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

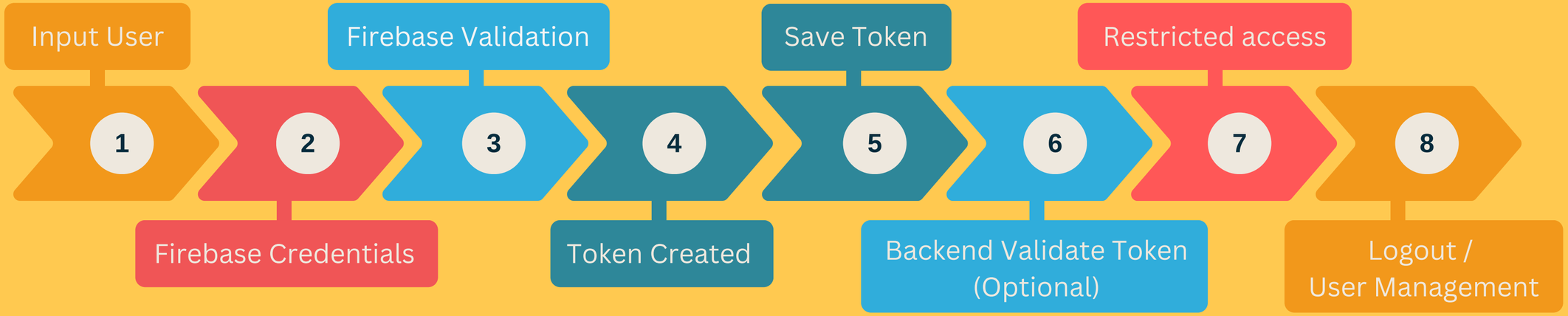
Participants understand how to integrate Firebase Authentication with Spring Boot and Spring Security for secure user management, REST API protection, and role-based authorization.

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.

```java
@Service
public class FirebaseAuthService {

    @Value("${firebase.api-key}")
    private String firebaseApiKey;

    private final RestTemplate restTemplate = new RestTemplate();

    public LoginResponse signup(LoginRequest signupRequest) {
        String url = "https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=" + firebaseApiKey;

        Map<String, Object> request = new HashMap<>();
        request.put("email", signupRequest.getEmail());
        request.put("password", signupRequest.getPassword());
        request.put("returnSecureToken", true);

        Map<String, Object> response = restTemplate.postForObject(url, request, Map.class);
        String idToken = (String) response.get("idToken");

        return new LoginResponse(idToken);
    }

    public LoginResponse login(LoginRequest loginRequest) {
        String url = "https://identitytoolkit.googleapis.com/v1/accounts:signInWithPassword?key=" + firebaseApiKey;

        Map<String, Object> request = new HashMap<>();
        request.put("email", loginRequest.getEmail());
        request.put("password", loginRequest.getPassword());
        request.put("returnSecureToken", true);

        Map<String, Object> response = restTemplate.postForObject(url, request, Map.class);
        String idToken = (String) response.get("idToken");

        return new LoginResponse(idToken);
    }

    public UserRecord getUserByEmail(String email) throws FirebaseAuthException {
        return FirebaseAuth.getInstance().getUserByEmail(email);
    }

    public UserRecord updateUser(String uid, String email, String password)
            throws FirebaseAuthException {
        UpdateRequest request = new UpdateRequest(uid)
                .setEmail(email)
                .setPassword(password);

        return FirebaseAuth.getInstance().updateUser(request);
    }

    public void deleteUser(String uid) throws FirebaseAuthException {
        FirebaseAuth.getInstance().deleteUser(uid);
    }

    public FirebaseToken verifyIdToken(String idToken) throws FirebaseAuthException {
        return FirebaseAuth.getInstance().verifyIdToken(idToken);
    }
}
```

