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


import com.example.auth.model.User;
import com.example.auth.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.Map;
import java.util.UUID;

@RestController
@RequestMapping("/api")
@CrossOrigin(origins = "*")
public class AuthController {
    @Autowired
    private UserService userService;

    @PostMapping("/register")
    public ResponseEntity<?> register(@RequestBody Map<String, String> req) {
        try {
            userService.registerUser(req.get("name"), req.get("email"), req.get("password"), req.get("mobileNumber"));
            return ResponseEntity.ok(Map.of("message", "User registered successfully."));
        } catch (Exception e) {
            return ResponseEntity.badRequest().body(Map.of("error", e.getMessage()));
        }
    }

    @PostMapping("/login")
    public ResponseEntity<?> login(@RequestBody Map<String, String> req) {
        try {
            User user = userService.authenticateUser(req.get("email"), req.get("password"));
            return ResponseEntity.ok(Map.of("message", "Login successful.", "name", user.getName()));
        } catch (Exception e) {
            return ResponseEntity.badRequest().body(Map.of("error", e.getMessage()));
        }
    }

    @PostMapping("/forgot-password")
    public ResponseEntity<?> forgotPassword(@RequestBody Map<String, String> req) {
        String email = req.get("email");
        try {
            String token = UUID.randomUUID().toString();
            userService.updateResetPasswordToken(token, email);
            // In real apps: send the token via email
            return ResponseEntity.ok(Map.of("message", "Reset token generated.", "token", token));
        } catch (Exception e) {
            return ResponseEntity.badRequest().body(Map.of("error", e.getMessage()));
        }
    }

    @PostMapping("/reset-password")
    public ResponseEntity<?> resetPassword(@RequestParam("token") String token, @RequestBody Map<String, String> req) {
        try {
            User user = userService.getByResetPasswordToken(token);
            if (user == null) {
                return ResponseEntity.badRequest().body(Map.of("error", "Invalid or expired reset token."));
            }
            userService.updatePassword(user, req.get("newPassword"));
            return ResponseEntity.ok(Map.of("message"
, "Password updated successfully."));
        } catch (Exception e) {
            return ResponseEntity.badRequest().body(Map.of("error", e.getMessage()));
        }
    }
    @DeleteMapping("/delete/{email}")
    public ResponseEntity<?> deleteUser(@PathVariable String email) {
        try {
            userService.deleteUserByEmail(email);
            return ResponseEntity.ok(Map.of("message", "User deleted successfully."));
        } catch (Exception e) {
            return ResponseEntity.badRequest().body(Map.of("error", e.getMessage()));
        }
    }

}

```

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

import com.example.auth.model.Owner;
import com.example.auth.model.User;
import com.example.auth.repository.OwnerRepository;
import com.example.auth.repository.UserRepository;
import org.springframework.beans.factory.annotation.
Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/admin")
public class AdminController {

    @Autowired
    private UserRepository userRepository;

    @Autowired
    private OwnerRepository ownerRepository;

    @GetMapping("/users")
    public List<User> getAllUsers() {
        return userRepository.findAll();
    }

    @GetMapping("/owners")
    public List<Owner> getAllOwners() {
        return ownerRepository.findAll();
    }
}

```

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

import com.example.auth.model.Booking;
import com.example.auth.service.BookingService;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.Map;

@RestController
@RequestMapping("/api/bookings")
public class BookingController {

    private final BookingService bookingService;

    public BookingController(BookingService bookingService) {
        this.bookingService = bookingService;
    }

    // Create a booking
    @PostMapping("/book/{propertyId}")
    public ResponseEntity<?> bookProperty(
            @PathVariable Long propertyId,
            @RequestParam String userEmail
    ) {
        try {
            Booking booking = bookingService.bookProperty(propertyId, userEmail);
            return ResponseEntity.ok(booking);
        } catch (Exception e) {
            return ResponseEntity.badRequest().body(Map.of("error", e.getMessage()));
        }
    }

    // Approve / Reject booking
    @PutMapping("/{bookingId}/status")
    public ResponseEntity<?> updateBookingStatus(
            @PathVariable Long bookingId,
            @RequestParam String status
    ) {
        try {
            Booking updated = bookingService.updateBookingStatus(bookingId, status);
            return ResponseEntity.ok(updated);
        } catch (Exception e) {
            return ResponseEntity.badRequest().body(Map.of("error", e.getMessage()));
        }
    }

