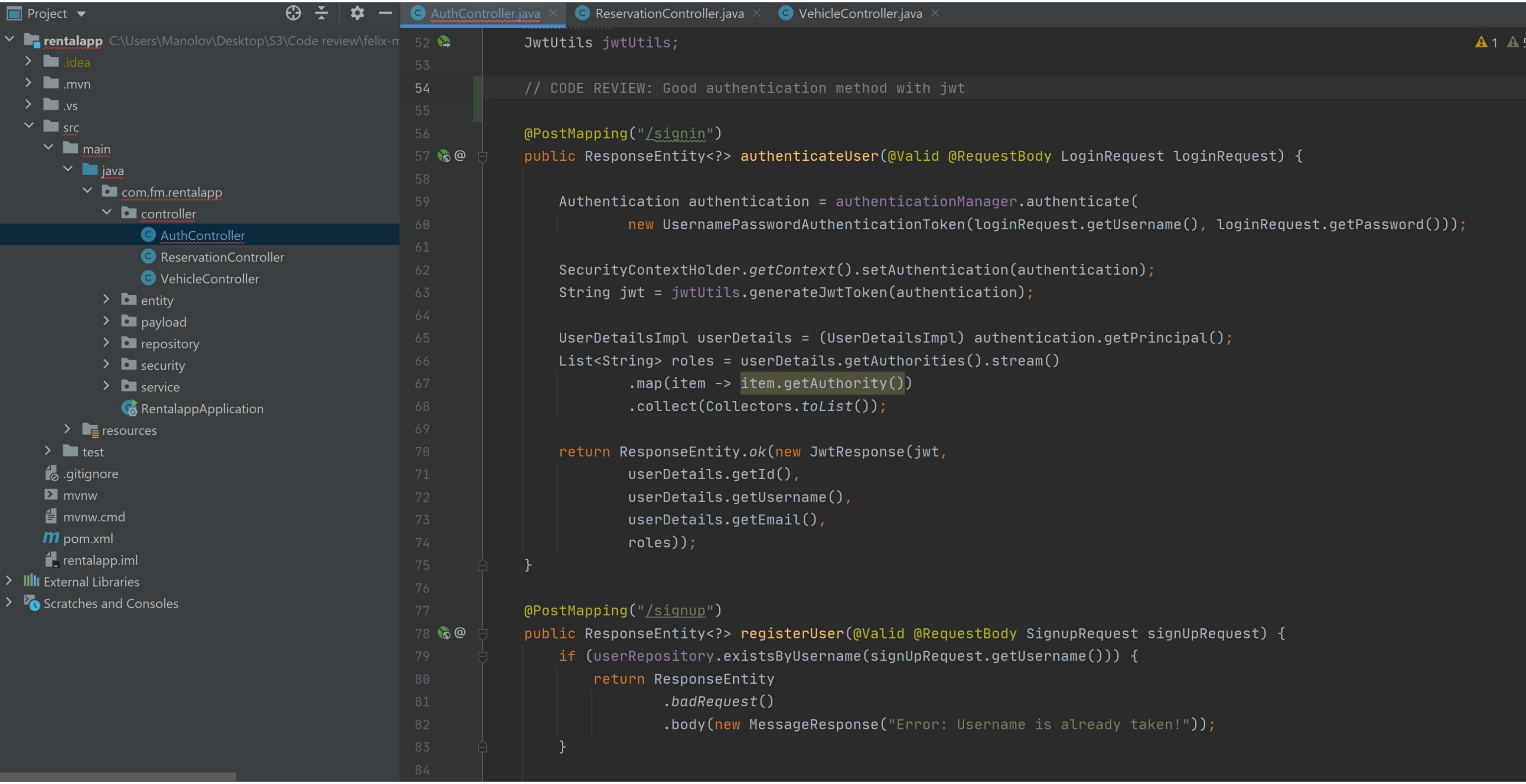
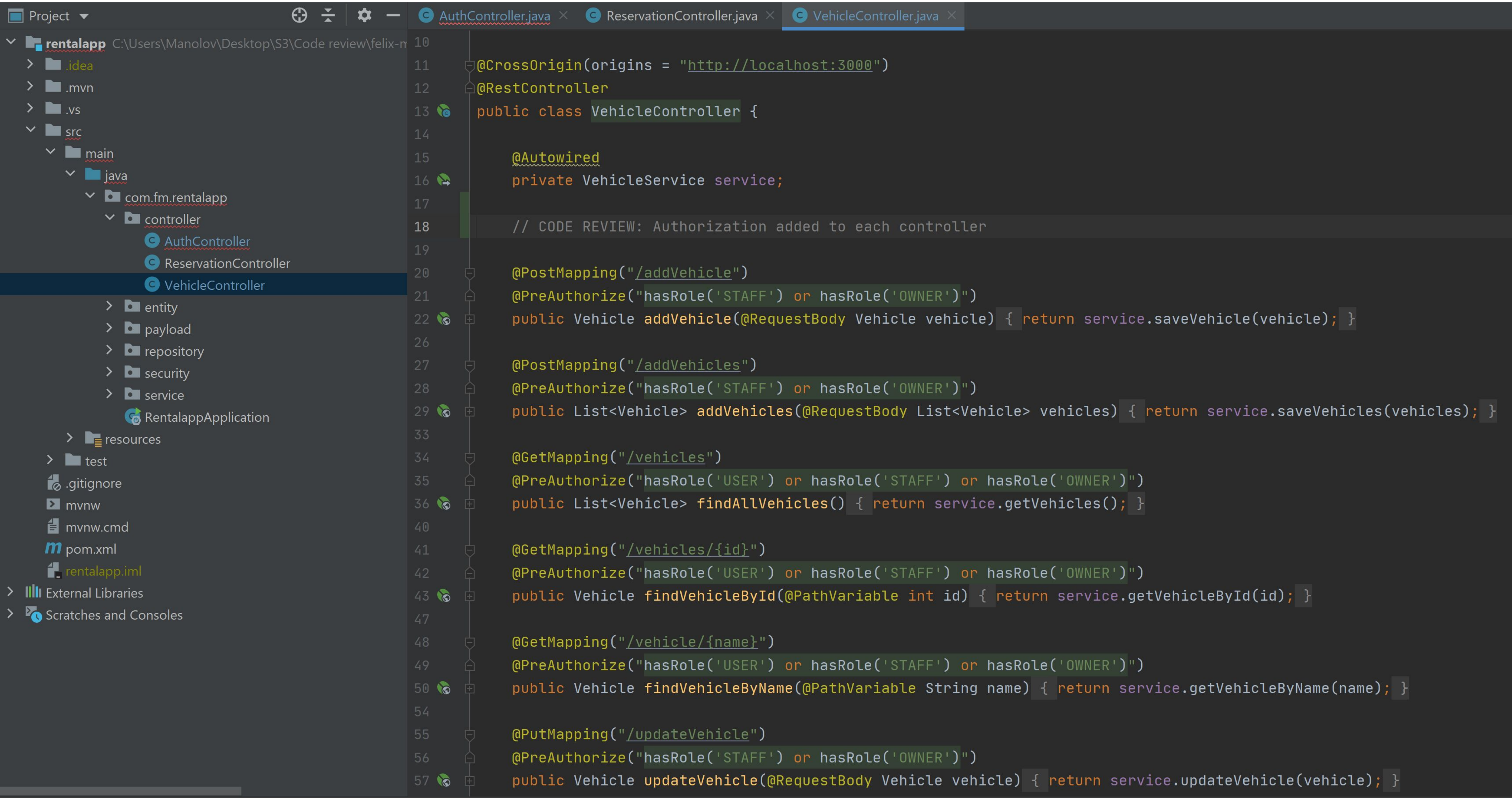
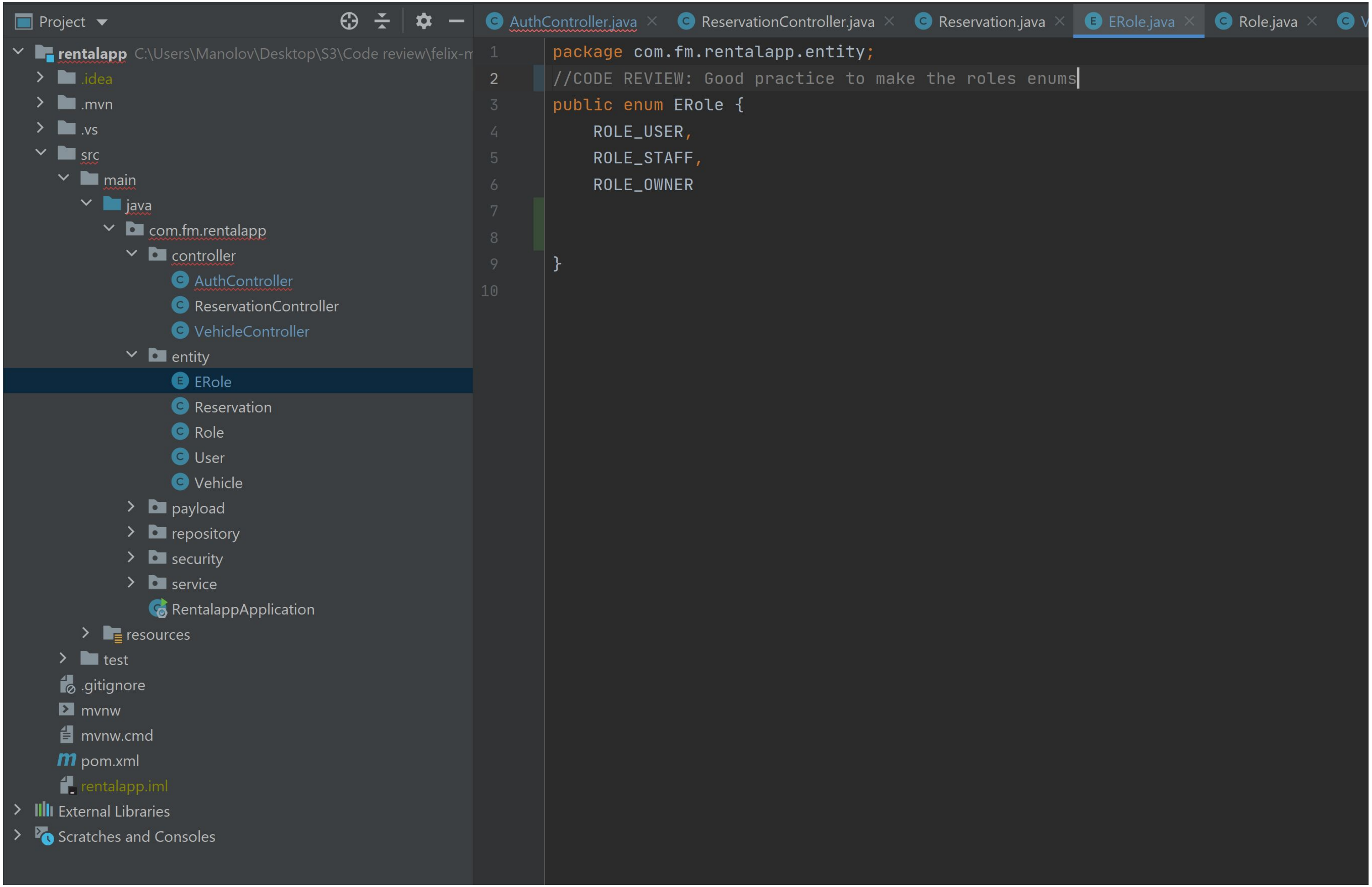


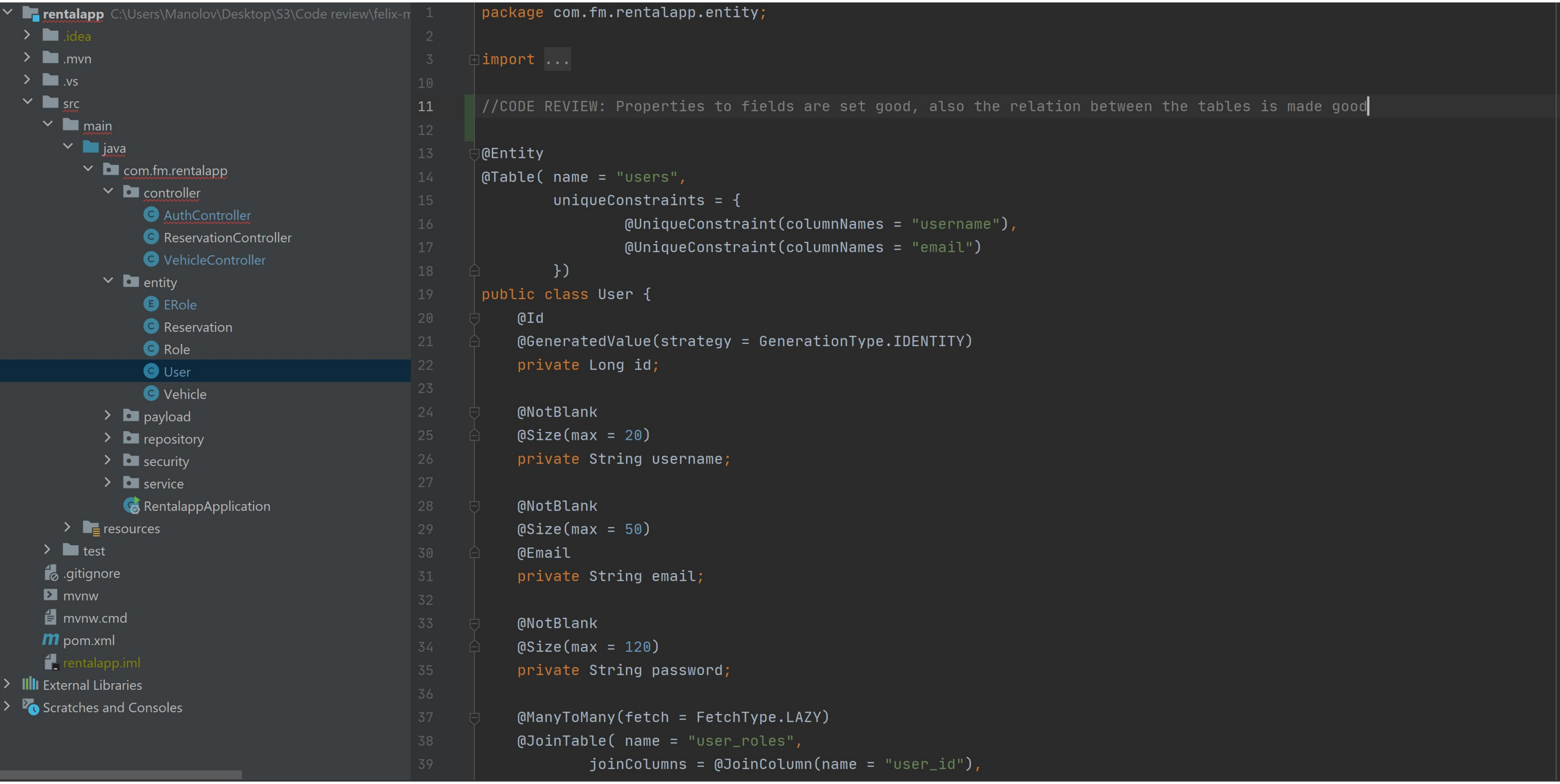
```java
    JwtUtils jwtUtils;

    // CODE REVIEW: Good authentication method with jwt

    @PostMapping("/signin")
    public ResponseEntity<?> authenticateUser(@Valid @RequestBody LoginRequest loginRequest) {

        Authentication authentication = authenticationManager.authenticate(
                new UsernamePasswordAuthenticationToken(loginRequest.getUsername(), loginRequest.getPassword()));

        SecurityContextHolder.getContext().setAuthentication(authentication);
