Code.Hub

Technikon
The Technicians on the Web

Requirements specification document

July 2023

Contents

1. Project general description

A Renovation Contractor Agency, Technikon, within the framework of its operation, needs an application that will enable the employees - managers of its platform to have access to information concerning customers and repairs. It will also allow its customers to oversee the progress of repair/renovation work on their property.

A full-stack application using React, Java Spring, and PostgreSQL should be designed and implemented.

2. User roles

The roles of the application users are:

- Admin (Employee of the Agency)
- Property Owner, User of the platform, customer

The pages of the admin and the property owner have similar layouts and functionalities. The main difference of course is that the user deals with his own data, whereas the admin can manage all. The differentiation of the pages is implemented with proper authentication and authorization.

3. Functional requirements

3.1 Property Owners' Functionalities

For property owners, the system will contain the following options: Create, Search, Update, Delete.

Create Owner of the Property with the following fields:
- TIN number (tax identification number): a unique identifier that characterizes users,
- Name,
- Surname,
- Address,
- Phone number,
- Email,
- Username
- password

Search: It will contain the following fields to search:
- TIN number
- email
- name

Update: the address, email, password can be changed.

Delete: the owner can be deleted permanently in case of a mistake. If it has been related to other entities can be deactivated.

3.2 Property Functionalities

It will contain the following functions: Search, Create, Delete, Update.

Create a Property with the following fields:

- A Property Identification Number, which coincides with the corresponding number of E9 and will uniquely characterize the property,
- Property address,
- Year of construction,
- Type of property (Detached house, Maisonette, Apartment building),
- TIN number of its owner.
- Upload picture of the property
- Add map location of the property

Search: The administrator will be able to search for properties based on various criteria, but mainly

- Property ID Number
- TIN number

Update: Can change any wrongly inserted data.

Delete: the property can be deleted permanently in case of a mistake. If it has been related to other entities can be deactivated.

Business rules

- Validations when entering data: For example, you cannot enter another user with an existing email or TIN number. Exceptions will be thrown.
- The owner can have more than one property.

3.3 Property Repair Functionalities

It will contain the following functions: Search, Create, Delete, Update.

Create:

- Date (datetime) of the scheduled repair
- Short Description of repair,
- Type of repair (Painting, Insulation, Frames, plumbing, electrical work),
- Status of the repair (Scheduled, In progress and Complete - default pending mode)
- Cost of repair,
- Owner id for whom the repair is made,
- Property id for which the repair is made

- Description as a free-text field for the work to be done (e.g., installation of a solar water heater).

Update: Can change any wrongly inserted data

Search: Based on
- Date or Range of dates
- User ID in case we want to display all the repairs made for a property owner.

Delete: The property repair can be deleted permanently in case of a mistake. If it has been related to other entities can be deactivated.

3.4 Reporting

The administrator can get the reports in a given time period about the

- owners that register,
- properties that are registered,
- repairs, their statuses, and total cost,
- users that login or perform transactions,
- usages of each page of the site

3.5 Push Notifications for the user

Implement any of the following push notifications

- a repair (of the user) has changed status
- user data have been changed by the admin

3.6 Push Notifications for the admin

Implement any of the following push notifications

- a new property owner has self-registered
- a new property has been added by a user
- a repair (of any user) has changed status

Important Note

The system is a multi-user system. Therefore, check the behavior of the system when there is a simultaneous change of database by multiple users.

4. FrontEnd requirements

For all the above requirements the corresponding components (“pages”) should be built using ReactJS. Feel free to design the frontend part of your app as you wish to provide a great user experience. Show data and images the way you want but do not forget to present your user with a map displaying the properties if geolocation data are provided.

Note: In this part of your project you will collaborate with a UI/UX specialist colleague from SKYTALYS. You are going to learn how you can "read" Figma to extract all the appropriate prototype info to drive the implementation process.

5. Use Cases

5.1 Data Population

Create a few data to insert in the database. The data will be owners, properties, and repairs.

5.2 Usage simulation (testing)

Create a property owner with all the necessary details. Test all interesting cases with appropriate calls.

6. Non-functional Requirements

The following requirements must be met for creating the project:

- Use the standard Java programming environment for the course.
- Use Java SE 11 or later.
- Use Spring and Spring Boot.
- Use PostgreSQL Server for a database.
- Use ReactJS framework
- The use of Interfaces, Exceptions and Dependency Injection is strongly recommended.
- Use proper coding and architectural standards and conventions.
- For version control: the project source code must be delivered on GitHub. The contribution of each team member must be shown with the use of commits with clear and descriptive messages. It is a good idea if team members use their own branches. Also, use branches for the development of each feature.

7. Milestones

The milestones are given as a logical separation of the various tasks and will help us to allocate the various stages of implementation in time. It is not obligatory to observe them as ordered, but it is suggested that they be followed so that there is an indication of the progress of the deliverables.

Backend
Final push: 24/7/2023 at 12:00.

- Define the java interfaces for the required services.
- Preparation and design of domain entities and consequently the shape of the database. Implementation of the domain classes and corresponding tables in the database
- Implementation the CRUD functionalities in the repository classes, i.e., create/ search/modification/deletion

- Implementation of the required methods for services
- Implementation of Unit tests.
- Prepare and configure the application server.
- Introduce dependency injections when needed.
- Implement the endpoints with the appropriate Spring classes.
- Implement the login functionality and security features.
- Test the artifact with unit testing and postman collections.
- Implement the nice-to-have functionalities if there is time.

Front end
Final push: 24/7/2023 at 12:00.

- Implement the home pages of Admin and Property Owner / User
- Design the User and Admin pages and navigation.
- Implement import, search, modification, deletion.
- Make sure that you have implemented all functionalities described above.
- Implement the nice-to-have pages if there is time.

8. Deliverables

The deliverable for this project is the working version of the application in Github, as it would be for uploading to the production system to be used by the customers.

Add the following names as collaborators to Github:

- kztoup
- tsevdos
- cgiannacoulis
- jondan97

In the GitHub repository, your code will need to be ready to execute for anyone sharing the project. Create a README.md file which lists the requirements and the steps for the project to run.

The deadline to submit your presentation files is on **24/7/2023 at 21:00.**

Your work will be evaluated for monitoring your progress and the deadline is an absolute business requirement – along with creating the code, you will be evaluated for how well you can handle the date of submission for your work. GitHub keeps timestamps so ensure that all updates to your work happen within the deadline.