

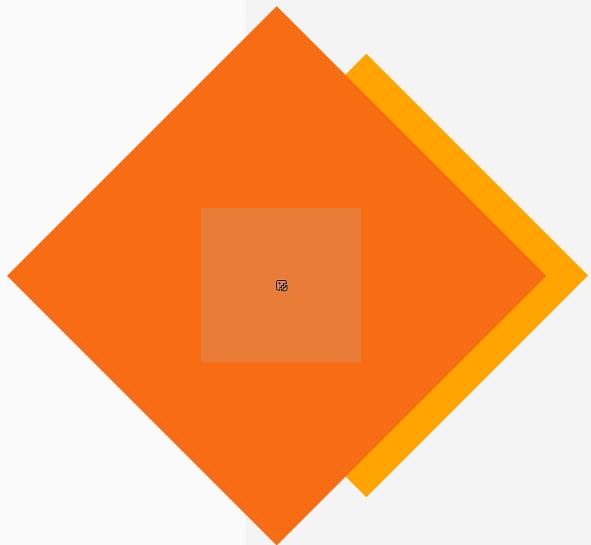
# Firebase Authentication and Integration with Spring Boot

| Session 6



# **Session Overview:**

## Training Approach and Interaction



### **Session 6**

Presentation of Training Materials

Practical Training

Discussion

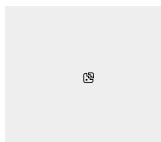
# Session 6 Objective

Participants understand how to integrate Firebase Authentication with Spring Boot and Spring Security for secure user management, REST API protection, and role-based authorization.

# MATERI PELATIHAN

---

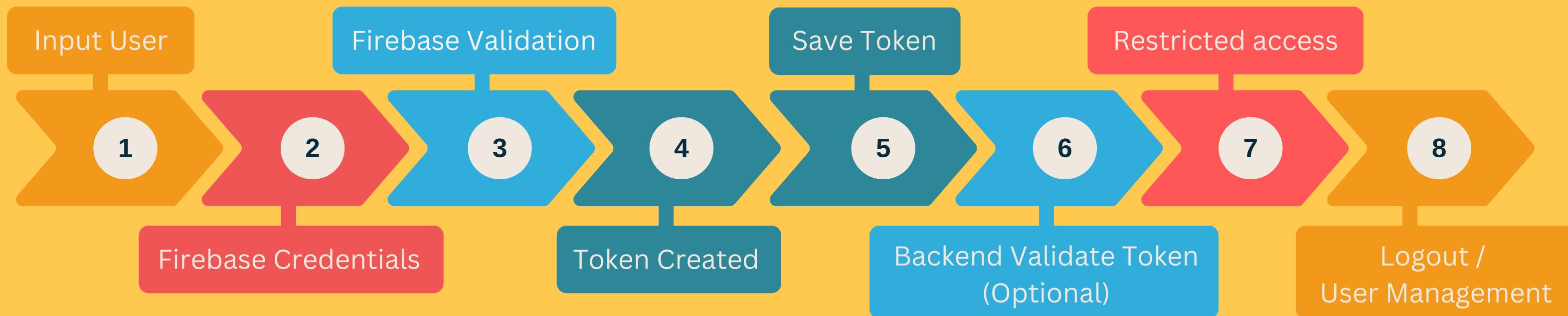
- **Understanding Firebase Authentication concepts and user management.**
- **Integrating Firebase Authentication with Spring Boot login and signup workflows.**
- **Securing REST APIs with Firebase Authentications tokens and user claims**
- **Implementing user information management and authorization based on roles.**
- **Integrating Firebase Authentication with Spring Security.**



# Firebase authentication concepts

---

- Firebase Authentication dirancang untuk menyederhanakan pembuatan sistem autentikasi yang aman sekaligus meningkatkan pengalaman login dan onboarding bagi pengguna.



