**K15t Java Meetup Specifications**

**Tech Stack Specifications:**

**Git URL:** <https://github.com/Harisanthosh/K15t-Webapp>

**Live URL:** <https://pacific-crag-83681.herokuapp.com/>

**Swagger API documentation:** <https://pacific-crag-83681.herokuapp.com/swagger-ui.html>

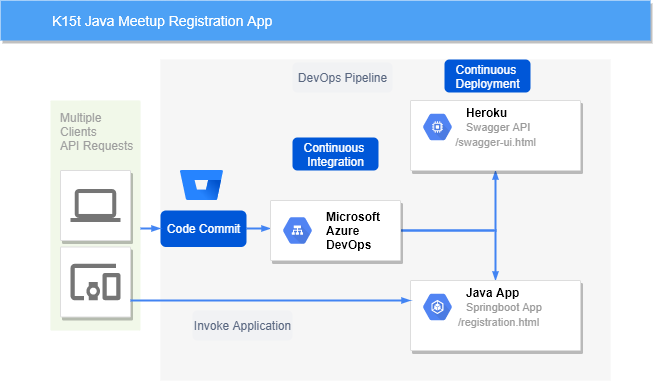
**Objective**

The objective is to develop a production ready Java Meetup application from the starter template provided by K15t team.

**Application Architecture**

The Application end to end architecture appears as indicated below

|  |  |
| --- | --- |
| Tech | Tools |
| Version Control | GitHub |
|  |  |
| Continuous Integration | Microsoft Azure Devops Pipeline |
| Continuous  Deployment | Microsoft Azure Devops Release |
| Deployment Server | Heroku Dyno |
|  |  |
| API Documentation | Swagger OpenAPI 2.0 |
| Application | Springboot + Thymeleaf + Springboot MVC |
|  |  |