        String jwt = jwtUtils.generateJwtToken(authentication);

        UserDetailsImpl userDetails = (UserDetailsImpl) authentication.getPrincipal();
        List<String> roles = userDetails.getAuthorities().stream()
                .map(item -> item.getAuthority())
                .collect(Collectors.toList());

        return ResponseEntity.ok(new JwtResponse(jwt,
                userDetails.getId(),
                userDetails.getUsername(),
                userDetails.getEmail(),
                roles));
    }

    @PostMapping("/signup")
    public ResponseEntity<?> registerUser(@Valid @RequestBody SignupRequest signUpRequest) {
        if (userRepository.existsByUsername(signUpRequest.getUsername())) {
            return ResponseEntity
                    .badRequest()
                    .body(new MessageResponse("Error: Username is already taken!"));
        }

```



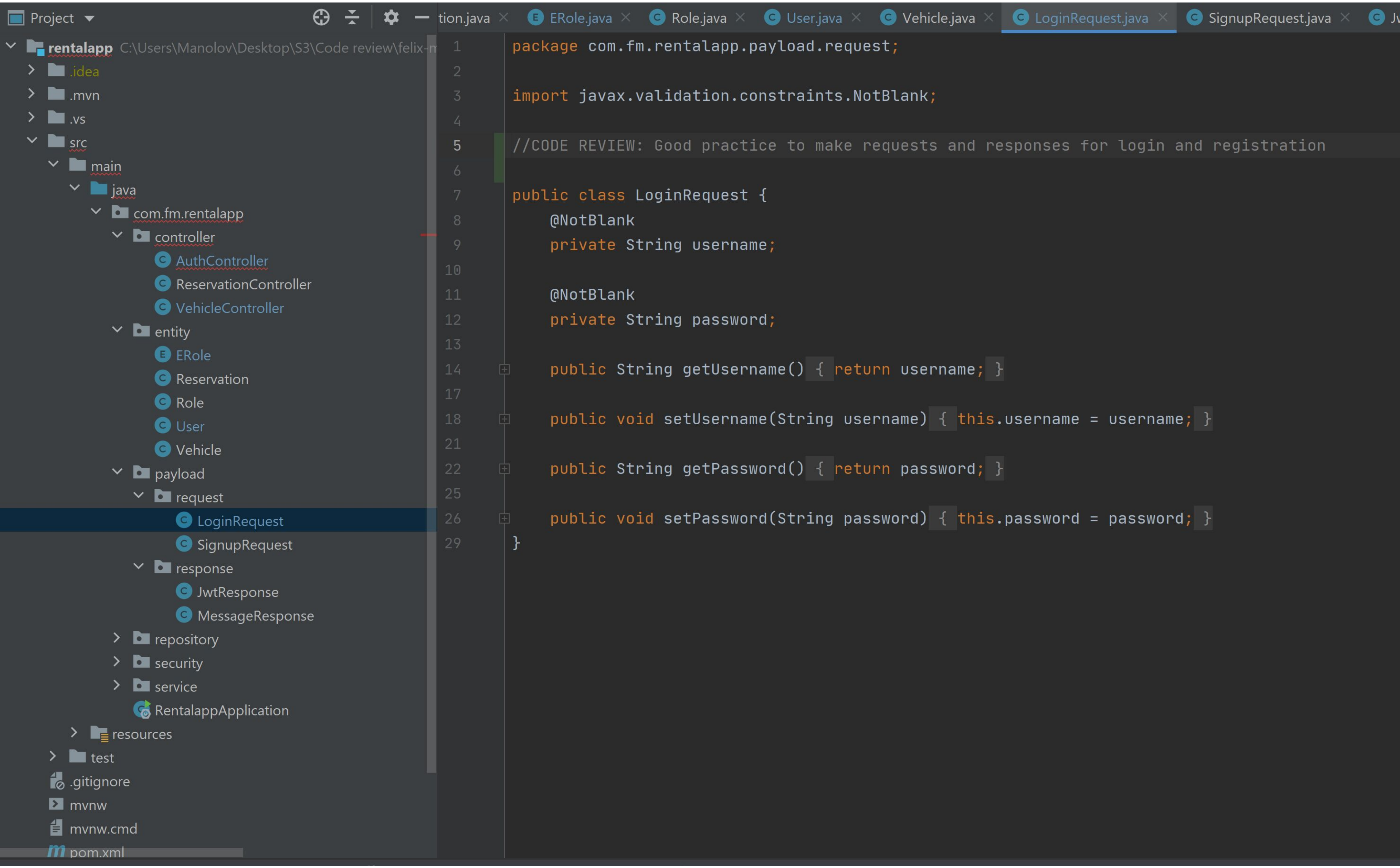
```java

@CrossOrigin(origins = "http://localhost:3000")
@RestController
public class VehicleController {

    @Autowired
    private VehicleService service;

    // CODE REVIEW: Authorization added to each controller

    @PostMapping("/addVehicle")
    @PreAuthorize("hasRole('STAFF') or hasRole('OWNER')")
    public Vehicle addVehicle(@RequestBody Vehicle vehicle) { return service.saveVehicle(vehicle); }

