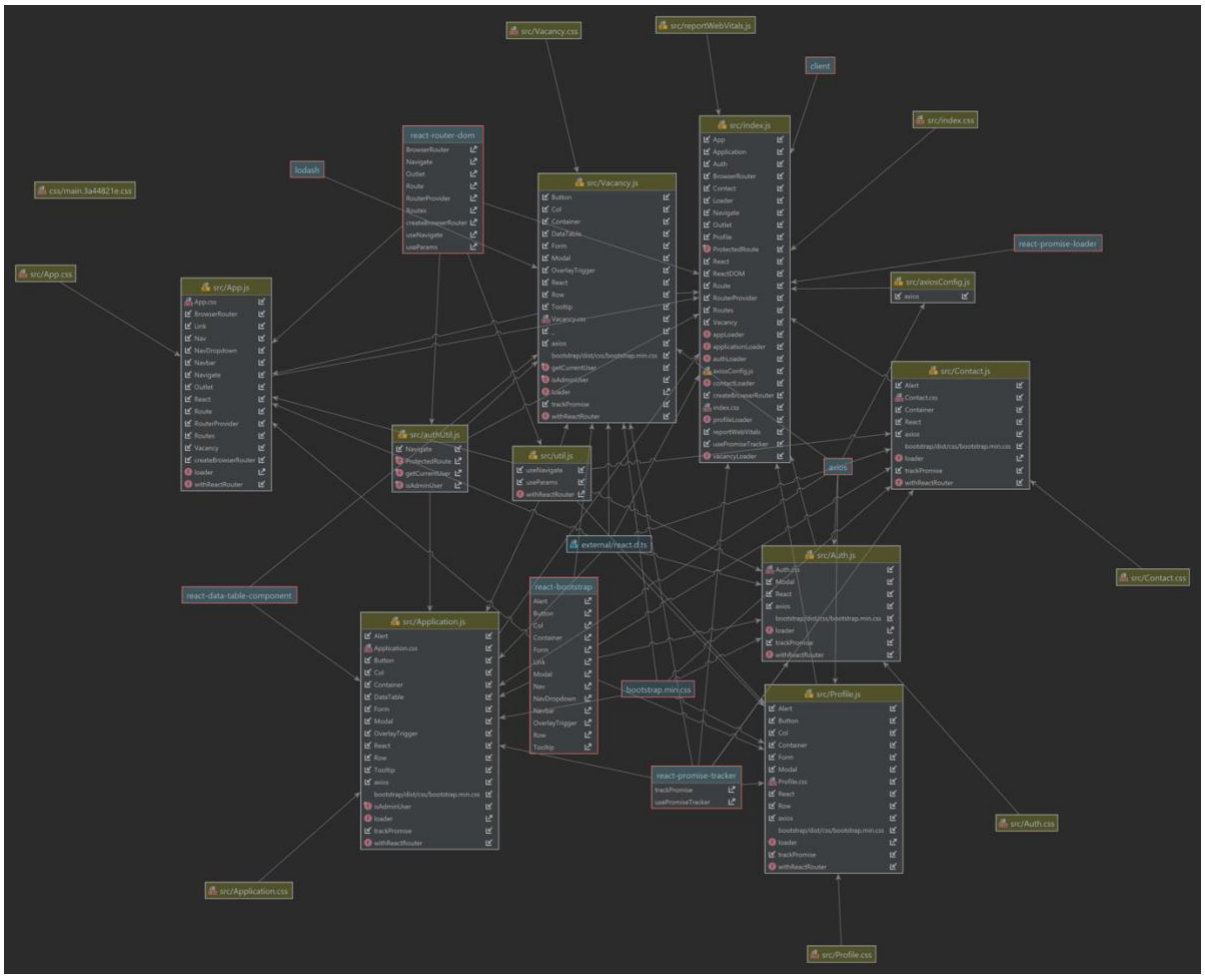
Vacancy Web application
Technical Design
V1.0.0

