# QA Automation Exam: “Contact Book”

**Exam** assignment for the [“QA Automation” Course @ SoftUni](https://softuni.bg/trainings/3179/qa-automation-january-2021) (22-Feb-2021).

Please submit your work as a single zip / rar / 7z archive holding the source code for each problem, without the binaries (exclude bin, obj, node\_modules and other unneeded folders).

## The “Contact Book” System

“**Contact Book**” is a simple information system for managing **contacts**. Each contact consists of first name, last name, email, phone and comments. Users can **view** the contacts, **search** by keyword, and **add** new contacts. You are given the RESTful **API** + **Web** client app + **Desktop** client app + **Android** mobile app client for the contact management system. Your task is writing **API tests and UI automated tests** for the system.

You are given the following project assets:

* <https://github.com/nakov/ContactBook> – ContactBook RESTful API and Web client app
* <https://github.com/nakov/ContactBook-DesktopClient> – ContactBook desktop app for Windows
* <https://github.com/nakov/ContactBook-AndroidClient> – ContactBook mobile app for Android

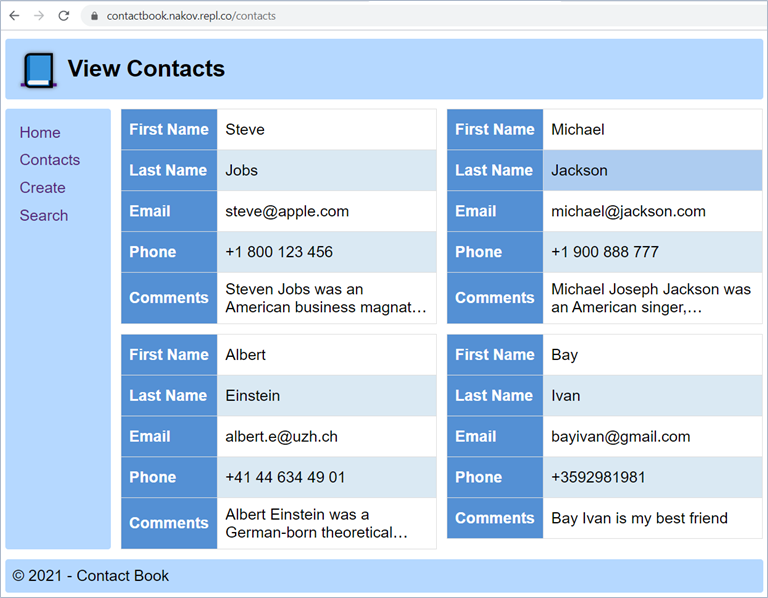
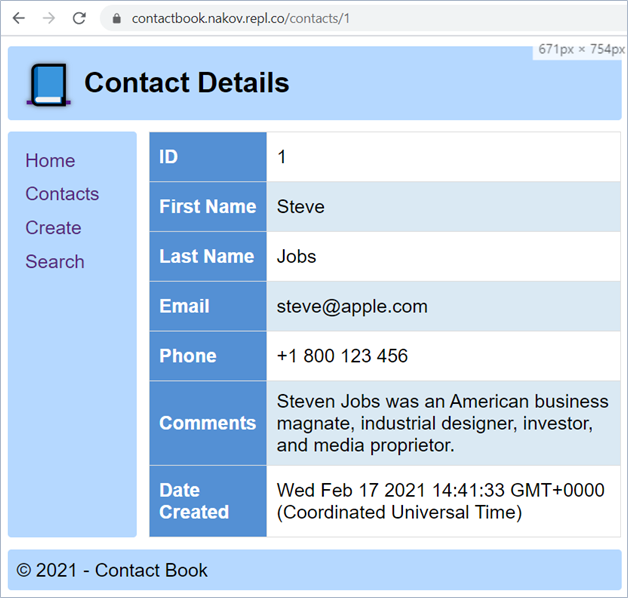
### Web App Functionality