# User management

- CRUD Operations tersedia dalam Firebase authentication.

```
17
18 @Service
19 public class FirebaseAuthService {
20
21     @Value("${firebase.api-key}")
22     private String firebaseApiKey;
23
24     private final RestTemplate restTemplate = new RestTemplate();
25
26     public LoginResponse signup(LoginRequest signupRequest) {
27         String url = "https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=" + firebaseApiKey;
28
29         Map<String, Object> request = new HashMap<>();
30         request.put("email", signupRequest.getEmail());
31         request.put("password", signupRequest.getPassword());
32         request.put("returnSecureToken", true);
33
34         Map<String, Object> response = restTemplate.postForObject(url, request, Map.class);
35         String idToken = (String) response.get("idToken");
36
37         return new LoginResponse(idToken);
38     }
39
40     public LoginResponse login(LoginRequest loginRequest) {
41         String url = "https://identitytoolkit.googleapis.com/v1/accountssignInWithPassword?key=" + firebaseApiKey;
42
43         Map<String, Object> request = new HashMap<>();
44         request.put("email", loginRequest.getEmail());
45         request.put("password", loginRequest.getPassword());
46         request.put("returnSecureToken", true);
47
48         Map<String, Object> response = restTemplate.postForObject(url, request, Map.class);
49         String idToken = (String) response.get("idToken");
50
51         return new LoginResponse(idToken);
52     }
53
54     public UserRecord getUserByEmail(String email) throws FirebaseAuthException {
55         return FirebaseAuth.getInstance().getUserByEmail(email);
56     }
57
58     public UserRecord updateUser(String uid, String email, String password)
59         throws FirebaseAuthException {
60         UpdateRequest request = new UpdateRequest(uid)
61             .setEmail(email)
62             .setPassword(password);
63
64         return FirebaseAuth.getInstance().updateUser(request);
65     }
66
67     public void deleteUser(String uid) throws FirebaseAuthException {
68         FirebaseAuth.getInstance().deleteUser(uid);
69     }
70
71     public FirebaseToken verifyIdToken(String idToken) throws FirebaseAuthException {
72         return FirebaseAuth.getInstance().verifyIdToken(idToken);
73     }
74 }
```

# Setup Firebase authentication

---

- Membuat firebase project:  
<https://console.firebaseio.google.com/>
- Menambahkan dependency firebase pada project Spring Boot

```
1 <dependency>
2   <groupId>com.google.firebaseio</groupId>
3   <artifactId>firebase-admin</artifactId>
4   <version>8.1.0</version>
5 </dependency>
```

# Setup Firebase authentication

---

- Download secret key dari firebase project

The screenshot shows the Firebase Admin SDK service account configuration page. On the left, there's a sidebar with options: Legacy credentials, Database secrets, All service accounts, and a highlighted section for 4 service accounts. The main content area is titled "Firebase Admin SDK" and describes how a service account can authenticate multiple Firebase features like Database, Storage, and Auth via the Admin SDK. It shows a service account email: "firebase-adminsdk-mbqxc@restapi-8fac8.iam.gserviceaccount.com". Below this is an "Admin SDK configuration snippet" for Java, which includes code for initializing the Firebase Admin SDK using a service account key file:

```
FileInputStream serviceAccount =  
    new FileInputStream("path/to/serviceAccountKey.json");  
  
FirebaseOptions options = new FirebaseOptions.Builder()  
    .setCredentials(GoogleCredentials.fromStream(serviceAccount))  
    .build();  
  
FirebaseApp.initializeApp(options);
```

At the bottom, there's a blue button labeled "Generate new private key".

# Config

## Penambahan Firebase Config

```
1 package com.example.demo.security;
2
3 import com.google.auth.oauth2.GoogleCredentials;
4 import com.google.firebaseio.FirebaseApp;
5 import com.google.firebaseio.FirebaseOptions;
6 import org.springframework.context.annotation.Bean;
7 import org.springframework.context.annotation.Configuration;
8
9 import java.io.File;
10 import java.io.FileInputStream;
11 import java.io.IOException;
12
13 @Configuration
14 public class FirebaseConfig {
15
16     @Bean
17     public FirebaseApp initializeFirebase() throws IOException {
18         File file = new File("src/main/resources.firebaseiokey.json");
19         FileInputStream serviceAccount = new FileInputStream(file);
20
21         FirebaseOptions options = new FirebaseOptions.Builder()
22             .setCredentials(GoogleCredentials.fromStream(serviceAccount))
23             .build();
24
25         return FirebaseApp.initializeApp(options);
26     }
27 }
```

Firebase Config digunakan untuk inisialisasi firebase connection pada project Spring Boot sesuai dengan credential akun project firebase.

