

```
1 package com.example.auth.controller;
2
3
4 import com.example.auth.model.User;
5 import com.example.auth.service.UserService;
6 import org.springframework.beans.factory.annotation.
    Autowired;
7 import org.springframework.http.ResponseEntity;
8 import org.springframework.web.bind.annotation.*;
9
10 import java.util.Map;
11 import java.util.UUID;
12
13 @RestController
14 @RequestMapping("/api")
15 @CrossOrigin(origins = "*")
16 public class AuthController {
17     @Autowired
18     private UserService userService;
19
20     @PostMapping("/register")
21     public ResponseEntity<?> register(@RequestBody
    Map<String, String> req) {
22         try {
23             userService.registerUser(req.get("name"
    ), req.get("email"), req.get("password"), req.get("
    mobileNumber"));
24             return ResponseEntity.ok(Map.of("message"
    , "User registered successfully.));
25         } catch (Exception e) {
26             return ResponseEntity.badRequest().body(
    Map.of("error", e.getMessage()));
27         }
28     }
29
30     @PostMapping("/login")
31     public ResponseEntity<?> login(@RequestBody Map<
    String, String> req) {
32         try {
33             User user = userService.authenticateUser(
    req.get("email"), req.get("password"));
```

```
34         return ResponseEntity.ok(Map.of("message"
35     , "Login successful.", "name", user.getName()));
36     } catch (Exception e) {
37         return ResponseEntity.badRequest().body(
38     Map.of("error", e.getMessage()));
39     }
40     }
41     @PostMapping("/forgot-password")
42     public ResponseEntity<?> forgotPassword(@
43     RequestBody Map<String, String> req) {
44         String email = req.get("email");
45         try {
46             String token = UUID.randomUUID().toString
47         ();
48             userService.updateResetPasswordToken(
49         token, email);
50             // In real apps: send the token via email
51             return ResponseEntity.ok(Map.of("message"
52     , "Reset token generated.", "token", token));
53         } catch (Exception e) {
54             return ResponseEntity.badRequest().body(
55     Map.of("error", e.getMessage()));
56         }
57     }
58     @PostMapping("/reset-password")
59     public ResponseEntity<?> resetPassword(@
60     RequestParam("token") String token, @RequestBody Map<
61     String, String> req) {
62         try {
63             User user = userService.
64         getByResetPasswordToken(token);
65             if (user == null) {
66                 return ResponseEntity.badRequest().
67         body(Map.of("error", "Invalid or expired reset token
68     ."));
69             }
70             userService.updatePassword(user, req.get(
71     "newPassword"));
72             return ResponseEntity.ok(Map.of("message"
```

```
61 , "Password updated successfully.));  
62         } catch (Exception e) {  
63             return ResponseEntity.badRequest().body(  
        Map.of("error", e.getMessage()));  
64         }  
65     }  
66     @DeleteMapping("/delete/{email}")  
67     public ResponseEntity<?> deleteUser(@  
        PathVariable String email) {  
68         try {  
69             userService.deleteUserByEmail(email);  
70             return ResponseEntity.ok(Map.of("message  
        ", "User deleted successfully.));  
71         } catch (Exception e) {  
72             return ResponseEntity.badRequest().body(  
        Map.of("error", e.getMessage()));  
73         }  
74     }  
75  
76 }  
77
```

```
1 package com.example.auth.controller;
2
3 import com.example.auth.model.Owner;
4 import com.example.auth.model.User;
5 import com.example.auth.repository.OwnerRepository;
6 import com.example.auth.repository.UserRepository;
7 import org.springframework.beans.factory.annotation.
    Autowired;
8 import org.springframework.web.bind.annotation.*;
9
10 import java.util.List;
11
12 @RestController
13 @RequestMapping("/admin")
14 public class AdminController {
15
16     @Autowired
17     private UserRepository userRepository;
18
19     @Autowired
20     private OwnerRepository ownerRepository;
21
22     @GetMapping("/users")
23     public List<User> getAllUsers() {
24         return userRepository.findAll();
25     }
26
27     @GetMapping("/owners")
28     public List<Owner> getAllOwners() {
29         return ownerRepository.findAll();
30     }
31 }
32
```

```
1 package com.example.auth.controller;
2
3 import com.example.auth.model.Booking;
4 import com.example.auth.service.BookingService;
5 import org.springframework.http.ResponseEntity;
6 import org.springframework.web.bind.annotation.*;
7
8 import java.util.Map;
9
10 @RestController
11 @RequestMapping("/api/bookings")
12 public class BookingController {
13
14     private final BookingService bookingService;
15
16     public BookingController(BookingService
bookingService) {
17         this.bookingService = bookingService;
18     }
19
20     // Create a booking
21     @PostMapping("/book/{propertyId}")