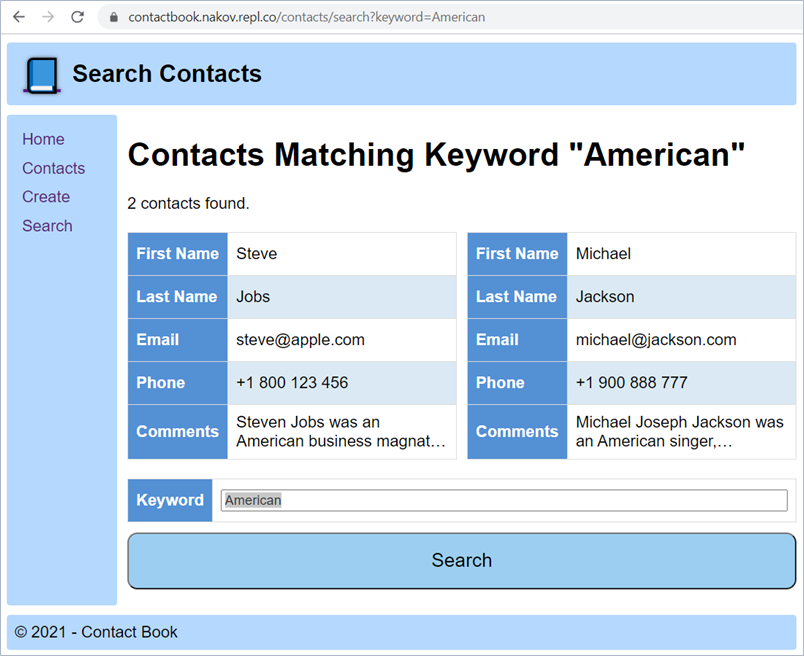
The **"ContactBook" Web app** supports the following operations:

* Add a new contact by given contact data
* View all contacts
* View contact details
* Search contacts by keyword

Run the Web app from: <https://contactbook.nakov.repl.co>.

You can browse the app **source code** and play with the app at: <https://repl.it/@nakov/contactbook>.