# Firebase authentication with Spring Security

---

- Menambahkan filter chain pada security config sesuai dengan firebase token filter

```
1 package com.example.demofb.security;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.context.annotation.Bean;
5 import org.springframework.context.annotation.Configuration;
6 import org.springframework.security.authentication.AuthenticationManager;
7 import org.springframework.security.config.annotation.authentication.builders.AuthenticationBuilder;
8 import org.springframework.security.config.annotation.web.builders.HttpSecurity;
9 import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
10 import org.springframework.security.config.http.SessionCreationPolicy;
11 import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
12 import org.springframework.security.crypto.password.PasswordEncoder;
13 import org.springframework.security.web.SecurityFilterChain;
14 import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;
15
16 @Configuration
17 @EnableWebSecurity
18 public class SecurityConfig {
19
20     @Autowired
21     private CustomUserDetailsService customUserDetailsService;
22
23     @Autowired
24     private FirebaseTokenFilter firebaseTokenFilter;
25
26     @Bean
27     public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
28         http
29             .authorizeRequests(authorizeRequests -> authorizeRequests
30                 .requestMatchers("/api/auth/**").permitAll()
31                 .requestMatchers("/api/users/**").permitAll()
32                 .requestMatchers("/api/task/**").authenticated()
33                 .anyRequest().permitAll())
34             .csrf(csrf -> csrf.disable())
35             .sessionManagement(
36                 sessionManagement -> sessionManagement.sessionCreationPolicy(SessionCreationPolicy.STATELESS));
37
38         http.addFilterBefore(firebaseTokenFilter, UsernamePasswordAuthenticationFilter.class);
39         return http.build();
40     }
41
42     @Bean
43     public PasswordEncoder passwordEncoder() {
44         return new BCryptPasswordEncoder();
45     }
46 }
```

# Service

## Penambahan Firebase Auth Service

```
17  @Service
18  public class FirebaseAuthService {
19      @Value("${firebase.api-key}")
20      private String firebaseApiKey;
21
22      private final RestTemplate restTemplate = new RestTemplate();
23
24      public LoginResponse signup(LoginRequest signupRequest) {
25          String url = "https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=" + firebaseApiKey;
26
27          Map<String, Object> request = new HashMap<>();
28          request.put("email", signupRequest.getEmail());
29          request.put("password", signupRequest.getPassword());
30          request.put("returnSecureToken", true);
31
32          Map<String, Object> response = restTemplate.postForObject(url, request, Map.class);
33          String idToken = (String) response.get("idToken");
34
35          return new LoginResponse(idToken);
36      }
37
38      public LoginResponse login(LoginRequest loginRequest) {
39          String url = "https://identitytoolkit.googleapis.com/v1/accountssignInWithPassword?key=" + firebaseApiKey;
40
41          Map<String, Object> request = new HashMap<>();
42          request.put("email", loginRequest.getEmail());
43          request.put("password", loginRequest.getPassword());
44          request.put("returnSecureToken", true);
45
46          Map<String, Object> response = restTemplate.postForObject(url, request, Map.class);
47          String idToken = (String) response.get("idToken");
48
49          return new LoginResponse(idToken);
50      }
51
52      public UserRecord getUserByEmail(String email) throws FirebaseAuthException {
53          return FirebaseAuth.getInstance().getUserByEmail(email);
54      }
55
56      public UserRecord updateUser(String uid, String email, String password)
57          throws FirebaseAuthException {
58          UpdateRequest request = new UpdateRequest(uid)
59              .setEmail(email)
60              .setPassword(password);
61
62          return FirebaseAuth.getInstance().updateUser(request);
63      }
64
65      public void deleteUser(String uid) throws FirebaseAuthException {
66          FirebaseAuth.getInstance().deleteUser(uid);
67      }
68
69      public FirebaseToken verifyIdToken(String idToken) throws FirebaseAuthException {
70          return FirebaseAuth.getInstance().verifyIdToken(idToken);
71      }
72
73  }
```

Firebase Atuh Service digunakan sebagai class service untuk proses operasi sesuai dengan Data Request yang dikirimkan.

