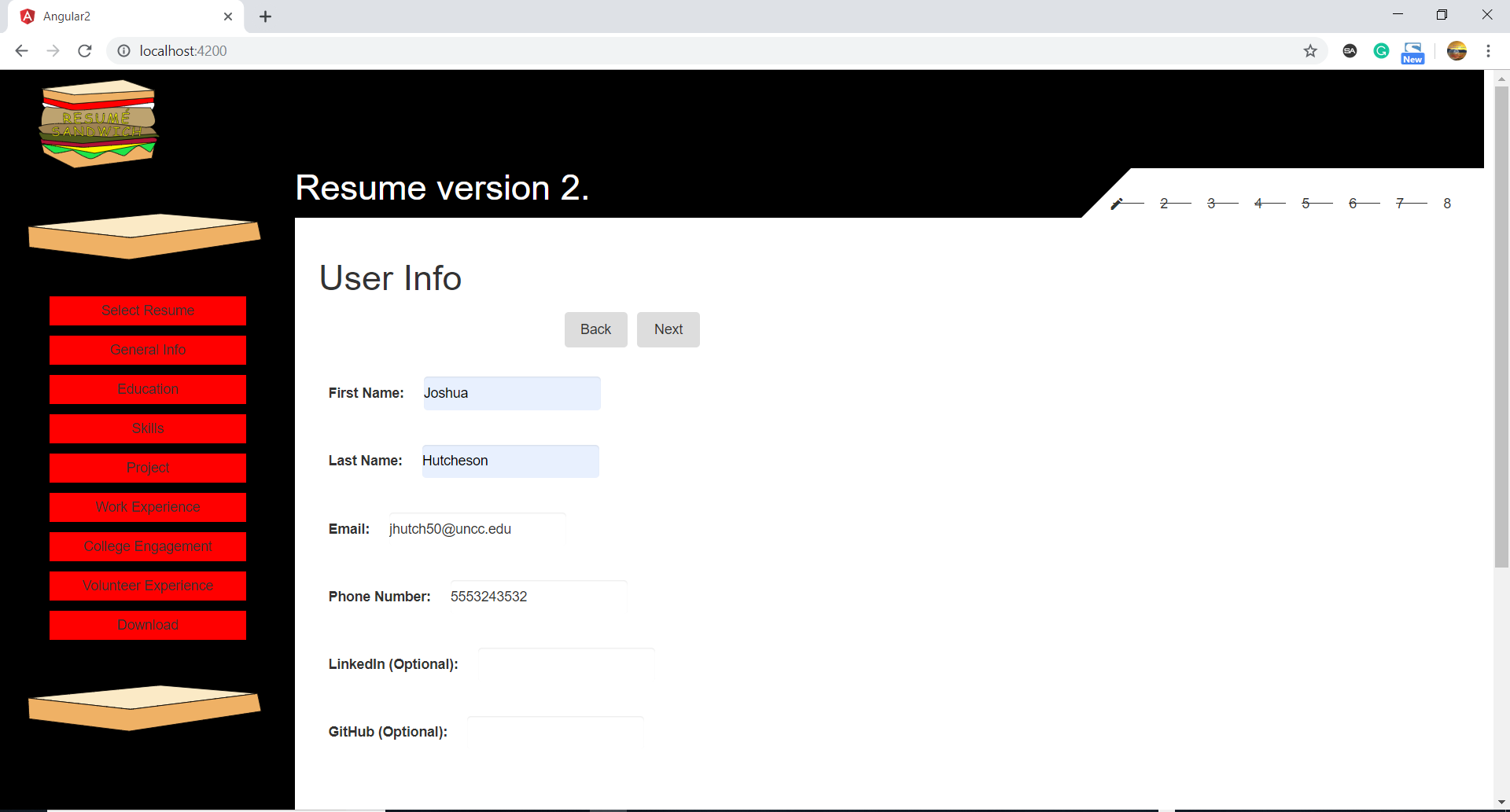
Source C – UML Entity Diagram



Source D – Dataflow Diagram



Source E – Final Site Look “User Info”



Source E – Result