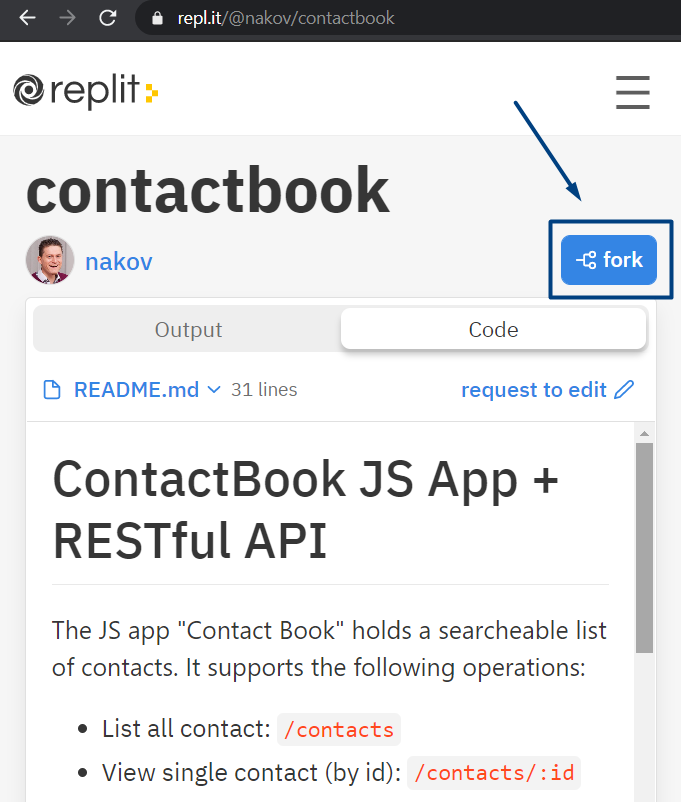
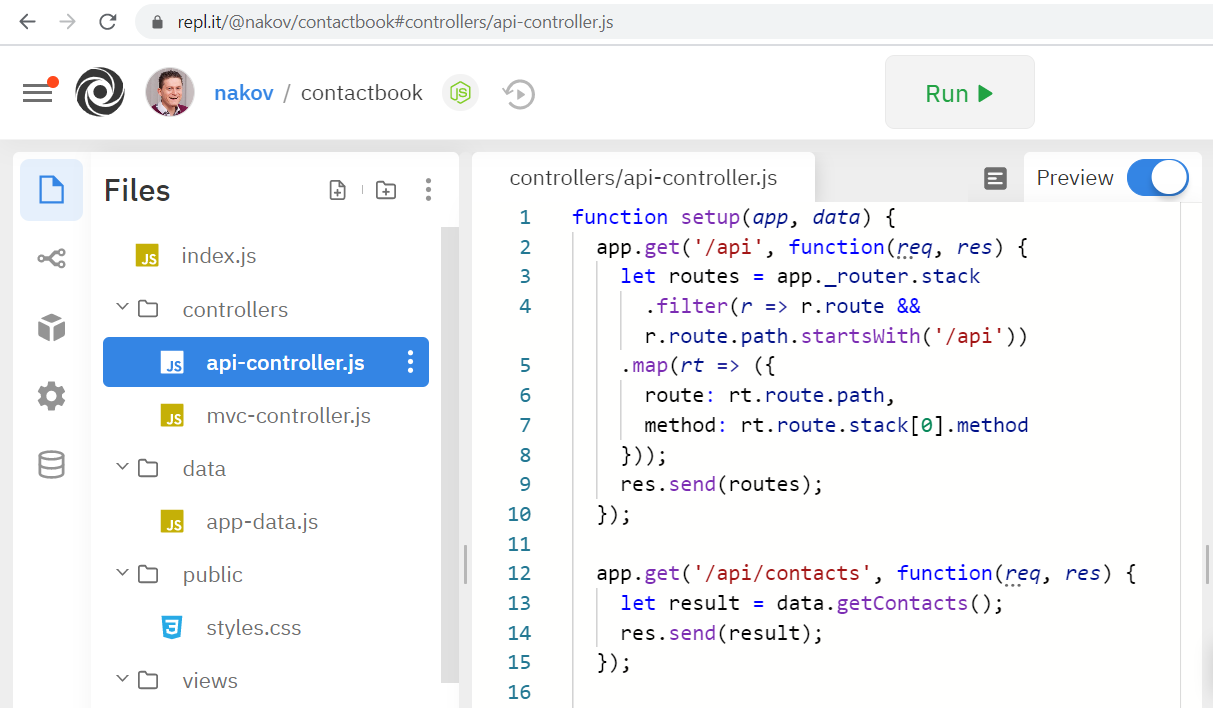
 

### Installing and Running the App

To avoid conflicts, it is highly recommended that you **fork this app** into your **own repl.it account**, e.g.

* [https://repl.it/@{your-account}/contactbook](https://repl.it/@%7byour-account%7d/contactbook)

Optionally, you can **install** and **run** the app on your **local machine**:

|  |
| --- |
| git clone https://github.com/nakov/ContactBook  cd ContactBook  npm install  npm start  start http://localhost:8080/api |

### Resetting the App

The app **does not have a persistent database** storage, so you can **reset it** by a simple **restart** (stop & start).

* After restart, you will lose all changes and the default sample data will be populated automatically.