# Controller

## Penambahan Auth Firebase Controller

```
1 package com.example.demo.controller;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.security.authentication.BadCredentialsException;
5 import org.springframework.security.core.context.SecurityContextHolder;
6 import org.springframework.web.bind.annotation.*;
7
8 import com.example.demo.model.LoginRequest;
9 import com.example.demo.model.LoginResponse;
10 import com.example.demo.service.FirebaseAuthService;
11
12 @RestController
13 @RequestMapping("/api/auth/firebase")
14 public class AuthFirebaseController {
15
16     @Autowired
17     private FirebaseAuthService firebaseAuthService;
18
19     @PostMapping("/signup")
20     public LoginResponse signup(@RequestBody LoginRequest signupRequest) {
21         return firebaseAuthService.signup(signupRequest);
22     }
23
24     @PostMapping("/login")
25     public LoginResponse authenticateUser(@RequestBody LoginRequest loginRequest) {
26         try {
27             return firebaseAuthService.login(loginRequest);
28         } catch (BadCredentialsException e) {
29             throw new RuntimeException("Invalid email or password");
30         }
31     }
32
33     @PostMapping("/logout")
34     public String logoutUser() {
35         SecurityContextHolder.clearContext();
36         return "User logged out successfully";
37     }
38 }
39 }
```

Auth Firebase Controller digunakan sebagai controller untuk menangani semua request CRUD operation yang terhubung ke firebase authentication service.

# Config

## Penambahan Firebase Token Filter

```
1 package com.example.demo.security;
2
3 import com.google.firebase.auth.FirebaseAuth;
4 import com.google.firebase.auth.FirebaseAuthException;
5 import com.google.firebase.auth.FirebaseToken;
6
7 import jakarta.servlet.FilterChain;
8 import jakarta.servlet.ServletException;
9 import jakarta.servlet.http.HttpServletRequest;
10 import jakarta.servlet.http.HttpServletResponse;
11
12 import org.springframework.security.core.context.SecurityContextHolder;
13 import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;
14 import org.springframework.stereotype.Component;
15 import org.springframework.web.filter.OncePerRequestFilter;
16
17 import java.io.IOException;
18
19 @Component
20 public class FirebaseTokenFilter extends OncePerRequestFilter {
21
22     @Override
23     protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain)
24             throws ServletException, IOException {
25         String header = request.getHeader("Authorization");
26         if (header != null && header.startsWith("Bearer ")) {
27             String idToken = header.substring(7);
28             try {
29                 FirebaseToken decodedToken = FirebaseAuth.getInstance().verifyIdToken(idToken);
30                 FirebaseAuthToken authentication = new FirebaseAuthenticationToken(decodedToken);
31                 authentication.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));
32                 SecurityContextHolder.getContext().setAuthentication(authentication);
33             } catch (FirebaseAuthException e) {
34                 e.printStackTrace();
35             }
36         }
37         filterChain.doFilter(request, response);
38     }
39 }
```

Firebase Token Filter digunakan untuk memverifikasi dan memproses token autentikasi yang diterima dari klien. Fungsinya adalah memastikan bahwa hanya pengguna yang telah diautentikasi dan memiliki token yang valid dapat mengakses resource atau endpoint tertentu.