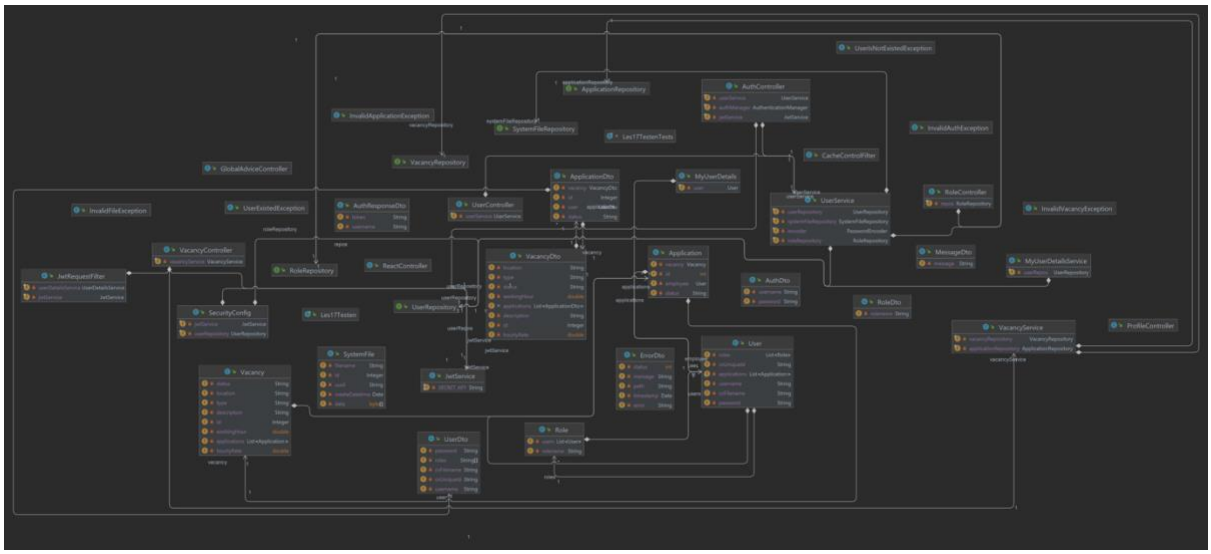
1. Overview

   Vacancy Web is Java spring-boot react web application that allow system owner to post the vacancy and let people applying for the position.

2. Technical Design
   - 2.1. Technical stack
     - 2.1.1. Back-end
       - Java latest stable version (> v11)
       - Maven latest stable version (v3.6)
       - Spring-boot (v2.7.4)
     - 2.1.2. Front-end
       - Nodejs latest stable version (v18)
       - Reactjs (v18.2.0)
       - Bootstrap (v4.6)
     - 2.1.3. Database
       - Postgresql lastest stable version
   - 2.2. System design UML
     - 2.2.1. Front-end UML



2.2.2. Back-end UML



2.3. Api design

2.3.1. Auth api (/api/auth)

- Sign In api:
  - Path: /api/auth/signIn
  - Method: POST
  - Request: AuthDto
  - Response: AuthResponseDto
  - Functionality: logged in user and generate jwt token and return in the response.
  - Error handling: return UNAUTHORIZED if the user credential is invalid.
- Sign Up api:
  - Path: /api/auth/signUp
  - Method: POST
  - Request: AuthDto
  - Response: UserDto
  - Functionality: create user and return detail in the response
  - Error handling:
    - return UNAUTHORIZED if the user credential is invalid.
    - Return BAD_REQUEST if the username is already existed

2.3.2. User api (/api/users)

- Get user info:
  - Path: /api/users/me
  - Method: GET
  - Request: jwt token - header
  - Response: Userdto
  - Functionality: find user detail by the provided jwt token and return in the response
- Upload Cv:
  - Path: /api/users/upload

