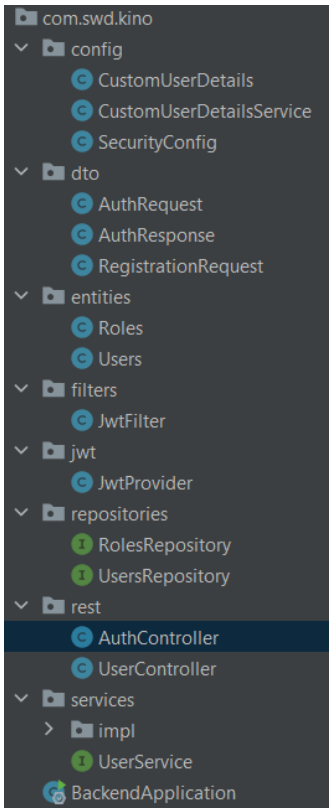
Report of Task 2

Backend



Auth Controller:

```java
@RestController
@CrossOrigin
public class AuthController {
    @Autowired
    private UserService userService;
    @Autowired
    private JwtProvider jwtProvider;

    @PostMapping("/register")
    public String registerUser(@RequestBody RegistrationRequest registrationRequest) {
        Users u = new Users();
        if (userService.findByEmail(registrationRequest.getEmail()) != null) return "SOMETHING WRONG";
        u.setPassword(registrationRequest.getPassword());
        u.setEmail(registrationRequest.getEmail());
        u.setFullName(registrationRequest.getFullName());
        userService.saveUser(u);
        return "OK";
    }

    @PostMapping("/auth")
    public ResponseEntity<?> auth(@RequestBody AuthRequest request) {
        Users user = userService.findByEmailAndPassword(request.getEmail(), request.getPassword());
        if (user != null) {
            String token = jwtProvider.generateToken(user.getEmail());
            return new ResponseEntity<>(new AuthResponse(token, user), HttpStatus.OK);
        } else return new ResponseEntity<>( body: "404", HttpStatus.OK);
    }
}
```

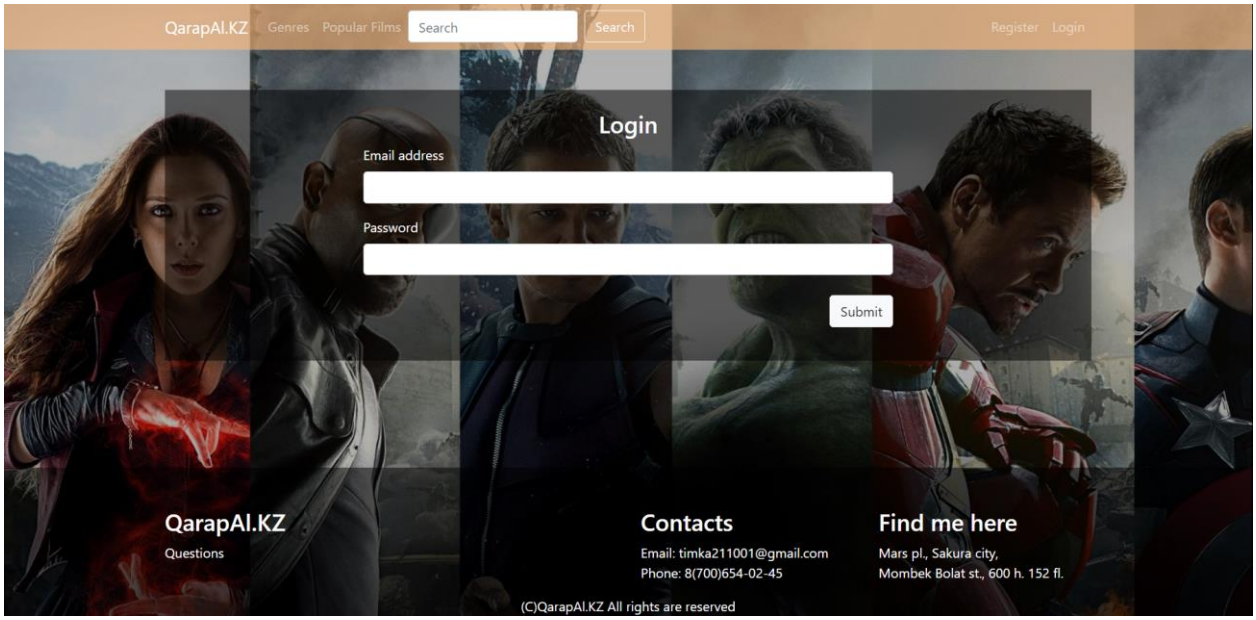
User Service:

```java
@Service
public class UserServiceImpl implements UserService {
    @Autowired
    private UsersRepository usersRepository;
    @Autowired
    private RolesRepository rolesRepository;
    @Autowired
    private PasswordEncoder passwordEncoder;

    @Override
    public Users saveUser(Users user) {
        Roles userRole = rolesRepository.findByRole("ROLE_USER");
        List<Roles> userRoles = new ArrayList<>();
        userRoles.add(userRole);
        user.setRoles(userRoles);
        user.setPassword(passwordEncoder.encode(user.getPassword()));
        return usersRepository.save(user);
    }

    @Override
    public Users findByEmail(String email) { return usersRepository.findByEmail(email); }

    @Override
    public Users findByEmailAndPassword(String email, String password) {
        Users user = findByEmail(email);
        if (user != null) {
            if (passwordEncoder.matches(password, user.getPassword())) {
                return user;
            }
        }
        return null;
    }
}
```