22     public ResponseEntity<?> bookProperty(
23         @PathVariable Long propertyId,
24         @RequestParam String userEmail
25     ) {
26         try {
27             Booking booking = bookingService.
bookProperty(propertyId, userEmail);
28             return ResponseEntity.ok(booking);
29         } catch (Exception e) {
30             return ResponseEntity.badRequest().body(
Map.of("error", e.getMessage()));
31         }
32     }
33
34     // Approve / Reject booking
35     @PutMapping("/{bookingId}/status")
36     public ResponseEntity<?> updateBookingStatus(
37         @PathVariable Long bookingId,
38         @RequestParam String status
```

```
39     ) {
40         try {
41             Booking updated = bookingService.
updateBookingStatus(bookingId, status);
42             return ResponseEntity.ok(updated);
43         } catch (Exception e) {
44             return ResponseEntity.badRequest().body(
Map.of("error", e.getMessage()));
45         }
46     }
47
48     // Get booking by ID
49     @GetMapping("/{bookingId}")
50     public ResponseEntity<?> getBooking(@PathVariable
Long bookingId) {
51         return bookingService.getBookingById(
bookingId)
52             .<ResponseEntity<?>>map(
ResponseEntity::ok)
53             .orElseGet(() -> ResponseEntity.
status(404).body(Map.of("error", "Not found")));
54     }
55 }
56
```

```
1 package com.example.auth.controller;
2
3 import com.example.auth.dto.PropertyDTO;
4 import com.example.auth.exception.BadRequestException
5 ;
6 import com.example.auth.exception.
  ResourceNotFoundException;
7 import com.example.auth.model.Owner;
8 import com.example.auth.model.Property;
9 import com.example.auth.service.OwnerService;
10 import com.example.auth.service.PropertyService;
11 import org.springframework.beans.factory.annotation.
  Autowired;
12 import org.springframework.http.ResponseEntity;
13 import org.springframework.web.bind.annotation.*;
14 import org.springframework.web.multipart.
  MultipartFile;
15
16 import java.io.IOException;
17 import java.nio.file.*;
18 import java.security.Principal;
19 import java.util.List;
20 import java.util.UUID;
21
22 @RestController
23 @RequestMapping("/api/properties")
24 public class PropertyController {
25     @Autowired
26     private PropertyService propertyService;
27
28     @Autowired
29     private OwnerService ownerService;
30
31     @PostMapping("/create-with-image")
32     public ResponseEntity<PropertyDTO>
  addPropertyWithImage(
33         @RequestParam String title,
34         @RequestParam String description,
35         @RequestParam String location,
36         @RequestParam Double price,
```

```
37         @RequestParam String type,
38         @RequestParam(value = "file", required =
    false) MultipartFile file,
39         Principal principal
40     ) {
41         try {
42             Property property = new Property();
43             property.setTitle(title);
44             property.setDescription(description);
45             property.setLocation(location);
46             property.setPrice(price);
47             property.setType(type);
48             property.setActive(true);
49
50             // resolve owner (Principal or testing
fallback)
51             Owner owner = null;
52             if (principal != null) {
53                 owner = ownerService.findByEmail(
    principal.getName());
54             }
55             if (owner == null) {
56                 owner = ownerService.findByEmail("
    owner@test.com"); // ensure exists or remove after
auth
57             }
58             if (owner == null) {
59                 throw new BadRequestException("No
    valid owner found. Login as an owner or create 'owner
    @test.com' for testing.");
60             }
61             property.setOwner(owner);
62
63             // save file
64             if (file != null && !file.isEmpty()) {
65                 Path uploadDir = Paths.get("uploads"
    );
66                 if (!Files.exists(uploadDir)) Files.
    createDirectories(uploadDir);
67
68                 String fileName = UUID.randomUUID
```



```

68 () + "_" + file.getOriginalFilename();
69         Path filePath = uploadDir.resolve(
        fileName);
70         Files.copy(file.getInputStream(),
        filePath, StandardCopyOption.REPLACE_EXISTING);
71
72         property.setImageUrl("/uploads/" +
        fileName);
73     }
74
75     Property saved = propertyService.
        addProperty(property);
76     return ResponseEntity.ok(PropertyDTO.
