Contents

[Tools used. 2](#_Toc128514495)

[Project setup 3](#_Toc128514496)

[Project and folder structure 3](#_Toc128514497)

[Repository and Model 5](#_Toc128514498)

[Service and its implementation 6](#_Toc128514499)

[Rest Controllers 7](#_Toc128514500)

[Database setup and connection 8](#_Toc128514501)

[Spring security 9](#_Toc128514502)

[Running the project 10](#_Toc128514503)

# Tools used.

|  |  |
| --- | --- |
| **Tools** | **Description** |
| Spring tool suite | * IDE for coding the spring application. * Comes with built in spring starter project builder and spring snippet |
| Postman | * Desktop application for testing REST API |
| GitHub | * Repository for code backup, storage, and version management |
| GitHub Link | <https://github.com/bibash44/spring-boot-addressbook-API> |
| Demo Video Link | <https://youtu.be/2Zryh_cge60> |

# Project setup

* Project was built using spring starter in Spring Tool Suite IDE
* Java Version 17 was used.
* Dependencies packages like **Spring Web, Spring Data JPA, Spring Security, Lombok, H2Database** were installed.

Graphical user interface, text, application, email

Description automatically generated Graphical user interface, text, application

Description automatically generated

# Project and folder structure

* Necessary packages like Model, Controller, Repository, Service, Service Implementation were created along with required files as shown below.

Graphical user interface, application

Description automatically generated with medium confidence

# Repository and Model

* Model AddressBook was created along with the required attributes.
* Getter, Setter, constructor were written (*Note: Lombok was not working*)
* Hibernate JPA annotation was used for identifying Entity and its attributes.
* Finally, repository for the Model was defined inside the repository package

Graphical user interface, text, application, email

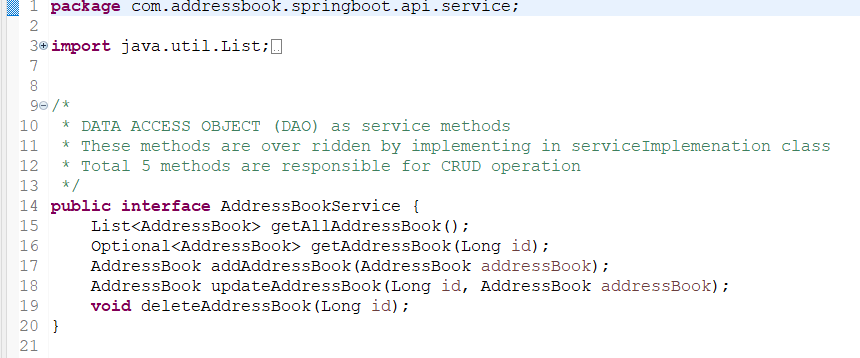
Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

# Service and its implementation

* Services were defined in a interface (Service Interface) and implemented (ServiceImpl.Java) in a service implementation class by overriding them
* CRUD operation DAO (Data Access Object) were written as service and the business logic was implemented using those DAOs as shown below
* Implemented methods CRUD operation were performed using JPA provided default ORM (Object Relation Mapping)



Graphical user interface, text, application, email

Description automatically generated

More on code …….

# Rest Controllers

* All the rest controllers and API end points were defined in controller file (AddressBookController.Java)
* Base end point was set to **/api/v1/address-book**
* GET, POST, PUT, DELETE were used for creating the API end points as shown below

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

More on code …….

# Database setup and connection

* H2 in-memory database was used.
* All the necessary variables and data source were defined in application.properties file for enabling database connection as shown below

Graphical user interface, text, application

Description automatically generated

# Spring security

* Spring security was configured using spring security dependency
* Security was defined for authorization and authentication.
* Authentication was defined for user by creating bean using UserDetailsService authentication service and SecurityFilterChain HTTPRequest authorization all provided by spring security
* HTTP basic authorization is used instead of form authorization.
* The defined user roles were assigned to each route in Controller routes where endpoints were created which only allows specific user to use that route.

Spring Security Configuration

Graphical user interface, text, application, email

Description automatically generated

Rest Controller

Graphical user interface, text, application, email

Description automatically generated

# Running the project

* Project can be run in the local machine.
* Download the project and import in the IDE like **VS CODE, SPRING TOOL SUITE or intellij IDE**
* Run Maven install or execute POM.XML before running the project for downloading maven and JAR files
* Finally run the project as Spring boot Web application
* Test the API using POST man or CURL