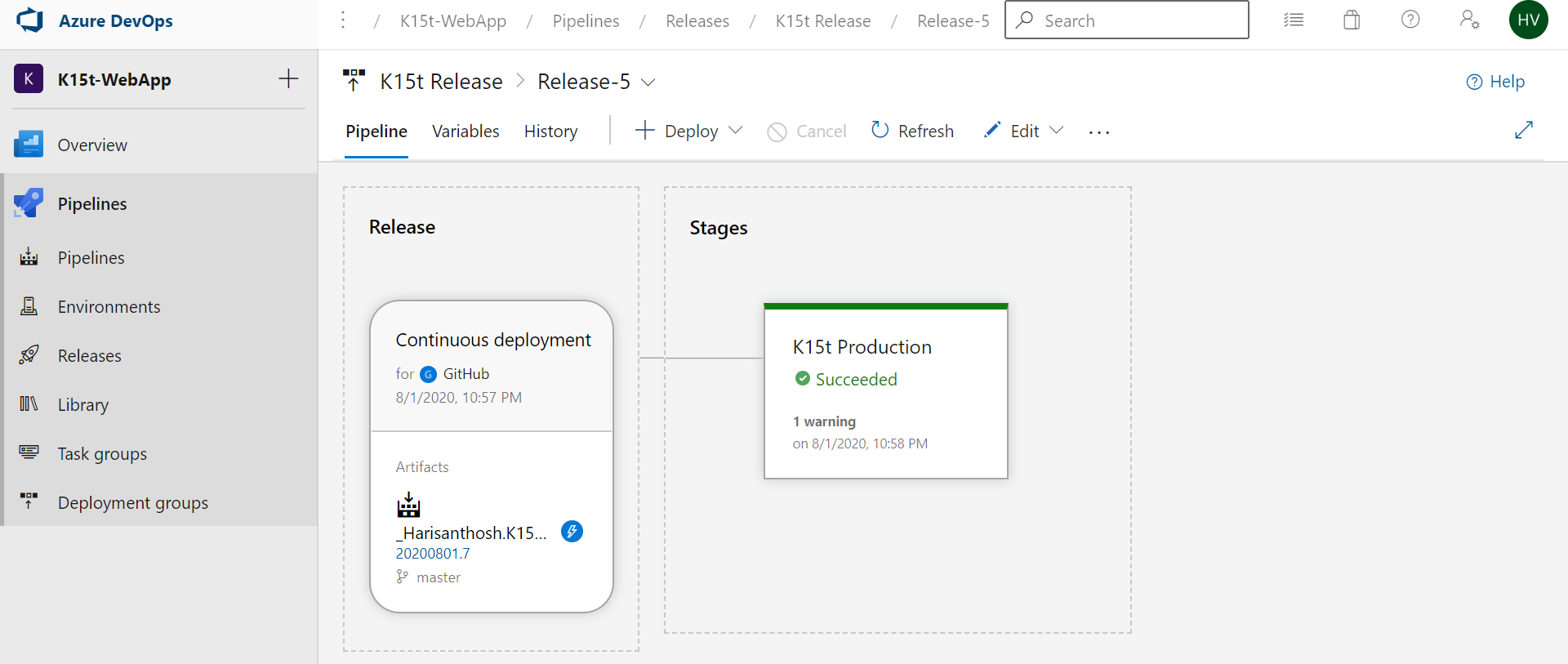
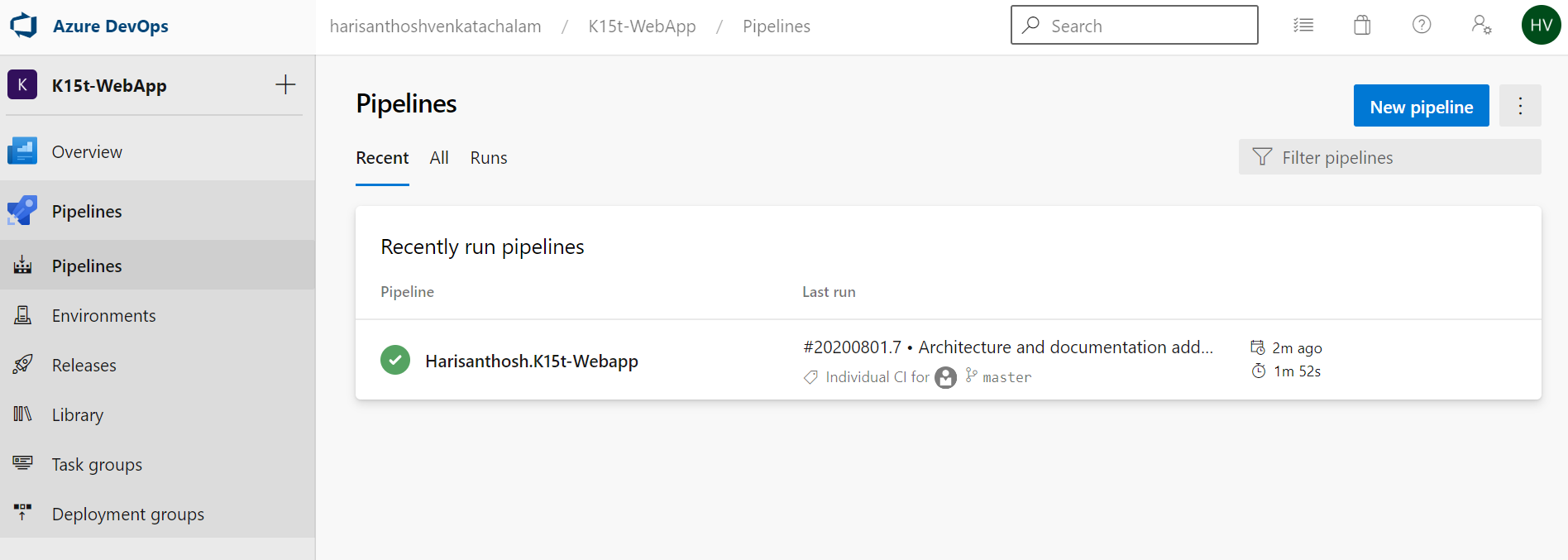