    @PostMapping("/addVehicles")
    @PreAuthorize("hasRole('STAFF') or hasRole('OWNER')")
    public List<Vehicle> addVehicles(@RequestBody List<Vehicle> vehicles) { return service.saveVehicles(vehicles); }

    @GetMapping("/vehicles")
    @PreAuthorize("hasRole('USER') or hasRole('STAFF') or hasRole('OWNER')")
    public List<Vehicle> findAllVehicles() { return service.getVehicles(); }

    @GetMapping("/vehicles/{id}")
    @PreAuthorize("hasRole('USER') or hasRole('STAFF') or hasRole('OWNER')")
    public Vehicle findVehicleById(@PathVariable int id) { return service.getVehicleById(id); }

    @GetMapping("/vehicle/{name}")
    @PreAuthorize("hasRole('USER') or hasRole('STAFF') or hasRole('OWNER')")
    public Vehicle findVehicleByName(@PathVariable String name) { return service.getVehicleByName(name); }

    @PutMapping("/updateVehicle")
    @PreAuthorize("hasRole('STAFF') or hasRole('OWNER')")
    public Vehicle updateVehicle(@RequestBody Vehicle vehicle) { return service.updateVehicle(vehicle); }
```



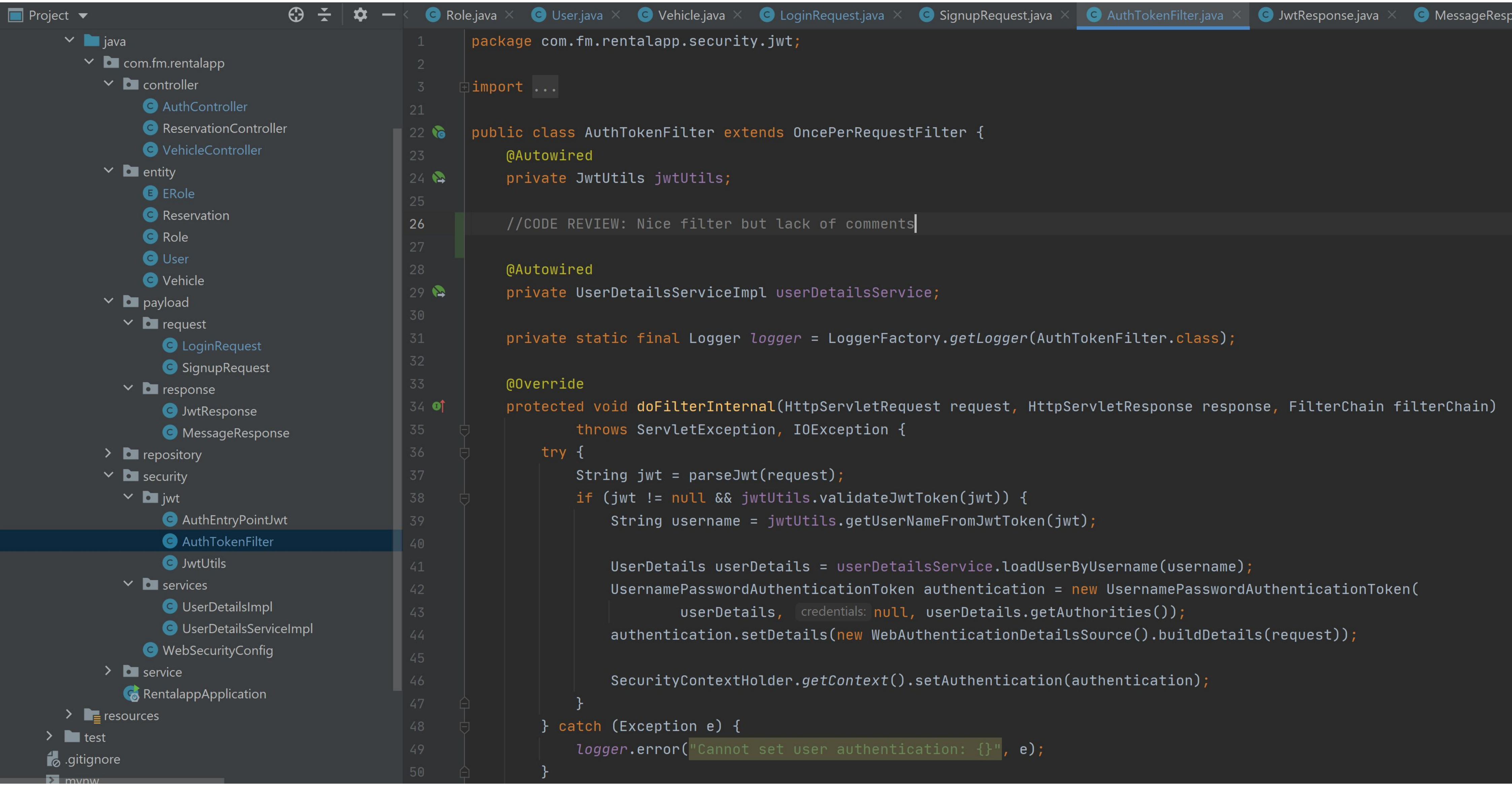
```java
package com.fm.rentalapp.entity;
//CODE REVIEW: Good practice to make the roles enums
public enum ERole {
    ROLE_USER,
    ROLE_STAFF,
    ROLE_OWNER


}
```



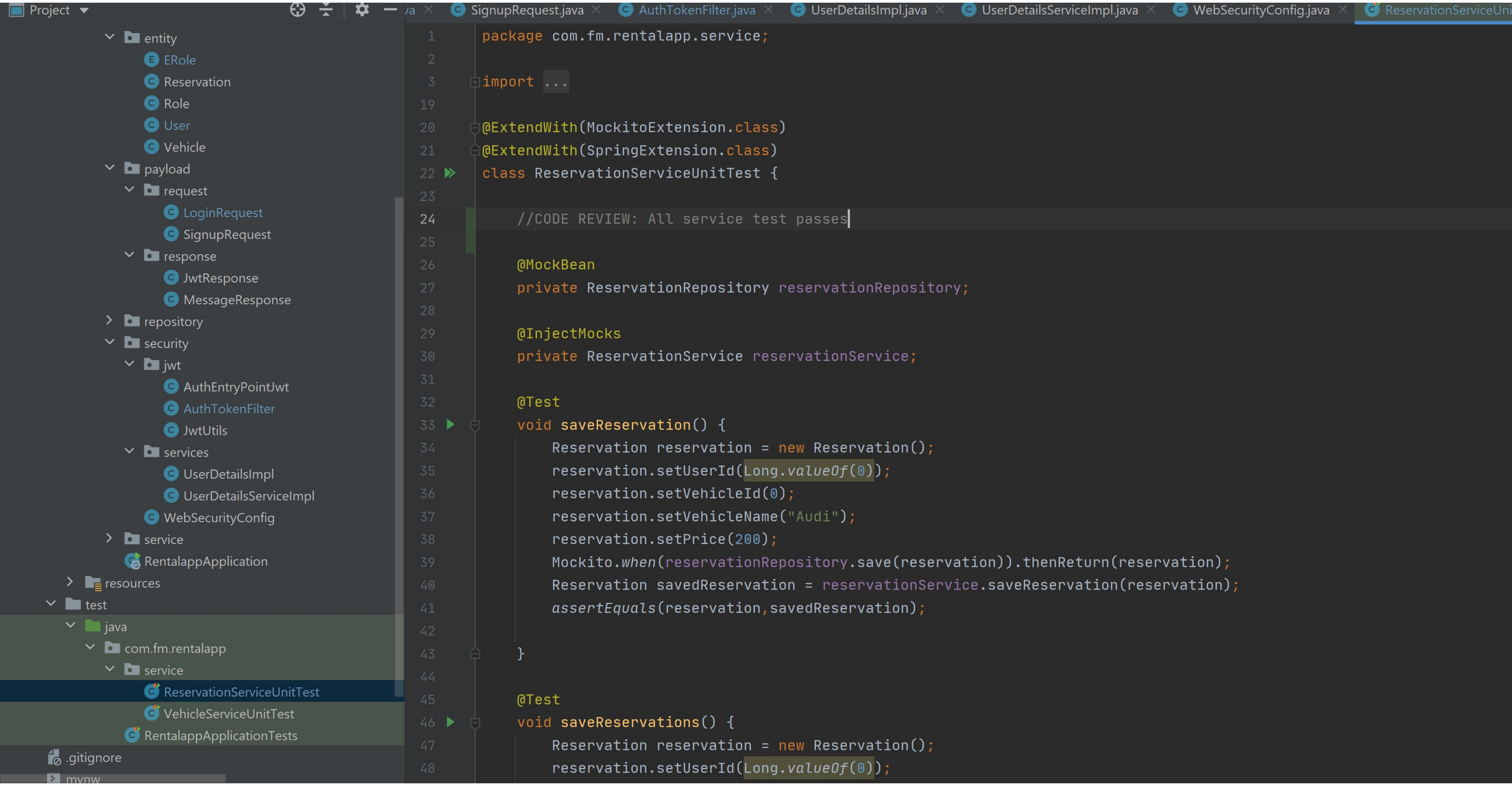
```java
package com.fm.rentalapp.entity;

import ...

