**Project Online Booking System – BookIt**

Basic web based online booking system for pet friendly hotels.

**Functionalities and features:**

In order to use the system, the user needs to be logged it.

* Basic search option
* Check room availability and rates
* Book rooms
* Discount if more than 1 night is booked
* **CRUD** operations like:
  + create reservations
  + view existing reservations
  + update reservations
  + cancel reservations
* Simple file uploads
* Error handling
* Data validation (client side and server side)
* Role based authorization and authentication
* Simple test payments via cloud-based payment service - Stripe API
* Google Maps API to (dynamically?) show maps of hotels
* Cloudinary API for file storge

The system:

**Technology:**

* Java
* Spring Framework (MVC architecture)
* Spring Boot
* Spring Security
* Spring Data
* JPA, Hibernate ORM
* MySQL
* Thymeleaf
* HTML, CSS, Bootsrtap, JQuery, Javascript, Maven

**Tests**

* Unit Tests, Integration tests
* Junit, Mockito
* H2 in-memory DB
* Coverage 61%

The project demonstrates:

* Restfull end points
* MVC architecture
* Use of intercepors
* Use of Scheduler
* Spring Event
* Use of Model mapper

**Admin credentials:**

**email**: [admin@admin.bg](mailto:admin@admin.bg)

**pass**: a123456789A#

**User credentials:**

**email**: [pesho@abv.bg](mailto:pesho@abv.bg)

**pass**: test

**email**: [sisi@abv.bg](mailto:sisi@abv.bg)

**pass**: a123456789A#

PAYMENT

Via Stripe

Test card:

* 4242 4242 4242 4242
* Use a valid future date, such as 12/34.
* Use any three-digit CVC (four digits for American Express cards).
* Use any value you like for other form fields.
* ZIP – 10001

Testing

Your tech lead trusts you to do a good job, but testing is important whether you're an excel number-cruncher or a full-stack coding superstar! The QA team at Super*Duper*Drive carries out extensive user testing. Still, your tech lead wants you to write some simple Selenium tests to verify user-facing functionality and prove that your code is feature-complete before the testers get their hands on it.

1. Write tests for user signup, login, and unauthorized access restrictions.

* Write a test that verifies that an unauthorized user can only access the login and signup pages.
* Write a test that signs up a new user, logs in, verifies that the home page is accessible, logs out, and verifies that the home page is no longer accessible.

1. Write tests for note creation, viewing, editing, and deletion.

* Write a test that creates a note, and verifies it is displayed.
* Write a test that edits an existing note and verifies that the changes are displayed.
* Write a test that deletes a note and verifies that the note is no longer displayed.

1. Write tests for credential creation, viewing, editing, and deletion.

* Write a test that creates a set of credentials, verifies that they are displayed, and verifies that the displayed password is encrypted.
* Write a test that views an existing set of credentials, verifies that the viewable password is unencrypted, edits the credentials, and verifies that the changes are displayed.
* Write a test that deletes an existing set of credentials and verifies that the credentials are no longer displayed.

**TESTING**

**Functional**

* + Unit testing
  + Integration testing
  + Black box testing
* Equivalence partitioning
* Boundary value analysis
* Decision table testing

[Decision Table Testing: Learn with Example (guru99.com)](https://www.guru99.com/decision-table-testing.html)

* + White box testing

**System testing**

**Entire system – front-end (UI logic), back-end (business logic), database**