**Figure 1 -** K15t Java Meetup architecture

The Java Meetup application tech stack used for development is as indicated below:

|  |  |
| --- | --- |
| Tech Stack | Comments |
| Spring Boot | 2.2.0 |
|  |  |
| Java | 1.8 |
| Swagger | For serving API documentation in OpenAPI Standards |
| Spring Rest MVC | Upgraded to Spring REST MVC instead of Jersey because |
|  | * we can quickly roll out our REST API and we can easily integrate with other tools |
|  | * Spring provides REST supports for both web applications and web services |
|  | * Jersey requires additional configurations |
| Thymeleaf | Better Spring boot support and easy choice for HTML 5 web development |
|  |  |
| Lombok.jar | Removes boiler plate code |
| Spring Security | Support for Password encoding |
| Crypto |  |
| Spring-boot-starter- | Test support utilities |
| test |  |
|  |  |

The DevOps Pipeline appears as shown below. Upon executing the pipeline, the executed bundles and the corresponding dependencies are deployed to Heroku.



**Business Requirement Considerations**

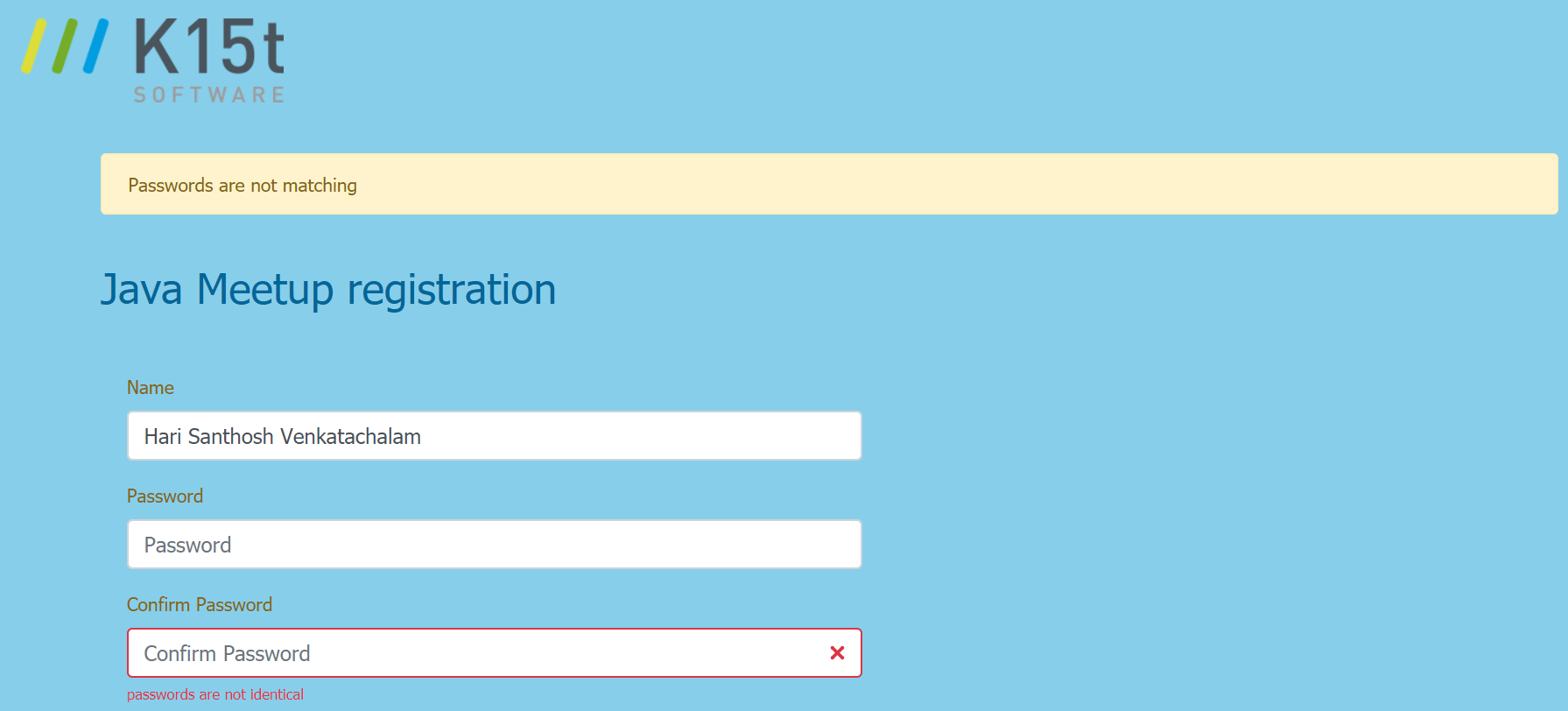
1. **Validations**

Validations has been implemented at U.I level as well as at the entity level. Password encryption has been implemented using Spring Security Crypto