//CODE REVIEW: Properties to fields are set good, also the relation between the tables is made good

@Entity
@Table( name = "users",
        uniqueConstraints = {
                @UniqueConstraint(columnNames = "username"),
                @UniqueConstraint(columnNames = "email")
        })
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotBlank
    @Size(max = 20)
    private String username;

    @NotBlank
    @Size(max = 50)
    @Email
    private String email;

    @NotBlank
    @Size(max = 120)
    private String password;

    @ManyToMany(fetch = FetchType.LAZY)
    @JoinTable( name = "user_roles",
            joinColumns = @JoinColumn(name = "user_id"),
```



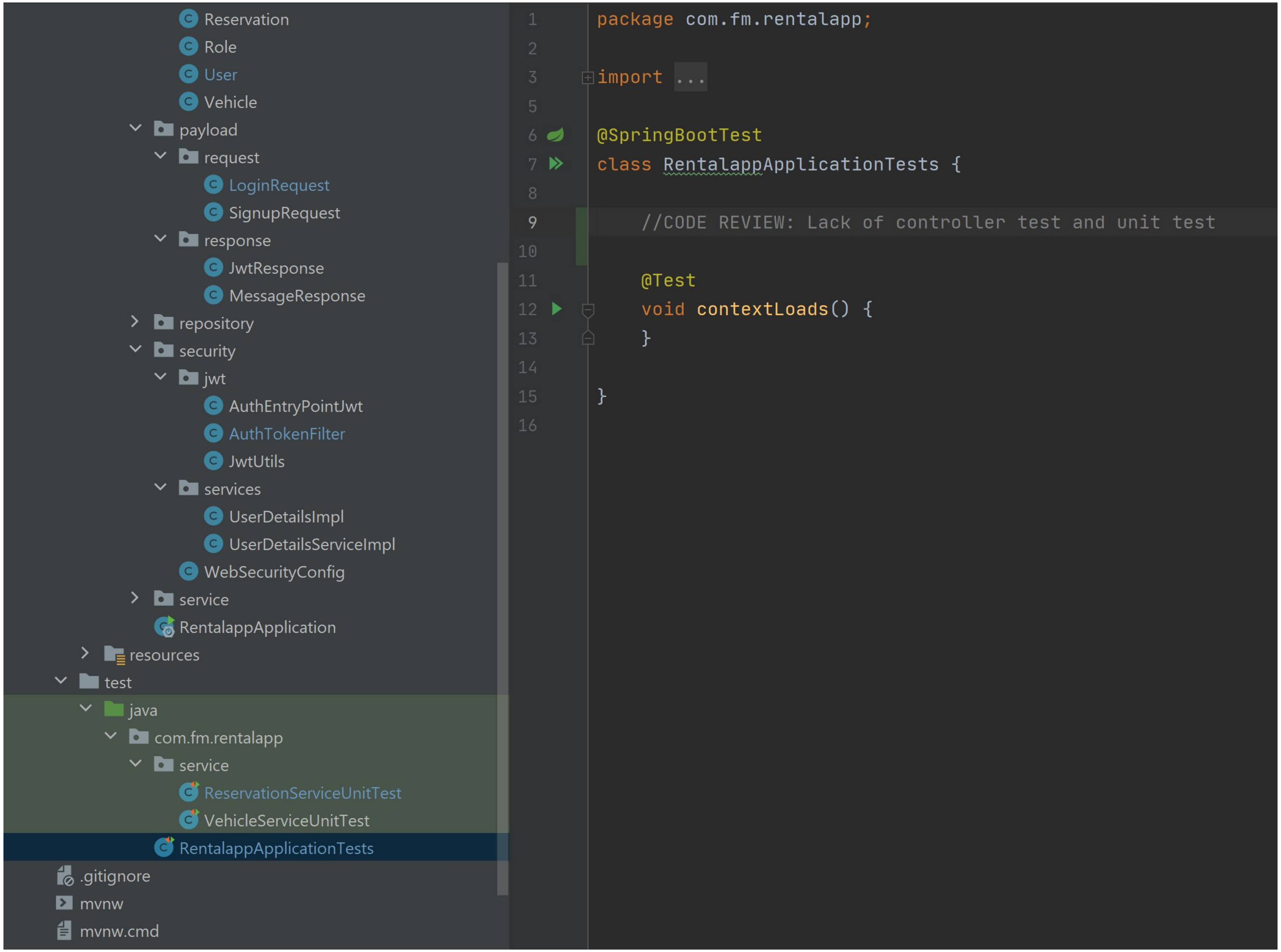
```java
package com.fm.rentalapp.payload.request;

import javax.validation.constraints.NotBlank;

//CODE REVIEW: Good practice to make requests and responses for login and registration

public class LoginRequest {
    @NotBlank
    private String username;

    @NotBlank
    private String password;

    public String getUsername() { return username; }

    public void setUsername(String username) { this.username = username; }

    public String getPassword() { return password; }

    public void setPassword(String password) { this.password = password; }
}
```



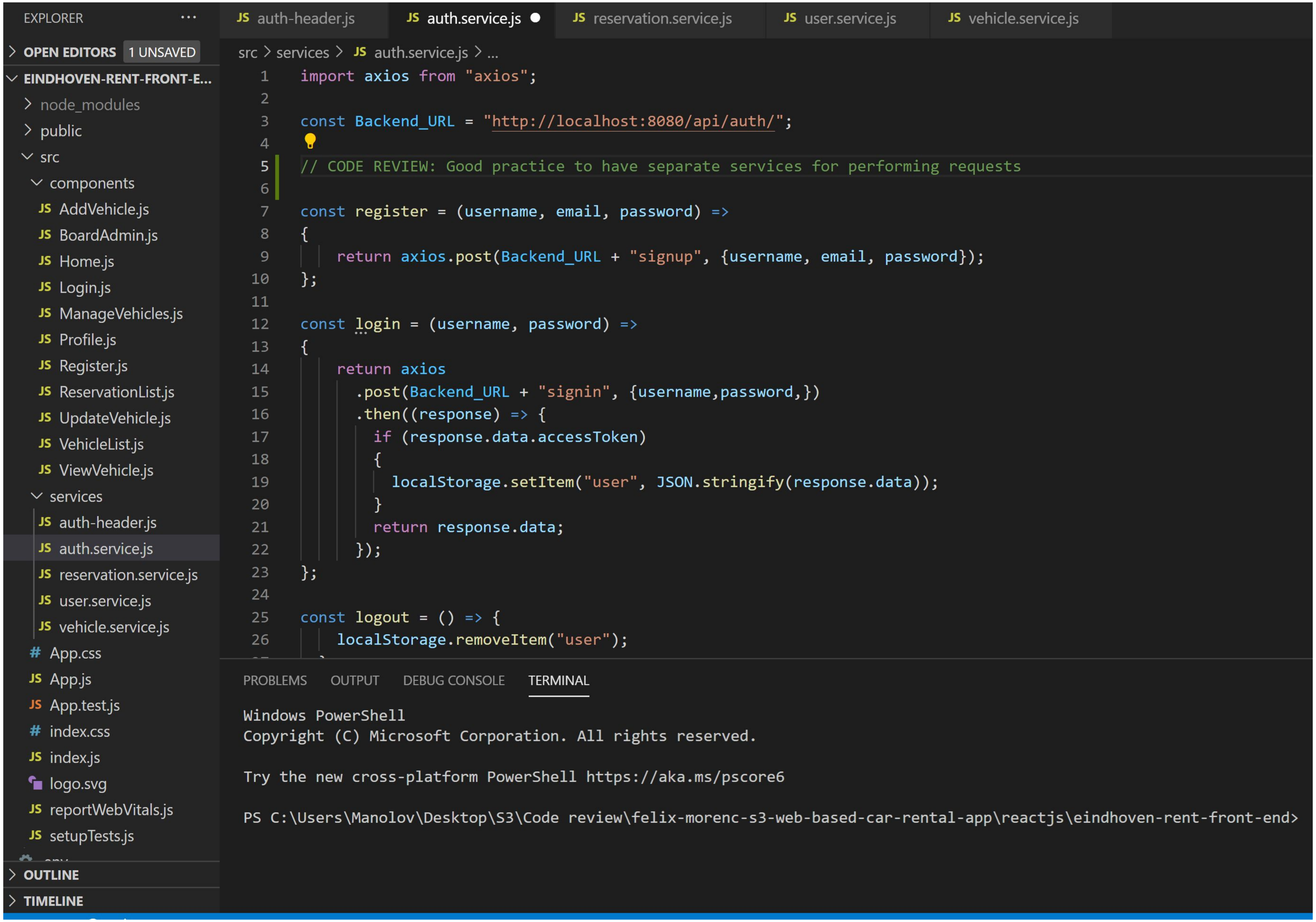
```java
package com.fm.rentalapp.security.jwt;

import ...

public class AuthTokenFilter extends OncePerRequestFilter {
    @Autowired
    private JwtUtils jwtUtils;

    //CODE REVIEW: Nice filter but lack of comments

    @Autowired
    private UserDetailsServiceImpl userDetailsService;

    private static final Logger logger = LoggerFactory.getLogger(AuthTokenFilter.class);

    @Override
    protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain)
            throws ServletException, IOException {
        try {
            String jwt = parseJwt(request);
            if (jwt != null && jwtUtils.validateJwtToken(jwt)) {
                String username = jwtUtils.getUserNameFromJwtToken(jwt);

                UserDetails userDetails = userDetailsService.loadUserByUsername(username);
                UsernamePasswordAuthenticationToken authentication = new UsernamePasswordAuthenticationToken(
                        userDetails, credentials: null, userDetails.getAuthorities());
                authentication.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));

                SecurityContextHolder.getContext().setAuthentication(authentication);
            }
        } catch (Exception e) {
            logger.error("Cannot set user authentication: {}", e);
        }
```



```java
package com.fm.rentalapp.service;

import ...

