Table of Contents

[Introduction 1](#_Toc63634399)

[Requirements 1](#_Toc63634400)

[Specific Exclusions 2](#_Toc63634401)

[Implementation 2](#_Toc63634402)

## Introduction

This document aims to provide some insight on the approach taken in completing the requested assessment task.

The task was approached as a demonstration of a competency in the suggested technology stack, and not an attempt to develop a production-ready system. All tasks were covered to some degree, in the time allocated, as opposed to thoroughly completing specific tasks. For example, only a single unit test was implemented to show familiarity with the framework, as opposed to fully testing the code.

## Requirements

Build a simple phone book application with the following functions:

* Add entry to my phone book
* View phonebook
* Search for entries (textbox to search and listview)

and data structures:

* PhoneBook { Name, Entries }
* Entry { Name, PhoneNumber }

The components to be developed include:

* an SPA front-end (React)
* an API (WebAPI)
* an embedded database (SQLite)

Additionally,

* Target framework: .Net5
* Docker-enabled
* Unit testing
* Demo video of solution working
* Solution should be broken down into areas of concern.

## Specific Exclusions

* Only a single, phonebook was implemented and loaded as the default

The following were excluded:

* Edit and delete functions
* Design for Security.
  + Specifically, access to the PhoneBook has not been secured for use only by authorised, authenticated users
* Field validation
* Error handling, logging
* User interface design
* Documentation
* Only the API is docker enabled

## Implementation

The solution comprises the following projects:

* PhoneBookEntity, library containing the entity and helper classes shared by the API and Client, to facilitate communication
* Database
  + The SQLite db, PhoneBook.sqlite, can be found in the API root folder.
  + Tables
    - Book { BookId, Name }
      * *hack:* A single book, with Name = “defaultBook” is created at setup time
    - Contact { BookId, Name, PhoneNumber }
* PhoneBookRepo, library containing the database and repository classes
  + ORM: Dapper
* PhoneBookAPI
* PhoneBookClient
  + On load, the client gets the default book. All operations then apply the loaded book.
  + *hack:* The search field, is passed to the API in the Contact.Name field. It is used to find contacts in the book with names or numbers that match the search text
* PhoneBookTest
  + *hack:* The BookId is set to 1 in the setup of the tests
  + *hack:* A new db is created with each test run, to guarantee the values in the db
* This document and the video, PhoneBook.mp4, can be found in the solution folder.