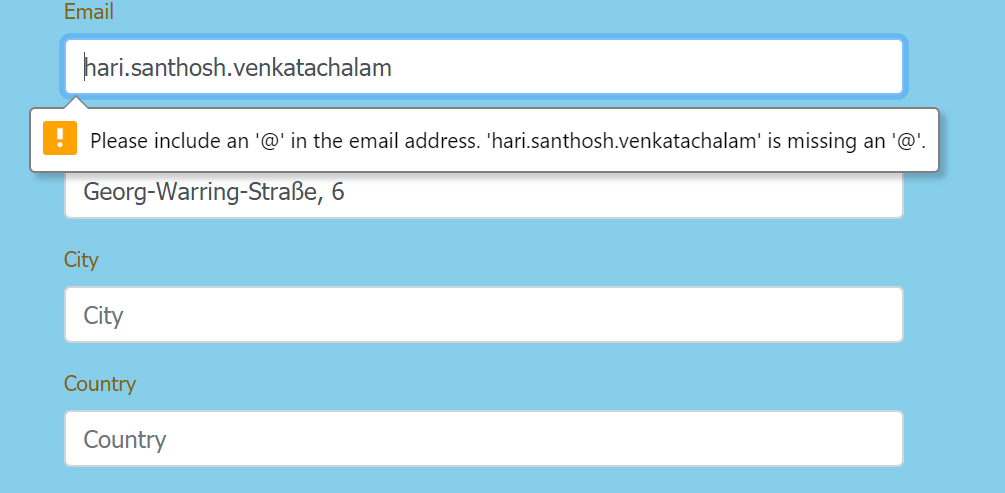
|  |  |
| --- | --- |
| **Validations** | **Comments** |
| Mandatory Fields | Name, Email, Password, Confirm Password, Phone, Terms & Conditions |
| Header | Include K15t logo |
| Footer | Include contact email link |
| **Field Validations** |  |
| Name | Minimum 2 characters |
| Password | Minimum 6 characters |
| Confirm Password | Should match with Password Fields |
| Email | Should be valid email id |
| Phone Number | Should be valid Phone number |

**The below example validations are done from the application.**

**Password Validation:**



**Email Validation:**



1. **Notifications Success Notifications:**

When user successfully registers for the meetup

Expected Notification: User Registration Successful

**Error Scenarios:**

1. When user tries to register with email id to avoid duplicate registrations, prompt the user with the following notifications.