@ExtendWith(MockitoExtension.class)
@ExtendWith(SpringExtension.class)
class ReservationServiceUnitTest {

    //CODE REVIEW: All service test passes

    @MockBean
    private ReservationRepository reservationRepository;

    @InjectMocks
    private ReservationService reservationService;

    @Test
    void saveReservation() {
        Reservation reservation = new Reservation();
        reservation.setUserId(Long.valueOf(0));
        reservation.setVehicleId(0);
        reservation.setVehicleName("Audi");
        reservation.setPrice(200);
        Mockito.when(reservationRepository.save(reservation)).thenReturn(reservation);
        Reservation savedReservation = reservationService.saveReservation(reservation);
        assertEquals(reservation,savedReservation);

    }

    @Test
    void saveReservations() {
        Reservation reservation = new Reservation();
        reservation.setUserId(Long.valueOf(0));
```

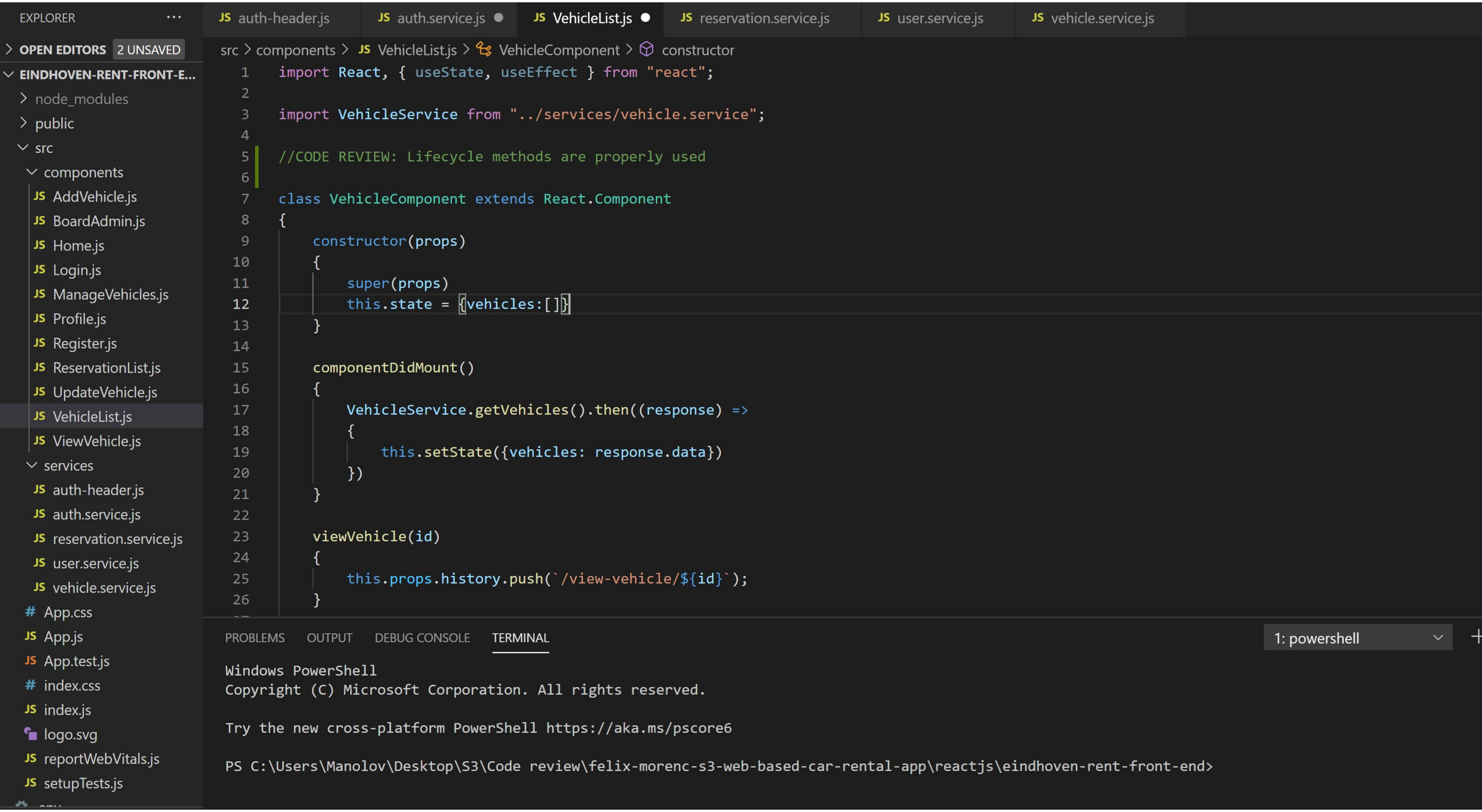


```
package com.fm.rentalapp;

import ...

@SpringBootTest
class RentalappApplicationTests {

    //CODE REVIEW: Lack of controller test and unit test

    @Test
    void contextLoads() {
    }

}
```



```javascript
import axios from "axios";

const Backend_URL = "http://localhost:8080/api/auth/";

// CODE REVIEW: Good practice to have separate services for performing requests

const register = (username, email, password) =>
{
    return axios.post(Backend_URL + "signup", {username, email, password});
};

const login = (username, password) =>
{
    return axios
      .post(Backend_URL + "signin", {username,password,})
      .then((response) => {
        if (response.data.accessToken)
        {
          localStorage.setItem("user", JSON.stringify(response.data));
        }
        return response.data;
      });
};

const logout = () => {
    localStorage.removeItem("user");
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Manolov\Desktop\S3\Code review\felix-morenc-s3-web-based-car-rental-app\reactjs\eindhoven-rent-front-end>
```



```javascript
import React, { useState, useEffect } from "react";

import VehicleService from "../services/vehicle.service";

//CODE REVIEW: Lifecycle methods are properly used

class VehicleComponent extends React.Component
{
    constructor(props)
    {
        super(props)
        this.state = {vehicles:[]}
    }

    componentDidMount()
    {
        VehicleService.getVehicles().then((response) =>
        {
            this.setState({vehicles: response.data})
        })
    }

    viewVehicle(id)
    {
        this.props.history.push(`/view-vehicle/${id}`);
    }
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Manolov\Desktop\S3\Code review\felix-morenc-s3-web-based-car-rental-app\reactjs\eindhoven-rent-front-end>
```