    - Method: POST
    - Request: jwt token – header, cv file – Multipart file
    - Functionality: find user detail by the provided jwt token, save the uploaded file into database “systemfile” table as blob data type and return the UUID of uploaded file in the response.
    - Error handling:
        - return UNAUTHORIZED if the user credential is invalid.
- Download Cv:
    - Path: /api/users/download
    - Method: GET
    - Request: jwt token – header, UUID of the uploaded file
    - Functionality: find the saved file by UUID and return as byte stream to client

2.3.3. Vacancy api (/api/vacancies)

- Get all vacancy:
    - Path: /api/vacancies
    - Method: GET
    - Request: jwt token - header
    - Response: VacancyDto
    - Functionality: find all vacancies and return in the response
- Get all vacancy by id:
    - Path: /api/vacancies/{id}
    - Method: GET
    - Request: jwt token - header
    - Response: VacancyDto
    - Functionality: find single vacancy by id and return in the response
- Create vacancy:
    - Path: /api/vacancies
    - Method: POST
    - Request: jwt token - header
    - Response: VacancyDto
    - Functionality: create vacancy and return in the response
    - Error handling:
        - return UNAUTHORIZED if the user credential is invalid or it’s not ADMIN user.
- Update vacancy:
    - Path: /api/vacancies
    - Method: PUT
    - Request: jwt token - header
    - Response: VacancyDto
    - Functionality: find vacancy by id and update it and return in the response
    - Error handling:
        - return UNAUTHORIZED if the user credential is invalid or it’s not ADMIN user.
- Apply for a vacancy:
    - Path: /api/vacancies/apply

    - Method: POST
    - Request: jwt token – header, vacancy id
    - Response: status message
    - Functionality:
        - check if the vacancy is still in OPEN status, if yes, allow to proceed with the following, otherwise return 'Already closed' message.
        - check if the user is not yet applied for vacancy, if not allow to apply and save the applied record otherwise return 'Already applied' message.
- Complete decision for the vacancy:
    - Path: /api/vacancies/complete
    - Method: POST
    - Request: jwt token – header, application id, accept/reject status
    - Response: status message
    - Functionality: find application by id, and set status to ACCEPT or REJECT status
    - Error handling:
        - return UNAUTHORIZED if the user credential is invalid or it's not ADMIN user.

2.4. Security design

- System using spring-boot framework together with spring-boot-security to enabling security around endpoint
- Pseudo-code for security configuration is as per below

```
-   @Bean
    public CorsConfigurationSource corsConfigurationSource() {
        final CorsConfiguration configuration = new
    CorsConfiguration();

    configuration.setAllowedOriginPatterns(Arrays.asList("*"));
        configuration.setAllowedMethods(Arrays.asList("HEAD",
                "GET", "POST", "PUT", "DELETE", "PATCH"));
        // setAllowCredentials(true) is important, otherwise:
        // The value of the 'Access-Control-Allow-Origin' header
    in the response must not be the wildcard '*' when the
    request's credentials mode is 'include'.
        configuration.setAllowCredentials(true);
        // setAllowedHeaders is important! Without it, OPTIONS
    preflight request
        // will fail with 403 Invalid CORS request

    configuration.setAllowedHeaders(Arrays.asList("Authorization
    ", "Cache-Control", "Content-Type"));
        final UrlBasedCorsConfigurationSource source = new
    UrlBasedCorsConfigurationSource();
        source.registerCorsConfiguration("/**", configuration);
        return source;
    }

    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http)
    throws Exception {
        http
```

```
                .cors().and()
                .httpBasic().disable()
                .authorizeRequests()
                .antMatchers(HttpMethod.GET,
    "/static/**").permitAll()
                .antMatchers(HttpMethod.GET,
    "/index.html").permitAll()
                .antMatchers(HttpMethod.POST,
    "/api/auth/*").permitAll()
                .antMatchers(HttpMethod.GET,
    "/api/users/cv/download").permitAll()
                .antMatchers("/api/**").hasAnyAuthority("USER",
    "ADMIN")
                .and()
                .addFilterBefore(new
    JwtRequestFilter(jwtService, userDetailsService()),
    UsernamePasswordAuthenticationFilter.class)
                .csrf().disable()

    .sessionManagement().sessionCreationPolicy(SessionCreationPo
    licy.STATELESS);

        return http.build();
    }
```