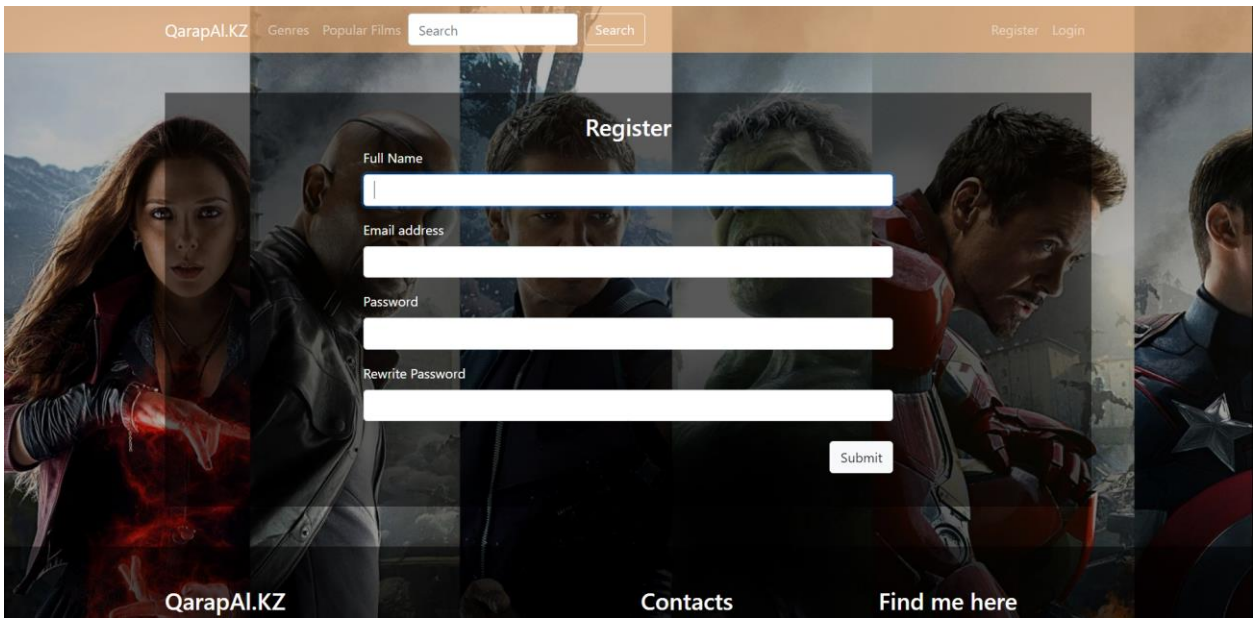
Security Config:

```
@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    @Autowired
    private JwtFilter jwtFilter;

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
                .httpBasic().disable() HttpSecurity
                .csrf().disable()
                .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS)
                .and() HttpSecurity
                .authorizeRequests() ExpressionUrlAuthorizationConfigurer<H>.ExpressionInterceptUrlRegistry
                .antMatchers( ...antPatterns: "/admin/*").hasRole("ADMIN")
                .antMatchers( ...antPatterns: "/register", "/auth").permitAll()
                .and() HttpSecurity
                .addFilterBefore(jwtFilter, UsernamePasswordAuthenticationFilter.class);
    }

    @Bean
    public PasswordEncoder passwordEncoder() { return new BCryptPasswordEncoder(); }
}
```

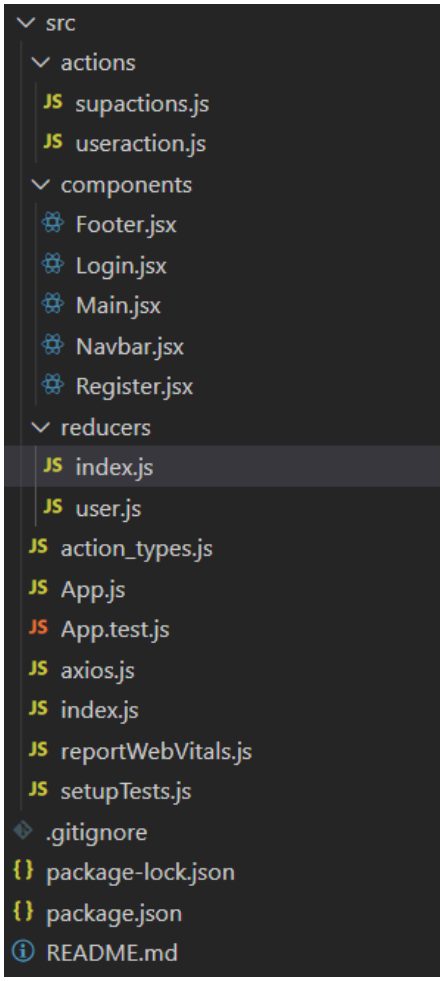
Frontend



Login page (on React)



Register Page (on React)




```
"dependencies": {
    "@testing-library/jest-dom": "^5.11.10",
    "@testing-library/react": "^11.2.6",
    "@testing-library/user-event": "^12.8.3",
    "axios": "^0.21.1",
    "bootstrap": "^4.6.0",
    "bootstrap-icons": "^1.4.1",
    "react": "^17.0.2",
    "react-bootstrap": "^1.5.2",
    "react-bootstrap-icons": "^1.4.0",
    "react-dom": "^17.0.2",
    "react-multilevel-dropdown": "^2.0.0",
    "react-redux": "^7.2.3",
    "react-router-dom": "^5.2.0",
    "react-scripts": "4.0.3",
    "redux": "^4.0.5",
    "web-vitals": "^1.1.1"
  }
```