# Config

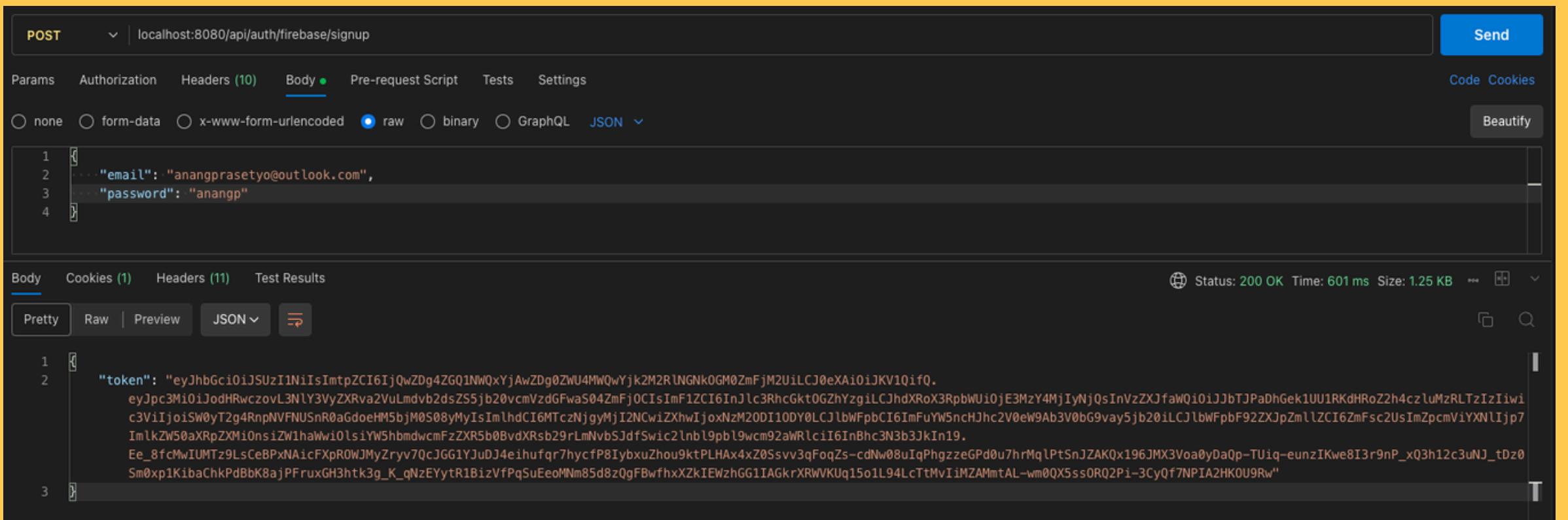
## Penambahan Firebase Authentication Token

```
1 package com.example.demo.security;
2
3 import com.google.firebase.auth.FirebaseToken;
4 import org.springframework.security.authentication.AbstractAuthenticationToken;
5
6 public class FirebaseAuthenticationToken extends AbstractAuthenticationToken {
7
8     private final FirebaseToken firebaseToken;
9
10    public FirebaseAuthenticationToken(FirebaseToken firebaseToken) {
11        super(null);
12        this.firebaseioToken = firebaseToken;
13        setAuthenticated(true);
14    }
15
16    @Override
17    public Object getCredentials() {
18        return null;
19    }
20
21    @Override
22    public Object getPrincipal() {
23        return this.firebaseioToken;
24    }
25
26    public FirebaseToken getFirebaseToken() {
27        return firebaseToken;
28    }
29 }
```

Firebase Authentication Token adalah class untuk mendapatkan token keamanan yang dihasilkan oleh Firebase Authentication setelah pengguna berhasil diautentikasi.

# Testing

## Endpoint Firebase Create User



The screenshot shows a Postman interface for a POST request to `localhost:8080/api/auth/firebase/signup`. The request body is set to `raw JSON` and contains the following data:

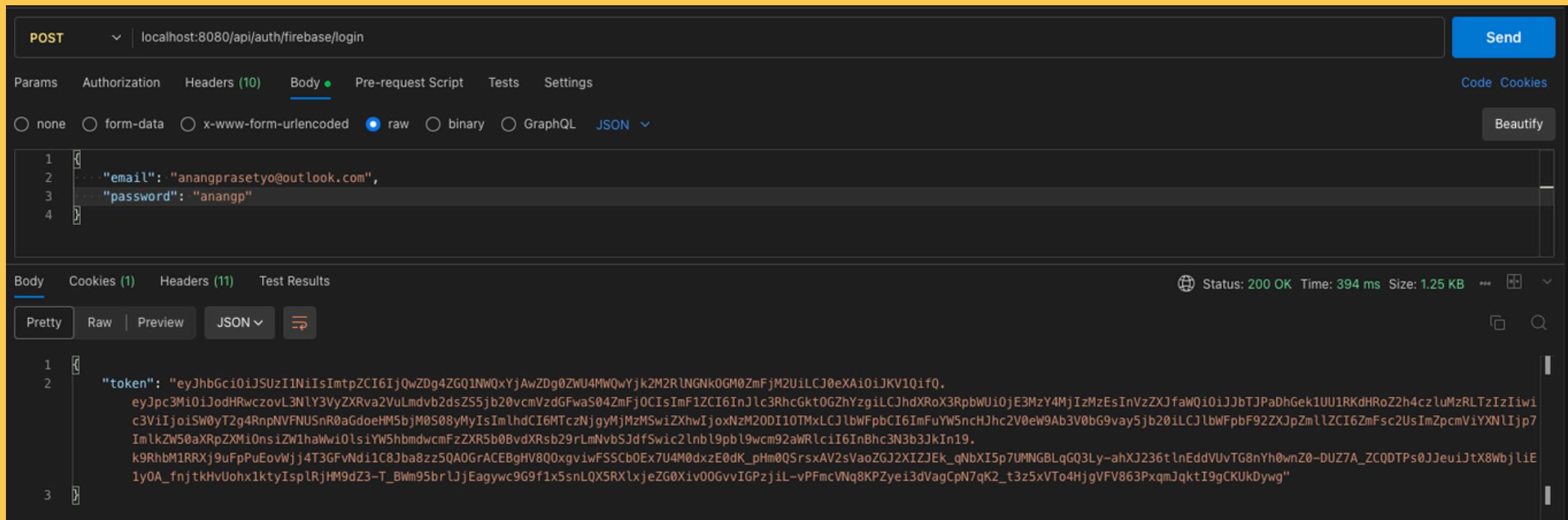
```
1 [{}]
2   ...
3     "email": "anangprasetyo@outlook.com",
4     "password": "anangp"
5 ]
```

The response status is `200 OK` with a response time of `601 ms` and a size of `1.25 KB`. The response body is a JSON object containing a token:

```
1 {
2   "token": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjQwZDg4ZGQ1NW0xYjAwZDg0ZWU4MWQwYjk2M2RlNGNkOGM0ZmFjM2UiLCJ0eXAiOiJKV1QiQfQ.
3     eyJpc3MiOiJodHRwczovL3NLY3VyZXRva2VuLmdvb2dsZS5jb20vcmVzdGFwaS04ZmFjOCIsImF1ZCI6InJlc3Rhcgkt0GZhYzgilCJhdXRox3RpbwUi0jE3MzY4MjIyNjQsInVzZXJfaWQiOijJbTJPaDhGek1UU1RKdHRoZ2h4czluMzRLTzIzIwi
4       c3ViIjo5W0yT2g4RnpNVNUSnR0aGdoeHM5bjM0S08yMyIsImlhCI6MTCzNjgyMjI2NCwiZKhwiJioxNzM2ODI1ODY0LCJlbWFpbCI6ImFuYW5nchJhc2V0eW9Ab3V0bG9vay5jb20iLCJlbWFpbF92ZXJpZmllZCI6ZmFsc2UsImZpcmViYXNlIjp7
5         ImlkZW50aXRpZXMiOnsiZW1haWwiOlsiYWh5bmdwcmFzZXr5b0BvdXRsb29rLmNvbSJdfSwic2lnbl9pbL9wcm92aWRlcii6InBhc3N3b3JkIn19.
6           Ee_8fcMwIUMtZ9LsCeBPxNAicFXpROWJMyZryv7QcJGG1YJuDJ4eiuhfqr7hycfP8IybxuZhou9ktPLHax4xZ0Ssvv3qFoqZs-cdNw08uIqPhgzzeGPd0u7hrMqlPtSnJZAKQx196JMX3Voa0yDaQp-TUIq-eunzIKwe8I3r9nP_xQ3h12c3uNJ_tDz0
7             Sm0xp1KibaChkPd8bK8ajPFrxGH3htk3g_K_qNzEYytR1BizVfPqSuEeoMNm85d8zQgFBwfhxXZkIEWzhGG1IAGkrXRWVKUq15o1L94LcTtMvIiMZAMmtAL-wm0QX5ss0RQ2Pi-3CyQf7NP1A2HK0U9Rw"
```

