Square Health

Test Assignment

Backend Developer

Instructions

- **Please note that we will review each part of your project. So even if your project is not fully completed / functional, please submit it before the project deadline. Code structure, design, your creativity all matters.**
- Submit your work in a public git repository and share it with us.
- Following standard coding conventions will be highly appreciated.
- You need to email your project's Github repository URL to the following addresses: career@squarehealth.com.bd

Project Description

You are going to build a web application for renting houses / places with the following functionalities.

1. This application will be used for renting houses for a user given check in - check out date period (kind of airbnb).
2. There will be 3 separate micro services.
    a. **Authenticator -** that will authenticate users.
    b. **Rental Processor -** that will include all api / services for rental management.
    c. **Rental Search -** that will include all api for searching houses, primarily handling booking requests and reviewing/rating houses. Booking requests will be forwarded to the **rental processor** provided API for actual booking request processing.
3. There will be two types of users. Admin & General registered users.
4. **Rental Processor** application will expose a list of api for following actions in rental management:
    a. Submit a house **as a host** for rent with proper description and address **by the registered user**. Newly submitted rental request will be in Pending status. **Proper validation** should be added for this submitted request (i.e required fields, valid address, logical price etc). A house with **Pending status** should not be available in search results.

b. Returns **registered user’s** submitted rental posts as a host (both list and single one in detail).
c. Approve & publish / Reject a rental post **by the Admin**. Approve & Publish action will make the rental post available for other user’s search results. Host will be notified through email. An api for returning pending rental posts should also be **available for the admin**.
d. Receiving a rental booking request from the **registered user**. Proper validation should be added for this submitted request (i.e house available for rental in the given Check In - Check out date range.) The house should not be already rented out for any part of the Check In - Check out date range.
e. Approve / Reject rental booking request **by the admin**. Proper email notification will be sent to the user who made the booking request. An api for returning pending booking requests should also be **available for the admin**.
f. Return **registered user**’s submitted booking requests (both list and single one in detail).
g. Cancel an already approved / pending booking request **by the registered user** who actually placed the booking. Canceling should be done at least 7 days before the start of the Check In date. On canceling the booking, house should be available on the user search result for canceled booking’s Check In - Check out date range.

5. **Rental Search** application will expose a list of api for following actions in user search / review management:
    a. Search a house with proper **search criteria by the registered user.** Search criteria should include address, date range, price range etc. Only the **matched houses that are available in the date range** will appear in the result. Users can also select a point in the map so all the **rental houses around this point’s 1km radius can be searched**. So the house's proper **latitude / longitude should be saved and processed properly**.
    b. Submit a request to book a house for a valid future **check In - check Out date period by the registered user.** The selected house should be available for booking in all parts of the requested check In - check Out date range. (No approved or Pending booking should be available for any part of the submitted date range for the selected house). This application should redirect this request to the **rental processor** and the response from the processor should be processed and returned (Inter service communication).
    c. Submit a review and rating by the general user. An user can submit a review / rating for a house that he has already rented and visited (booking date range is already passed).
6. Functionality of Admin will be
    a. Login using username / password.
    b. Create another Admin account.
    c. Can view a list of pending submitted rental posts.
    d. Can view a list of pending booking requests.
    e. Approve / Reject House rental submission request.

    f. Approve / Reject House booking request.
7. Registered Users have following functionality.
    a. Registration in the system with proper email verification.
    b. Log in with username / password.
    c. Submit a house with proper description / address for rental as a host.
    d. Search for a house to rent in a date period.
    e. Can book an available house found in the search result.
    f. Can cancel the user's own submitted pending / approved booking at least 7 days before the Check In date.
    g. Can write a review on the already booked house by that user.
    h. Can view the list of bookings that the user submitted.
    i. Can view the list of rental posts that the user submitted as a host.
8. **An admin user** should be created on application initialization if **no admin user exists** in the system.

Tools

All required domains, service, repositories, security checking, REST API endpoints will be included into a spring boot application that needs to be built with the following framework / tool / library support.

1. Gradle / Maven / Ant as build tool
2. Spring boot for auto configuration and embedded web container
3. Any relational database (either embedded or dockerized)
4. JPA as Object Relational Mapping tool (Hibernate as implementation)
5. Spring data for DAO layer Interfaces
6. Spring security for user authentication checking
7. Spring REST Controller for building API endpoints

Technical Requirements (Backend)

1. Although we have 3 separate microservices, we will have a single database for all services.
2. An api gateway application can be added for redirecting requests to proper microservice based on sub paths in the url. Nginx can also be used for this purpose.
3. Domains can be included in a library jar that will be added as dependency in all micro services.
4. Auth application will use **spring security** in a **stateless** way. On successful login users will be given an **JWT token where all details about the user will be included**. So no server side user session maintenance needed. Every other microservices will validate this token and check if the user is authorized to do the action. Authorization checking should be properly implemented. Users should not be allowed to do admin's job. Also one user should not be able to cancel another's booking.
5. Rental post submission should have proper address information. There should be a flexible way for searching a house by address for booking.

6. In booking action, concurrency must be handled properly so that the same house should not be booked by multiple persons.
7. **[Optional]** Hazelcast / Redis should be used for in memory caching purposes. Users' recent search criteria will be saved in cache, so if users revisit our site then the same criteria will be applied in filters.
8. **[For Bonus Points]** All services should be dockerized and running a single docker compose file will make all services available.
9. Following good coding standards will be highly appreciable.

Technical Requirements (Frontend)

If all the **given backend features** are done then the frontend should be implemented using Angular as view technology for the frontend. **We are not expecting any rich UI for this assignment**. We just need the frontend for this project only for reviewing the backend's proper functionality.