### API Endpoints

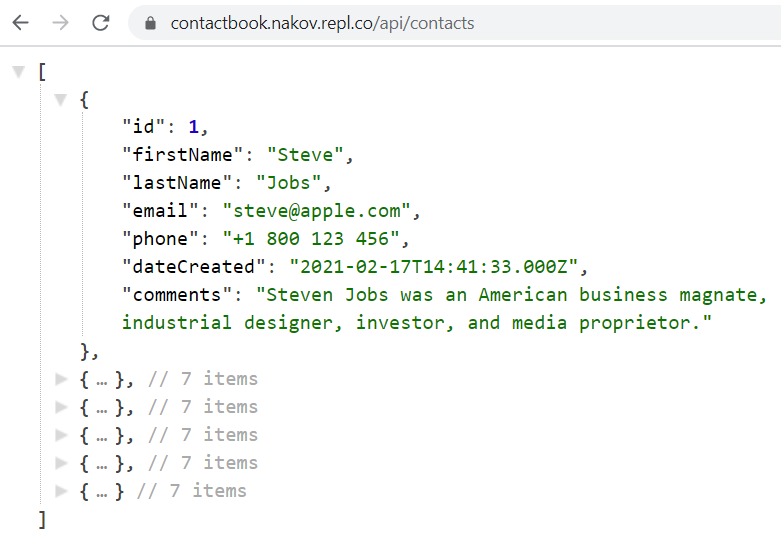
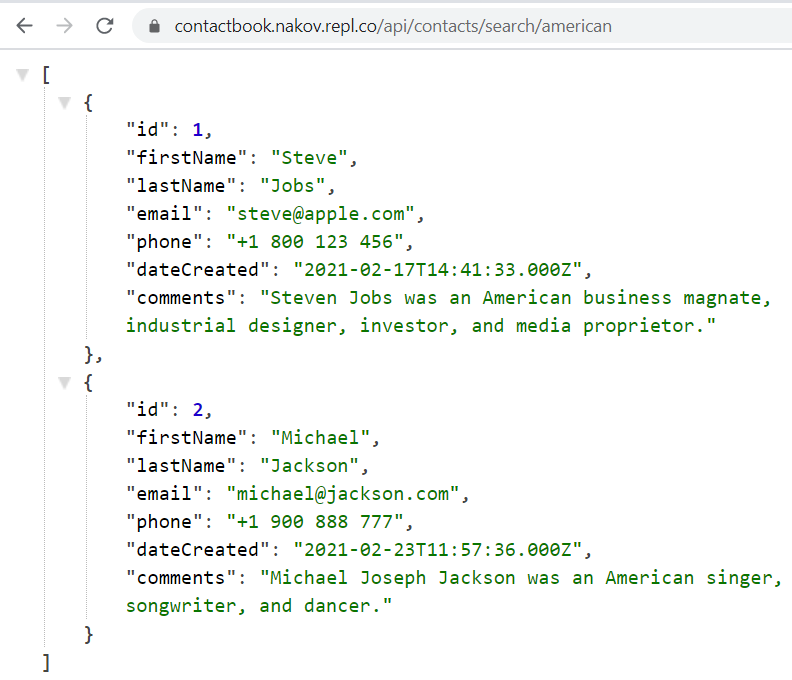
ContactBook exposes a **RESTful API**, available at:

* <https://contactbook.nakov.repl.co/api>

The following **API endpoints** are supported:

* GET /api – list all API endpoints
* GET /api/contacts – list all contacts (returns JSON array of contacts)
* GET /api/contacts/:id – view contact details by id (returns JSON object)
* GET /api/contacts/search/:keyword – find contact by keyword
* POST /api/contacts – create a new contact
  + Post a JSON object in the request body, e. g.  
    {"firstName":"Marie", "lastName":"Curie", "email":"marie67@gmail.com", "phone":"+1 800 200 300", "comments":"Old friend"}
  + Fields firstName, lastName and phone are mandatory
* DELETE /api/contacts/:id – delete a contact by id

This is a sample output from an API call to /api/contacts:

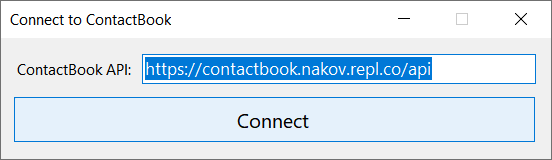
 

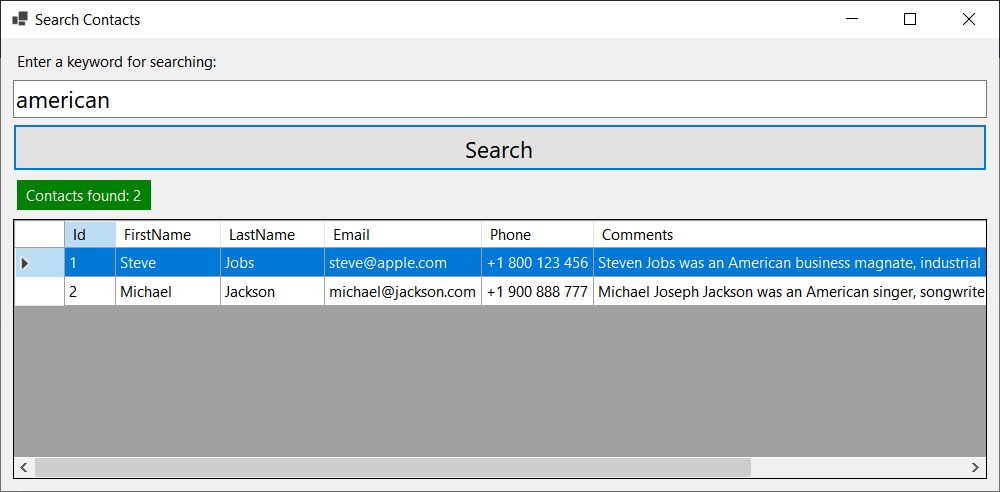
### Windows Desktop Client

ContactBook has a Windows desktop client app, available from:

* <https://github.com/nakov/ContactBook-DesktopClient/releases>

The app supports **searching for contact**, using the ContactBook API:

[](https://user-images.githubusercontent.com/1689586/108594524-2e527d00-7383-11eb-8958-a5a1ce0b5e29.png)

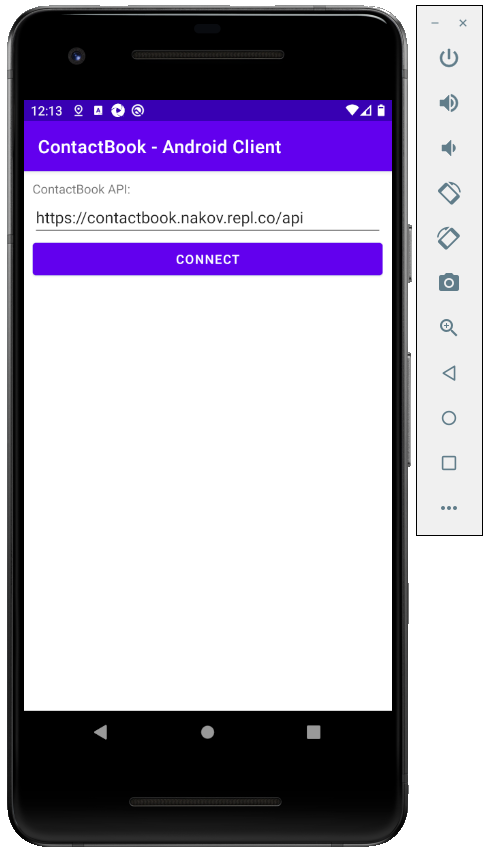
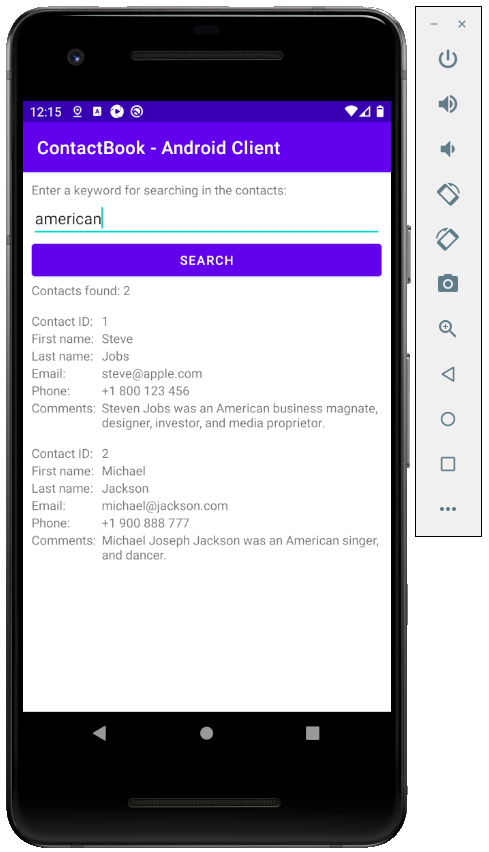
[](https://user-images.githubusercontent.com/1689586/108594544-47f3c480-7383-11eb-8b36-38cc05c77729.png)