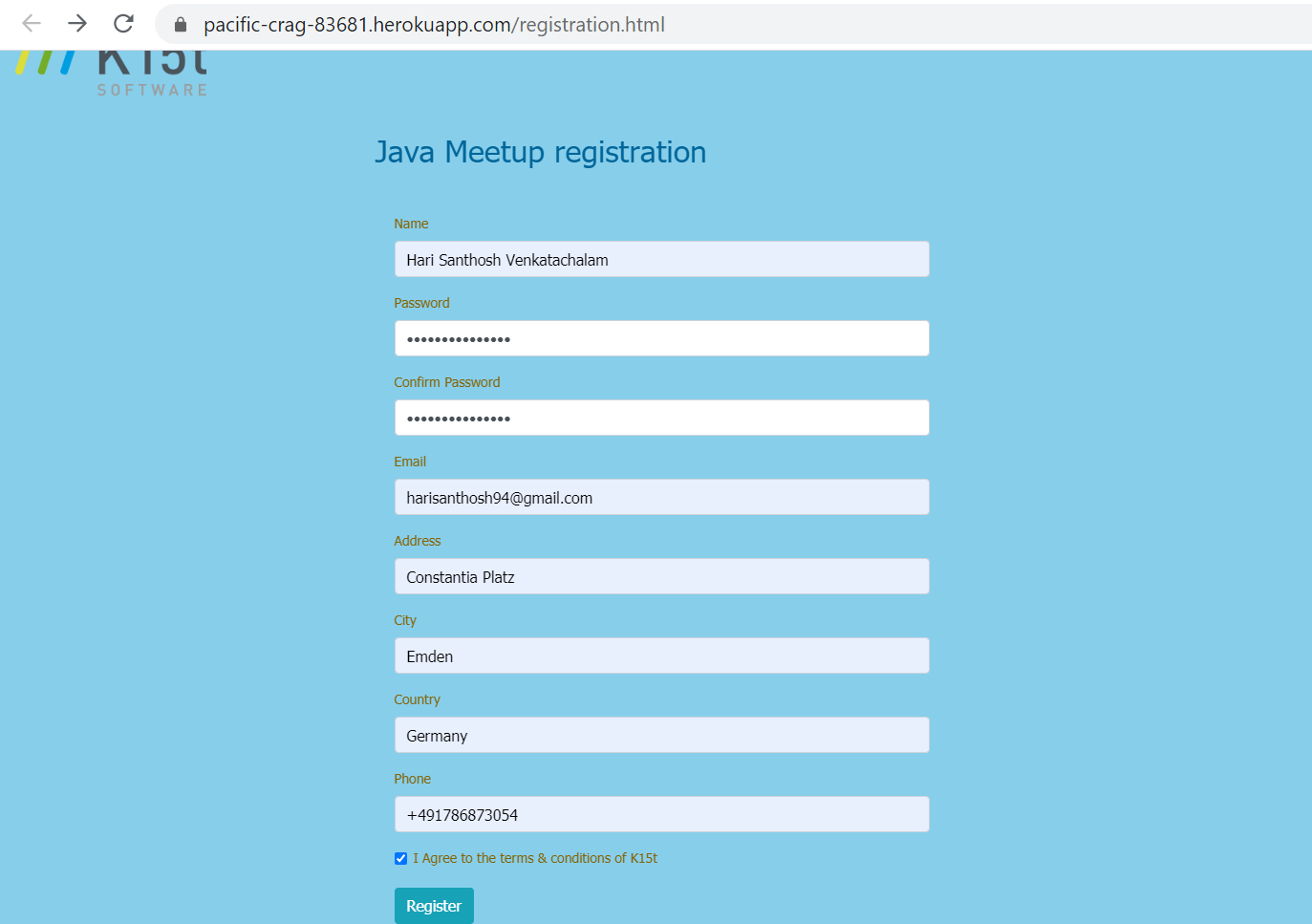
User has been already registered

1. When user tries to submit the form with invalid password combinations. Prompt the user with the following notifications.