# Testing

## Endpoint Firebase User Login



The screenshot shows a Postman request for a POST endpoint at `localhost:8080/api/auth/firebase/login`. The request body is set to "raw" JSON:

```
1 [{}  
2 ... "email": "anangprasetyo@outlook.com",  
3 ... "password": "anangp"  
4 ]
```

The response status is 200 OK, with a response body containing a JSON token:

```
1 {  
2   "token": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjQwZDg4ZGQ1NWQxYjAwZDg0ZWU4MwQwYjk2M2R1NGNkOGM0ZmFjM2UiLCJ0eXAiOiJKV1QiAQ.  
eyJpc3Mi0iJodHRwczovL3NLY3VyZXRxva2VuLmdvb2dsZS5jb20vcmVzdGFwaS04Zmfj0CisImF1ZCI6InJlc3Rhcgkt0GZhYzgilCJhdXRox3RpbwUi0jE3MjIzMzEsInVzxJfaWQioiJJbTJPaDhGek1UU1RKdHRoZ2h4czluMzRLTzIzIiwi  
c3ViIjoiSW0yT2g4RnpNVNUsnR0aGdoehM5bjM0S08yMyisImlhdi6MtczNjgyMjMzMSwiZXhwIjoxNzM20D110TMxLCJlbWFpbCI6ImFuYW5ncHJhc2V0eW9Ab3V0bG9vay5jb20iLCJlbWFpbF92ZXJpZmlZCI6ZmFsc2UsImZpcmViYXNlIjp7  
ImlkZW50aXRpZXM1OnsiZW1haWwiOlsiyW5hbmdwcmF2ZXr5b0BvdXRsB29rLmNvbS3dfSwic2lnb19pb19wcm92aWRlcii6InBhc3N3b3JkIn19.  
k9Rhbm1RRXj9uFpPuEovWjj4T3GFvNdi1C8Jba8zz5QA0GrACEBgHV8Q0xgvivFSSCb0Ex7U4M0dxzE0dK_pHm0Q5rsxA2sVaoZGJ2XIZJEK_qNbXI5p7UMNGBLqGQ3Ly-ahXJ236tlnEddUVvTG8nYh0wnZ0-DUZ7A_ZCQDTPs0JJjeuiJtX8WbjliE  
1y0A_fnjtkHvUohx1ktyIsplRjHM9dZ3-T_Bwm95brljjeagywc9G9f1x5snLQX5Rx1xeZG0Xiv00GvvIGPzjiL-vPFmcVNq8KPZyei3dVagCpN7qK2_t3z5xVTo4HjgVFV863PxqmJqktI9gCKUkDywg"  
3 }
```

# Testing

## Endpoint Task User

The screenshot shows the Postman application interface for testing a REST API endpoint. The URL is set to `localhost:8080/api/task/user/1`. The `Headers` tab is selected, displaying the following configuration:

Key	Value
Authorization	Bearer eyJhbGciOiJSUzI1NiIsImtpZCI6IjQwZDg4ZGQ1NWQxYjAwZDg0ZWU4MWQwYjk2M2RINGNkOGM0ZmFjM2UiL...
Key	Value

The `Body` tab displays the JSON response received from the server. The response is a single array containing one task object:

```
1 [ { 2   "id": 1, 3     "title": "Complete the project", 4     "description": "Finish the project by the end of the week", 5     "priority": "HIGH", 6     "status": true, 7     "user": { 8       "id": 1, 9       "name": "Anang", 10      "email": "anangprasetyo@outlook.com" 11     } 12   } 13 ] 14 }
```

The status bar at the bottom indicates a successful response: `Status: 200 OK Time: 80 ms Size: 532 B`.

# REFERENCES

- GeeksforGeeks. (2024, May 10). What is Firebase Authentication. GeeksforGeeks.  
<https://www.geeksforgeeks.org/what-is-firebase-authentication/>
  
- GeeksforGeeks. (2024, May 10). What is Firebase Authentication. GeeksforGeeks.  
<https://www.geeksforgeeks.org/what-is-firebase-authentication/>



# THANK YOU

Thank you for following session 6 to the end.  
See you at the next meeting