<Company Name>

Phone book Application
Test Plan

Version 1.0

Revision History

Date	Version	Description	Author
13/feb/21	1.0	First version	Krivosheev Svyatoslav

Test Plan

1. Introduction

1.1 Purpose

This Test Plan document for the Phone book application helps to get a better understanding about how to test the project.

1.2 Background

This application is no-UI application which can manage user records and their contacts records. It can create, read, update or delete them.

1.3 Scope

The goal of testing REST backend application is to increase the level of trust in the service, try to find errors.
The final results of the testing process will be the document REPORT.txt
This plan contains only functional tests contains all functionality of the application.
There are no risks or contingencies that may affect the design, development or implementation of testing.

Requirements for Test

The listing below identifies those items—use cases, functional requirements, and non-functional requirements—that have been identified as targets for testing. This list represents what will be tested.

Functional tests:

API should correctly perform such User actions as:

1) add a user
2) remove a user
3) change a user
4) find users by name
5) get a list of all users

API should correctly perform such Contact actions as:

1) add a contact
2) remove a contact
3) change a contact
4) find contacts in a user’s phone book by a phone number or contact’s ID
5) get a list of all user’s contacts

2. Test Strategy

There are three stages of testing are planned:

1. Analysis, test plan creation, partial fulfilment of some functional tests.
2. Detailed fulfilment of functional tests uncovering and describing the bugs.
3. Fixed bugs check.

2.1 Testing Types

2.1.1 Function Testing

Http method & URL:	Short description
GET /users/	Should return all users from database in json format (array). Sends 200 – OK code.
POST /users	If body contains proper User in json format, the answer must contain user object in json format, id generation is on server side. Sends 201 - CREATED code. Users with same names should have different ids. FirstName shouldn't be empty. Sends 400 – Bad Request otherwise.
GET /users/{userId}	Should return one user from database with proper id attribute. Sends 404 – Not found code if userId was wrong.
PUT /users/{userId}	Should return 200 – OK if json object in body was fine. Should have firstName – otherwise sends 400 – Bad Request code. Should update user with such userId.
DELETE /users/{userId}	Deletes the user with userId. Should return 202 – Accepted. Should really deletes such user with iserId. Sends 404 – NOT FOUND if user with such userId was not found in database.
GET /users/search	Name parameter is required. Otherwise sends 400 – Bad Request. Should search all users with such firstName or lastName as name parameter. Returns 200 – OK code and sends array of users. If no users with such names found, returns 404 – Not Found.
GET /users/{userId}/contacts	Get all contacts of user with id userId in json format (array). Returns id, firstName, lastName, phone, email of every contact. If there are no user with id userId, returns 404 – Not found with error "User with id: <id> not found".
POST /users/{userId}/contacts	Adds a contact of some user with userId. Phone must be not empty. Phone must be 10 digits long. Other fields may be null. Returns 201 – Created. If userId was wrong returns 404 – Not Found.
GET /users/{userId}/contacts/search	Will be able in version 2.0 of plan
DELETE /users/{userId}/contacts /{contactId}	Will be able in version 2.0 of plan
PUT /users/{userId}/contacts/{contactId}	Will be able in version 2.0 of plan
GET /users/{userId}/contacts/{contactId}	Will be able in version 2.0 of plan

Tools

The following tools will be employed for this project:

Postman, Swagger, Java Tests