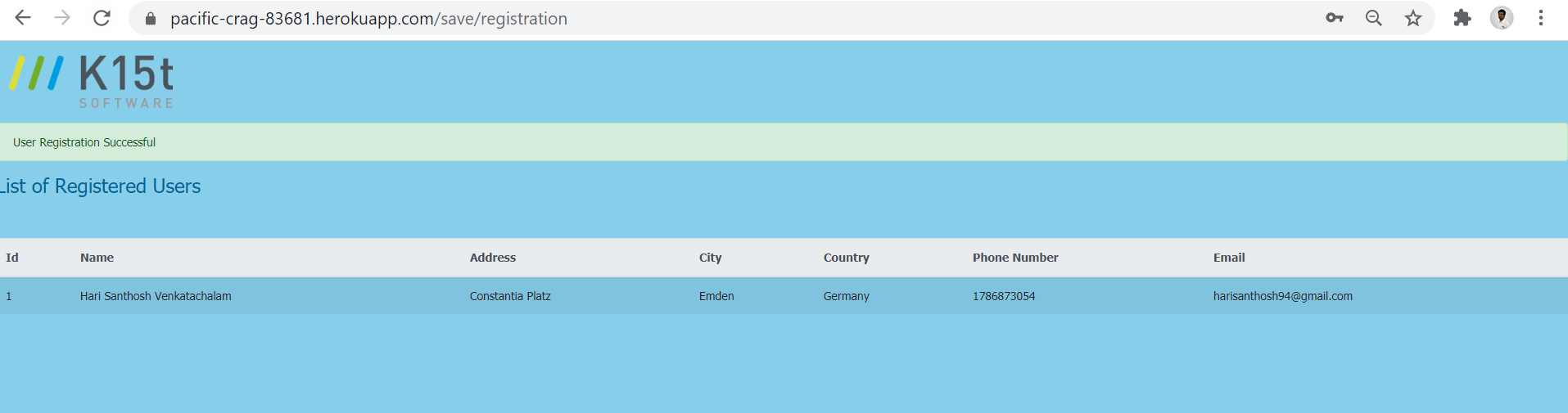
Passwords are not matching

**U.I Layouts:**

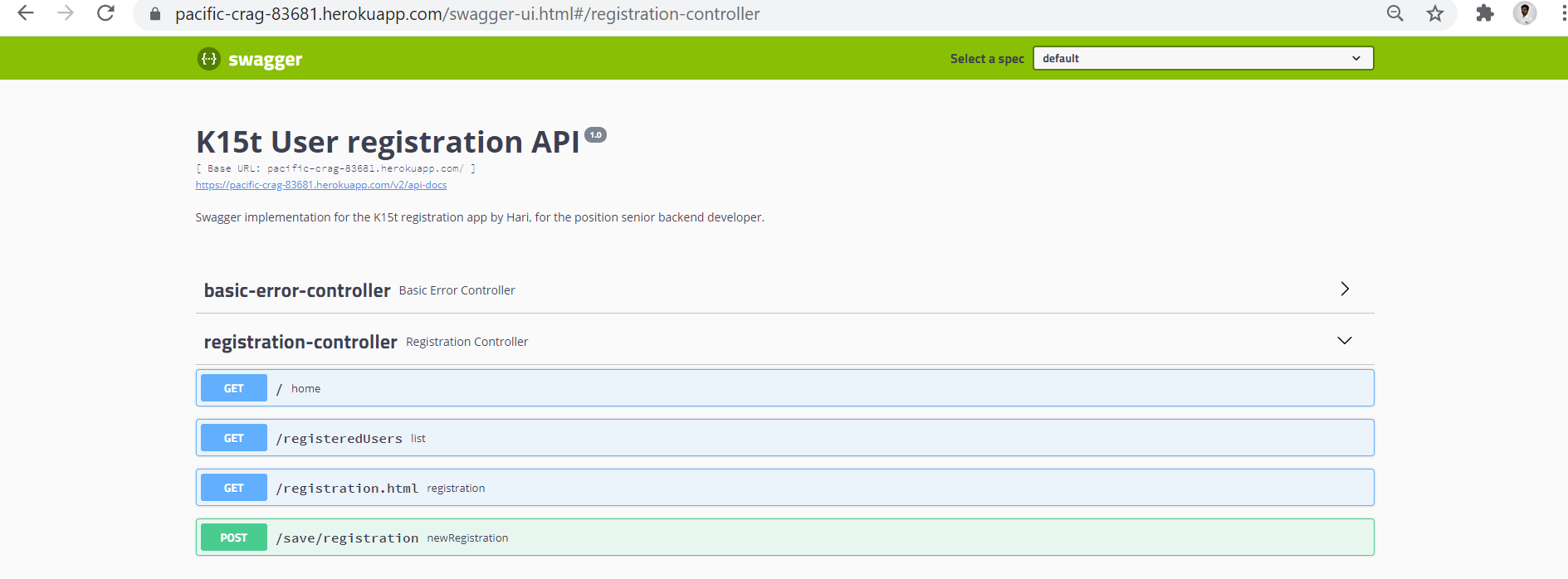
1. Landing Page / Home Page



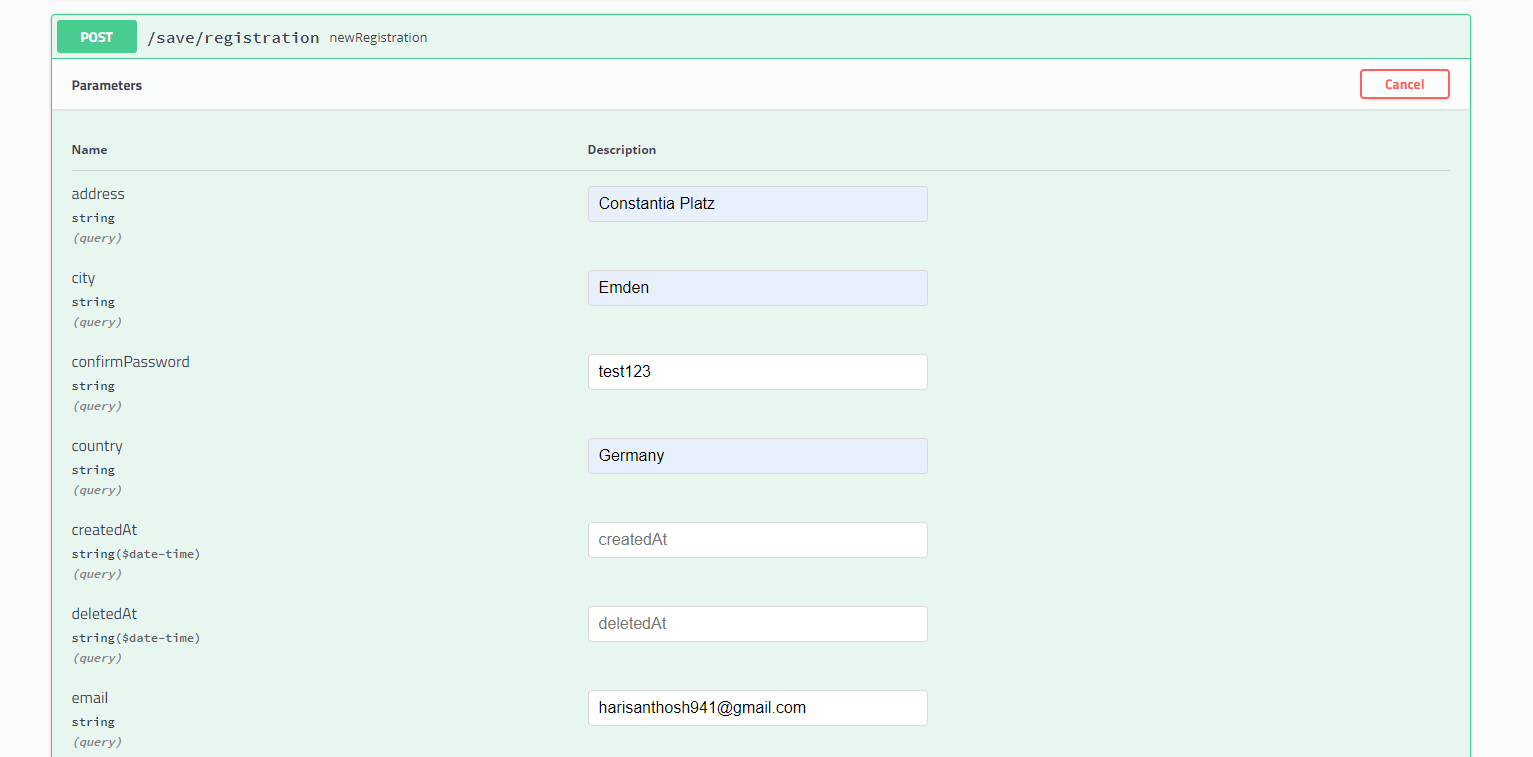
1. After successful registration redirect to success page



1. API Documentation



When a new user is created via POST /save/registration API the user gets created as shown below



Upon successful API call, the new user is added and displayed via the registeredUsers template. 

5. Data Storage:

The embedded database H2 is used for its simplicity and for demonstrating the concept of persistence from Spring boot. Password field will get stored using encryption as shown below.

