**System Design Document**

**JobSearch app and service**

**autumn 2019**

**Table of Contents**

[Introduction 3](#_Toc401238542)

[Purpose 3](#_Toc401238543)

[System Overview 3](#_Toc401238544)

[Design Constraints 4](#_Toc401238545)

[Roles and Responsibilities 5](#_Toc401238546)

[Project References 5](#_Toc401238547)

[System Architecture 6](#_Toc401238548)

[Database Design 7](#_Toc401238549)

[System Security and Integrity Controls 9](#_Toc401238552)

# Introduction

This System Design Document has been created to outline the proposed system design for new app and it’s administrative tools to help the author effectively look for the job as a frontend developer. I’ve thought, that it would be handy if I could put all the answers on pretty typical questions, such as medical insurance, project info, company profile, and others, in an app of some kind, and be able to compare them in a table and/or diagram of some kind. All of this instead of using plane pen and paper, putting some interesting project in my portfolio meanwhile.

# System Overview

JobSearch product will provide the following capabilities:

* Register answers to typical interview questions with pre-composed dictionaries of options.
* For questions, that may or may not need some additional comments, one can provide such comments, that will be displayed during the comparison phase of Job Search process.
* For open ended questions, such as “Could You please tell me more about project I’m about to work on, when and If I get a job-offer”, product will provide a textfield to put answer to
* App will also allow it’s user to write down questions of employer one wasn’t able answer to. These questions may be used later to form a kind of a learning program to get better before next interviews in line.
* Desktop part of the product will give user a ways to create dictionaries of options, used later during the interviews, as well as company’s profiles, and other research-heavy datam that will be used later in a mobile app.
* Desktop part of the product will allow user to compare several interviews, to decide which offer user should accept.

# Design Constraints

I’ve identified some constraits before developong the product:

* Mobile part of the product will be done using “offline first” ideology.
* There will be no backend or web-deployed part of the product. All CRUD logic and data storage will utilize MogoDB Atlas BaaS.
* I’ll use Ant Design system for visual representation of all components.
* Product will be developt using monorepo repository structure
* All reusable code should be put in separate packages, maximising SOLID and DRY principles
* I’ll use Electron + React + Redux-dynamic-modules+Redux-saga for desktop app
* I’ll use ReactNative for mobile app
* I’ll use NodeJs+Express+Mongoose+MongoDB Stitch for backend
* Since there is no MongoMobile solution present at the moment, I plan to use ReactNative AsyncStorage to use phone’s local storage capabilities for “Offline first” principle

# System Architecture

**Hardware:**

I’m basically do this product for my sole purposes, so there will be Linux build of Electron app, and Android version of a mobile app.

I’m pretty sure, that MongoAtlas Sandbox clustr will be more than enough for all backend logic I need.

**Software:**

**Electron App:**

Main goal is to do all research-heavy and input-heavy actions on a desktop computer, while preparing for the interviews. Mobile app meant to replace paper and pen, and to provide CRUD for the interviews, and that’s it. Furthermore, the less I have to type in my phone during the interview – the better. I’m quite terrible in this.

* Dictionaries – in this part user will CRUD dictionaries. Under the “Dictionary” I mean a list of options I can put into the <Select /> component in Mobile App, so during interview I can simply choose one of the options for the answer. App is to have the ability for agile modification of the dictionaries.
* Companies – in this part user will CRUD companies profiles. Later on, during the actual interview, I’ll have to just pick the company to start interview in, and that’s it. All company’s information, like reputation, time to commute, and other info, non related to actual vacancy, must be set here on desktop app.
* Unanswered questions – the part, where all questions, unanswered by me during the interview, will be put at, and form some kind of an education plan later on. I think of a list of some kind, that have the dunction of cross-and-hide questions, much like Wunderlist app (maybe do some Wunderlist integraton later on)
* Job Offers comparison module – main valuable part, where I can compare several job offers, head to toe, prsenting all of their respective fields on one screen. Salaries, time to commute, pensions, project details and so on,

**Android App:**

* CRUD operations on an interviews entities
* Fill out question answers on an created interview (Update operation in fact)
* CRUD of unanswered questions during the interview

And that’s it!!!

ANY other features of an Android app are strictly forbidden.

# Database Design

I’ll use MongoDB for a product database. And I’ll use MongoDB Atlas for BaaS.

I’m gonna need these entities:

* Interview – basically a list of links to answered questions
* Company
* Dictionary – list of options to select from, distincted by title
* Question – contains interviewId, dictionaryId, selectedOptionId (one from this dictionary), title (displayed in Android app and during the comparison), and Comment field. Each question will be created when Interview will be submitted onto the backend. The idea is to assemble the Interview entity for comparrison dynamicaly from database instead of hard-recording data in database. But I might reconsider this later.

# System Security and Integrity Controls

As the last feature I’ll introduce social network authentication and authorization. This might be good idea not to give anyone else potential access to my interview data by NOT protecting the access with authorization policies. I plan to use OAuth 2.0  packages to achieve this (Mongo Stitch allows it)