        from(saved));
77
78     } catch (IOException e) {
79         throw new BadRequestException("File
        upload failed: " + e.getMessage());
80     } catch (Exception e) {
81         // let GlobalExceptionHandler turn this
        into 500 JSON
82         throw e;
83     }
84 }
85
86 @GetMapping("/all")
87 public ResponseEntity<List<PropertyDTO>>
        getAllProperties() {
88     List<PropertyDTO> list = propertyService.
        getAllProperties()
89         .stream().map(PropertyDTO::from).
        toList();
90     return ResponseEntity.ok(list);
91 }
92
93 @GetMapping("/mine")
94 public ResponseEntity<List<PropertyDTO>>
        getMyProperties(Principal principal) {
95     Owner owner = null;
96     if (principal != null) {
97         owner = ownerService.findByEmail(

```

```

97 principal.getName());
98     }
99     if (owner == null) {
100         owner = ownerService.findByEmail("owner@
test.com"); // fallback for local testing
101     }
102     if (owner == null) {
103         throw new BadRequestException("No valid
owner found.");
104     }
105
106     List<PropertyDTO> list = propertyService.
getPropertiesByOwner(owner)
107         .stream().map(PropertyDTO::from).
toList();
108     return ResponseEntity.ok(list);
109 }
110
111 @GetMapping("/{id}")
112 public ResponseEntity<PropertyDTO>
getPropertyById(@PathVariable Long id) {
113     Property p = propertyService.getPropertyById
(id)
114         .orElseThrow(() -> new
ResourceNotFoundException("Property not found"));
115     return ResponseEntity.ok(PropertyDTO.from(p
));
116 }
117
118 @PutMapping("/{id}")
119 public ResponseEntity<PropertyDTO>
updateProperty(@PathVariable Long id, @RequestBody
Property property) {
120     Property updated = propertyService.
updateProperty(id, property);
121     return ResponseEntity.ok(PropertyDTO.from(
updated));
122 }
123
124 @DeleteMapping("/{id}")
125 public ResponseEntity<Void> deleteProperty(@

```

```
125 PathVariable Long id) {  
126     propertyService.deleteProperty(id);  
127     return ResponseEntity.noContent().build();  
128 }  
129 }  
130
```

```
1 package com.example.auth.controller;
2
3 import com.example.auth.model.Owner;
4 import com.example.auth.service.OwnerService;
5 import org.springframework.beans.factory.annotation.
    Autowired;
6 import org.springframework.http.ResponseEntity;
7 import org.springframework.web.bind.annotation.*;
8
9 import java.util.Map;
10 import java.util.UUID;
11
12
13 @RestController
14 @RequestMapping("/api/owner")
15 @CrossOrigin(origins = "http://localhost:3000")
16 public class OwnerAuthController {
17
18     @Autowired
19     private OwnerService ownerService;
20
21     @PostMapping("/register")
22     public Owner register(@RequestBody Owner owner) {
23         return ownerService.registerOwner(owner.
    getName(), owner.getEmail(), owner.getPassword(),
    owner.getMobileNumber());
24     }
25
26     @PostMapping("/login")
27     public ResponseEntity<?> login(@RequestBody Map<
    String, String> req) {
28         try {
29             Owner owner = ownerService.
    authenticateOwner(req.get("email"), req.get("password
    "));
30             return ResponseEntity.ok(Map.of(
31                 "message", "Login successful.",
32                 "name", owner.getName()
33             ));
34         } catch (Exception e) {
35             return ResponseEntity.badRequest().body(
```

```
35 Map.of("error", e.getMessage())));
36     }
37 }
38
39 @PostMapping("/forgot-password")
40 public ResponseEntity<?> forgotPassword(@
RequestParam String email) {
41     try {
42         String token = UUID.randomUUID().toString
43 ();
44         ownerService.updateResetPasswordToken(
45 token, email);
46         return ResponseEntity.ok(Map.of(
47             "message", "Reset token generated
48 .",
49             "token", token
50 ));
51     } catch (Exception e) {
52         return ResponseEntity.badRequest().body(
53 Map.of(
54             "error", e.getMessage()
55 ));
56     }
57 }
58
59 @PostMapping("/reset-password")
60 public String resetPassword(@RequestParam String
61 token, @RequestParam String newPassword) {
62     Owner owner = ownerService.
63 getByResetPasswordToken(token);
64     if (owner == null) {
65         return "Invalid token";
66     }
67     ownerService.updatePassword(owner,
68 newPassword);
69     return "Password updated successfully";
70 }
```