```java
package com.example.auth.service;

import com.example.auth.model.User;
import com.example.auth.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class UserService {
    @Autowired
    private UserRepository userRepository;

    public User registerUser(String name, String email, String password, String mobileNumber) {
        if (userRepository.findByEmail(email.toLowerCase()) != null) {
            throw new RuntimeException("Email already registered");
        }
        User user = new User();
        user.setName(name);
        user.setEmail(email.toLowerCase());
        user.setPassword(password); // Plain text password
        user.setMobileNumber(mobileNumber);
        return userRepository.save(user);
    }

    public User authenticateUser(String email, String password) {
        User user = userRepository.findByEmail(email.toLowerCase());
        if (user == null || !password.equals(user.getPassword())) {
            throw new RuntimeException("Invalid credentials");
        }
        return user;
    }

```

```
33      public void updateResetPasswordToken(String token
    , String email) {
34          User user = userRepository.findByEmail(email.
    toLowerCase());
35          if (user == null) throw new RuntimeException(
    "No user found with this email");
36          user.setResetPasswordToken(token);
37          userRepository.save(user);
38      }
39
40      public User getByResetPasswordToken(String token
    ) {
41          return userRepository.
    findByResetPasswordToken(token);
42      }
43
44      public void updatePassword(User user, String
    newPassword) {
45          user.setPassword(newPassword); // Plain text
    password
46          user.setResetPasswordToken(null);
47          userRepository.save(user);
48      }
49
50      public void deleteUserByEmail(String email) {
51          User user = userRepository.findByEmail(email.
    toLowerCase());
52          if (user == null) {
53              throw new RuntimeException("User not
    found with email: " + email);
54          }
55          userRepository.delete(user);
56      }
57
58  }
59
```

```java
package com.example.auth.service;

import com.example.auth.model.Owner;
import com.example.auth.repository.OwnerRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class OwnerService {

    @Autowired
    private OwnerRepository ownerRepository;

    // ✅ Always returns Owner or null
    public Owner findByEmail(String email) {
        return ownerRepository.findByEmail(email.toLowerCase()).orElse(null);
    }

    // ✅ Uses Optional to check if email already exists
    public Owner registerOwner(String name, String email, String password, String mobileNumber) {
        if (ownerRepository.findByEmail(email.toLowerCase()).isPresent()) {
            throw new RuntimeException("Email already registered");
        }
        Owner owner = new Owner();
        owner.setName(name);
        owner.setEmail(email.toLowerCase());
        owner.setMobileNumber(mobileNumber);
        owner.setPassword(password); // ⚠️ plain text (not secure)
        return ownerRepository.save(owner);
    }

    // ✅ Optional + filter for authentication
    public Owner authenticateOwner(String email, String password) {
```

```
34          return ownerRepository.findByEmail(email.
   toLowerCase())
35                  .filter(o -> password.equals(o.
   getPassword()))
36                  .orElseThrow(() -> new
   RuntimeException("Invalid credentials"));
37      }
38
39      // ✅ Uses Optional safely
40      public void updateResetPasswordToken(String token
   , String email) {
41          Owner owner = ownerRepository.findByEmail(
   email.toLowerCase())
42                  .orElseThrow(() -> new
   RuntimeException("No owner found with this email"));
43          owner.setResetPasswordToken(token);
44          ownerRepository.save(owner);
45      }
46
47      public Owner getByResetPasswordToken(String token
   ) {
48          return ownerRepository.
   findByResetPasswordToken(token);
49      }
50
51      public void updatePassword(Owner owner, String
   newPassword) {
52          owner.setPassword(newPassword); // ⚠️ plain
   text (not secure)
53          owner.setResetPasswordToken(null);
54          ownerRepository.save(owner);
55      }
56 }
57
```

```java
package com.example.auth.service;


import com.example.auth.model.Booking;
import com.example.auth.model.Property;
import com.example.auth.repository.BookingRepository;
import com.example.auth.repository.PropertyRepository;
import org.springframework.stereotype.Service;

import java.util.Optional;

@Service
public class BookingService {
    private final BookingRepository bookingRepository;
    private final PropertyRepository propertyRepository;

    public BookingService(BookingRepository bookingRepository, PropertyRepository propertyRepository) {
        this.bookingRepository = bookingRepository;
        this.propertyRepository = propertyRepository;
    }

    // Create a booking request
    public Booking bookProperty(Long propertyId, String userEmail) {
        Property property = propertyRepository.findById(propertyId)
                .orElseThrow(() -> new RuntimeException("Property not found"));

        Booking booking = new Booking();
        booking.setProperty(property);
        booking.setUserEmail(userEmail);
        booking.setStatus("PENDING");

        return bookingRepository.save(booking);
    }
```

```
34
35      // Update booking status (Approve / Reject)
36      public Booking updateBookingStatus(Long bookingId
    , String status) {
37          return bookingRepository.findById(bookingId).
    map(b -> {
38              b.setStatus(status.toUpperCase());
39              return bookingRepository.save(b);
40          }).orElseThrow(() -> new RuntimeException("
    Booking not found"));
41      }
42
43      // Fetch booking by ID
44      public Optional<Booking> getBookingById(Long
    bookingId) {
45          return bookingRepository.findById(bookingId);
46      }
47  }
48
```

```java
package com.example.auth.service;

import com.example.auth.model.Owner;
import com.example.auth.model.Property;
import com.example.auth.repository.PropertyRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;

@Service
public class PropertyService {
    @Autowired
    private PropertyRepository propertyRepository;

    public Property addProperty(Property property) {
        return propertyRepository.save(property);
    }

    // ✅ Now accepts Owner directly
    public List<Property> getPropertiesByOwner(Owner owner) {
        return propertyRepository.findByOwner(owner);
    }

    public Optional<Property> getPropertyById(Long id) {
        return propertyRepository.findById(id);
    }

    public Property updateProperty(Long id, Property updatedProperty) {
        Optional<Property> optionalProperty = propertyRepository.findById(id);
        if (optionalProperty.isEmpty()) {
            throw new RuntimeException("Property not found");
        }
```

```
35          Property property = optionalProperty.get();
36          property.setTitle(updatedProperty.getTitle
    ());
37          property.setDescription(updatedProperty.
    getDescription());
38          property.setLocation(updatedProperty.
    getLocation());
39          property.setPrice(updatedProperty.getPrice
    ());
40          property.setType(updatedProperty.getType());
41          property.setActive(updatedProperty.isActive
    ());
42          return propertyRepository.save(property);
43      }
44
45      public void deleteProperty(Long id) {
46          propertyRepository.deleteById(id);
47      }
48
49      public List<Property> getAllProperties() {
50          return propertyRepository.findAll();
51      }
52  }
53
```