Technical Design Document

**Title:** Code Test Challenge

**CreatedBy:** Preeti

**Submitted on:** 28th August 2019

**Introduction**

This is a code test challenge done to compete for the internship program in Scientia. This is a web app that enables to manipulate records stored in the database.

**Platforms Used**

Python3, Flask, Sqlite3, Javascript, HTML, CSS, also incorporated bootstrap and JSON

**Proposed Design**

This app will enable the user to add, view, search and delete database records stored in sqlite3 database. The UI rendered will follow responsive layout for clean and precise operable environment for devices with varying screen sizes, i.e., from wider desktops to smaller mobiles.

**System Architecture**

The flask framework of python is used to handle the server side functionalities. When the user visits this app, they are directed to the appropriate content based on the routing made according to the url. The required template is rendered according to the needs of the user operations. Sqlite3 database is used to store the employee records of name, designation, address and phone number. All pages are designed using bootstrap hence it will be mobile responsive.

The user can

* View all existing records on landing page
* Search for an employee by typing their name, designation or phone number in the text box. If no matches are found, appropriate message is shown.
* Delete an employee by simply clicking the delete icon corresponding to the employee record
* Add an employee by directing to another page, where the user can add employee details. The user data will be validated in the client-side using javascript. From there, the user will be directed to the home page after submitting the results, or simply click the cancel button to return back to the home page without data submission.

**Interface**

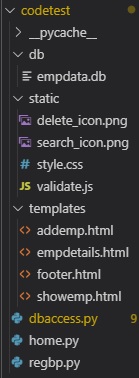
The main mode of communicating with views is templates. Templates are used extensively in this project by rendering, extending and including the desired templates in the appropriate places. Also, Fetch API is used along with JSON to render template while search operation is performed to load the view table instead of loading the entire page.

The blueprint of the app is taken by registering it to enable precise reading and understanding and also to avoid confusions while scaling it, if necessary, in the future.

**Data Model**

A single table is used here with database as empdata.db, table and columns as follows

**Folder Structure**



Files explained:

* empdata.db - database file
* static folder - resources used within this project, includes images, css and js files
* addemp.html - view for adding employee
* empdetails.html - skeleton for templates, used for extending
* footer.html - app footer, included in the template skeleton
* showemp.html - main landing page view, used to search, display and delete employees
* dbaccess.py - contains functions used for database operations
* home.py - main home page from which this app is run
* regbp.py - used to register blueprint

**Conclusion**

This is a fully functional app. Since sqlite is used, it is suitable for single user. For large scale implementation, it is wise to make use of other databases like MySql.