    // Get booking by ID
    @GetMapping("/{bookingId}")
    public ResponseEntity<?> getBooking(@PathVariable Long bookingId) {
        return bookingService.getBookingById(bookingId)
                .<ResponseEntity<?>>map(ResponseEntity::ok)
                .orElseGet(() -> ResponseEntity.status(404).body(Map.of("error", "Not found")));
    }
}

```

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

import com.example.auth.dto.PropertyDTO;
import com.example.auth.exception.BadRequestException;
import com.example.auth.exception.ResourceNotFoundException;
import com.example.auth.model.Owner;
import com.example.auth.model.Property;
import com.example.auth.service.OwnerService;
import com.example.auth.service.PropertyService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;

import java.io.IOException;
import java.nio.file.*;
import java.security.Principal;
import java.util.List;
import java.util.UUID;

@RestController
@RequestMapping("/api/properties")
public class PropertyController {

    @Autowired
    private PropertyService propertyService;

    @Autowired
    private OwnerService ownerService;

    @PostMapping("/create-with-image")
    public ResponseEntity<PropertyDTO> addPropertyWithImage(
            @RequestParam String title,
            @RequestParam String description,
            @RequestParam String location,
            @RequestParam Double price,
            @RequestParam String type,
            @RequestParam(value = "file", required = false) MultipartFile file,
            Principal principal
    ) {
        try {
            Property property = new Property();
            property.setTitle(title);
            property.setDescription(description);
            property.setLocation(location);
            property.setPrice(price);
            property.setType(type);
            property.setActive(true);

            // resolve owner (Principal or testing fallback)
            Owner owner = null;
            if (principal != null) {
                owner = ownerService.findByEmail(principal.getName());
            }
            if (owner == null) {
                owner = ownerService.findByEmail("owner@test.com"); // ensure exists or remove after auth
            }
            if (owner == null) {
                throw new BadRequestException("No valid owner found. Login as an owner or create 'owner@test.com' for testing.");
            }
            property.setOwner(owner);

            // save file
            if (file != null && !file.isEmpty()) {
                Path uploadDir = Paths.get("uploads");
                if (!Files.exists(uploadDir)) Files.createDirectories(uploadDir);

                String fileName = UUID.randomUUID
```

```java
() + "_" + file.getOriginalFilename();
                Path filePath = uploadDir.resolve(fileName);
                Files.copy(file.getInputStream(), filePath, StandardCopyOption.REPLACE_EXISTING);

                property.setImageUrl("/uploads/" + fileName);
            }

            Property saved = propertyService.addProperty(property);
            return ResponseEntity.ok(PropertyDTO.from(saved));

        } catch (IOException e) {
            throw new BadRequestException("File upload failed: " + e.getMessage());
        } catch (Exception e) {
            // let GlobalExceptionHandler turn this into 500 JSON
            throw e;
        }
    }

    @GetMapping("/all")
    public ResponseEntity<List<PropertyDTO>> getAllProperties() {
        List<PropertyDTO> list = propertyService.getAllProperties()
                .stream().map(PropertyDTO::from).toList();
        return ResponseEntity.ok(list);
    }

    @GetMapping("/mine")
    public ResponseEntity<List<PropertyDTO>> getMyProperties(Principal principal) {
        Owner owner = null;
        if (principal != null) {
            owner = ownerService.findByEmail(
principal.getName());
        }
        if (owner == null) {
            owner = ownerService.findByEmail("owner@test.com"); // fallback for local testing
        }
        if (owner == null) {
            throw new BadRequestException("No valid owner found.");
        }

        List<PropertyDTO> list = propertyService.getPropertiesByOwner(owner)
                .stream().map(PropertyDTO::from).toList();
        return ResponseEntity.ok(list);
    }

    @GetMapping("/{id}")
    public ResponseEntity<PropertyDTO> getPropertyById(@PathVariable Long id) {
        Property p = propertyService.getPropertyById(id)
                .orElseThrow(() -> new ResourceNotFoundException("Property not found"));
        return ResponseEntity.ok(PropertyDTO.from(p));
    }

    @PutMapping("/{id}")
    public ResponseEntity<PropertyDTO> updateProperty(@PathVariable Long id, @RequestBody Property property) {
        Property updated = propertyService.updateProperty(id, property);
        return ResponseEntity.ok(PropertyDTO.from(updated));
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteProperty(@
PathVariable Long id) {
        propertyService.deleteProperty(id);
        return ResponseEntity.noContent().build();
    }
}

```

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

import com.example.auth.model.Owner;
import com.example.auth.service.OwnerService;
import org.springframework.beans.factory.annotation.
Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.Map;
import java.util.UUID;


@RestController
@RequestMapping("/api/owner")
@CrossOrigin(origins = "http://localhost:3000")
public class OwnerAuthController {

    @Autowired
    private OwnerService ownerService;

    @PostMapping("/register")
    public Owner register(@RequestBody Owner owner) {
        return ownerService.registerOwner(owner.
getName(), owner.getEmail(), owner.getPassword(),
owner.getMobileNumber());
    }

    @PostMapping("/login")
    public ResponseEntity<?> login(@RequestBody Map<
String, String> req) {
        try {
            Owner owner = ownerService.
authenticateOwner(req.get("email"), req.get("password
"));
            return ResponseEntity.ok(Map.of(
                    "message", "Login successful.",
                    "name", owner.getName()
            ));
        } catch (Exception e) {
            return ResponseEntity.badRequest().body(
Map.of("error", e.getMessage()));
        }
    }

    @PostMapping("/forgot-password")
    public ResponseEntity<?> forgotPassword(@RequestParam String email) {
        try {
            String token = UUID.randomUUID().toString();
            ownerService.updateResetPasswordToken(token, email);
            return ResponseEntity.ok(Map.of(
                    "message", "Reset token generated.",
                    "token", token
            ));
        } catch (Exception e) {
            return ResponseEntity.badRequest().body(Map.of(
                    "error", e.getMessage()
            ));
        }
    }


    @PostMapping("/reset-password")
    public String resetPassword(@RequestParam String token, @RequestParam String newPassword) {
        Owner owner = ownerService.getByResetPasswordToken(token);
        if (owner == null) {
            return "Invalid token";
        }
        ownerService.updatePassword(owner, newPassword);
        return "Password updated successfully";
    }
}

```