- Public recourses – under static folder, index.html page is allowed public GET

- Authentication api (/api/auth) is allowed public POST

- CV download api /api/cv/download is allowed public GET

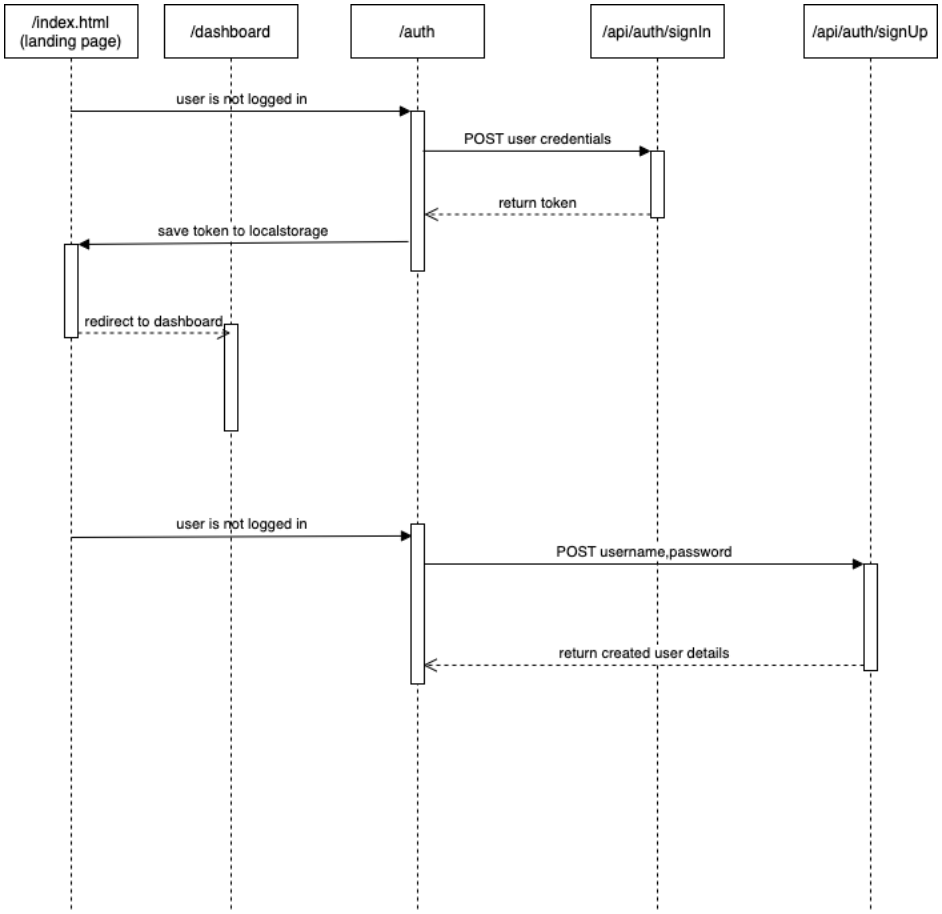
- Other api are secured with jwt token

- Jwt token is secured generated using strong encryption method HS256 by using io.jsonwebtoken jjwt library. Pseudo code for token generation is below