### Android Mobile Client

ContactBook has an Android mobile client app, available from:

* <https://github.com/nakov/ContactBook-AndroidClient/releases/>

The app supports **searching for contact**, using the ContactBook API:

## ContactBook RESTful API: Automated API Tests

Your task is to write **automated tests** (in C#, Java, JavaScript or other language) for the above provided API endpoints. You should implement the following automated tests:

* List contacts and assert that the first contact is “Steve Jobs” (5 score).
* Find contacts by keyword “albert” and assert that the first result holds “Albert Einstein” (5 score).
* Find contacts by keyword “missing{*randnum*}” and assert that the results are empty (5 score).
* Try to create a new contact, holding invalid data, and assert an error is returned (5 score).
* Create a new contact, holding valid data, and assert the new contact is added and is properly listed in the contacts list (15 score).

You are free to use a **testing framework** of choice (e. g. NUnit or JUnit) and **external libraries** (e. g. RestSharp).

## ContactBook Web App: Automated Selenium UI Tests

Write **Selenium-based automated UI tests** for the “**ContactBook**” app. You should implement the following **automated UI tests**:

* List contacts and assert that the first contact is “Steve Jobs” (5 score).
* Find contacts by keyword “albert” and assert that the first result holds “Albert Einstein” (5 score).
* Find contacts by keyword “invalid2635” and assert that the results are empty (5 score).
* Try to create a new contact, holding invalid data, and assert an error is shown (5 score).
* Create a new contact, holding valid data, and assert the new contact is added and is properly listed at the end of the contacts page (15 score).

You are free to use a **testing framework** of choice (e. g. NUnit or JUnit), but your primary Web UI automation tool should be **Selenium**. You are free to use **external libraries and tools**.

## Appium Tests

Choose one of the next two problems: Android app UI tests or Windows app UI tests.

## ContactBook Mobile App: Automated Appium UI Tests

Write **Appium-based automated mobile UI tests** for the “**ContactBook**” Android mobile app. Implement the following automated testing **scenario** (30 score):

* Open the app.
* Connect to your backend API service.
* Search for keyword “**steve**”.
* Assert that “**Steve Jobs**” is returned as result.

You are free to use a **testing framework** of choice (e. g. NUnit or JUnit), but your primary Android UI automation tool should be **Appium**. You are free to use **external libraries and tools**.

You are free to implement and run the tests in cloud-based testing platform like **BrowserStack** or in a local instance of Appium with Android Emulator or physical Android device.

## ContactBook Desktop App: Automated Appium UI Tests

Write **Appium-based automated Windows UI tests** for the “**ContactBook**” Windows desktop app. Implement the following automated testing **scenario** (30 score):

* Start the app.
* Connect to your backend API service.
* Search for keyword “**steve**”.
* Assert that “**Steve Jobs**” is returned as result.