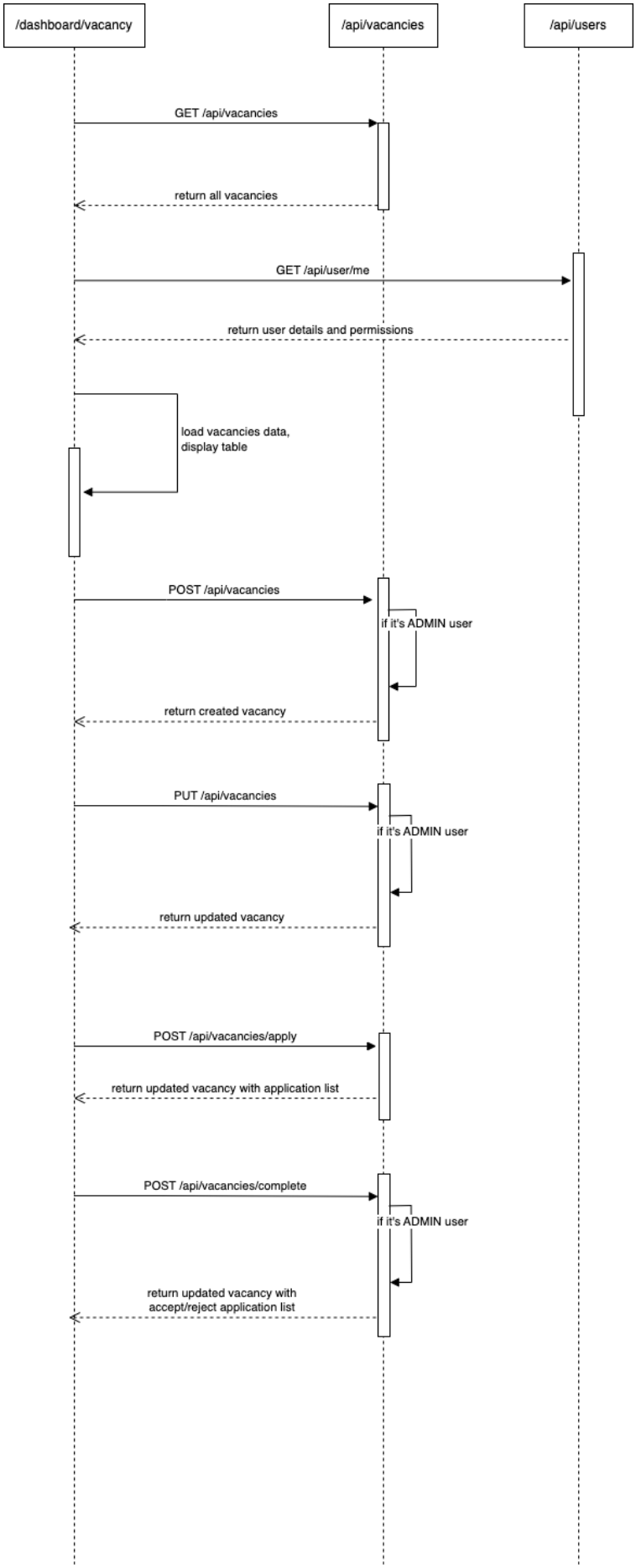
```
private String createToken(Map<String, Object> claims, String
        subject) {
    long validPeriod = 1000 * 60 * 60 * 24 * 10; // 10 days in ms
    long currentTime = System.currentTimeMillis();
    return Jwts.builder()
            .setClaims(claims)
            .setSubject(subject)
            .setIssuedAt(new Date(currentTime))
            .setExpiration(new Date(currentTime + validPeriod))
            .signWith(SignatureAlgorithm.HS256, SECRET_KEY)
            .compact();
}
```

2.5. Application sequence diagram

2.5.1. User sign up/ sign in flow



2.5.2. Vacancy flow


/dashboard/vacancy
/api/vacancies
/api/users
GET /api/vacancies
return all vacancies
GET /api/user/me
return user details and permissions
load vacancies data,
display table
POST /api/vacancies
if it's ADMIN user
return created vacancy
PUT /api/vacancies
if it's ADMIN user
return updated vacancy
POST /api/vacancies/apply
return updated vacancy with application list
POST /api/vacancies/complete
if it's ADMIN user
return updated vacancy with
accept/reject application list

2.5.3. User profile flow