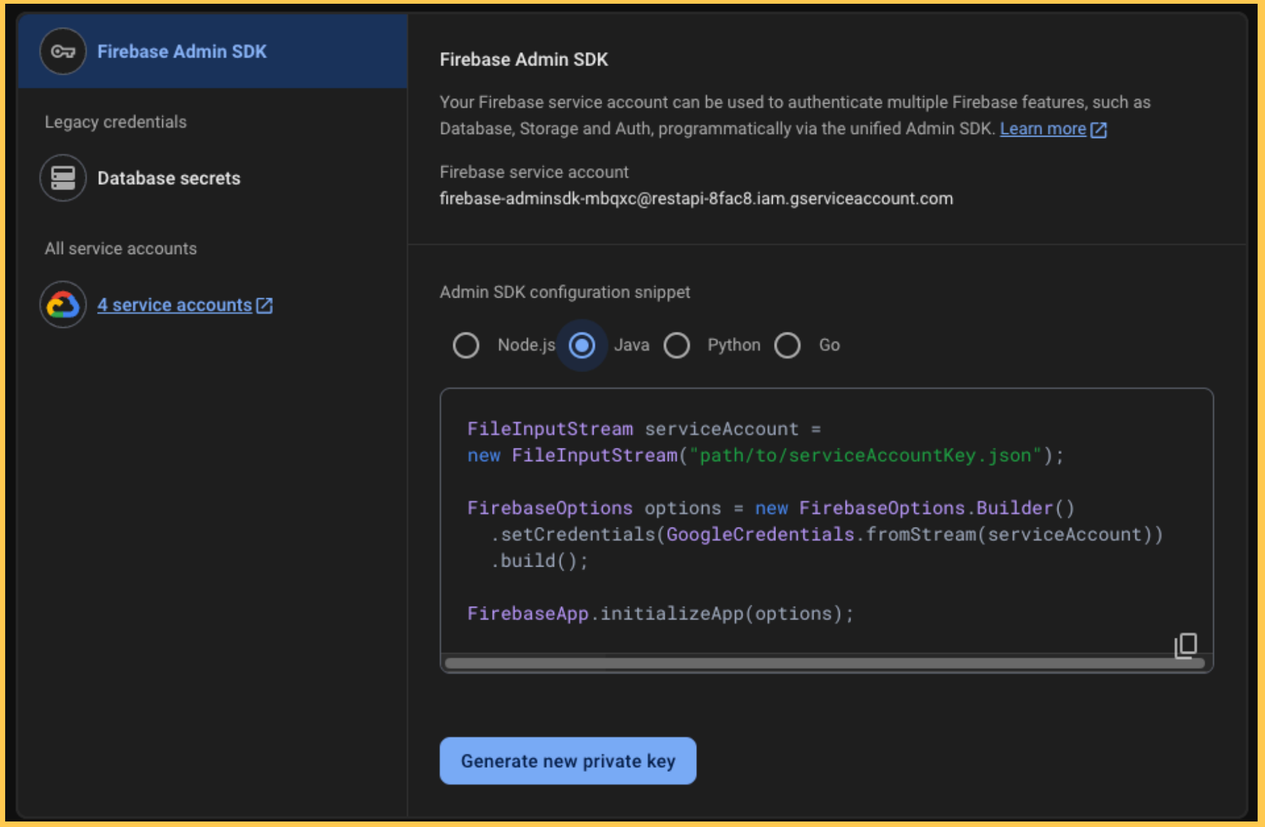
Setup Firebase authentication

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

```
<dependency>
    <groupId>com.google.firebase</groupId>
    <artifactId>firebase-admin</artifactId>
    <version>8.1.0</version>
</dependency>
```

Setup Firebase authentication

- Download secret key dari firebase project



Config

Penambahan Firebase Config

```java
package com.example.demo.security;

import com.google.auth.oauth2.GoogleCredentials;
import com.google.firebase.FirebaseApp;
import com.google.firebase.FirebaseOptions;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;

@Configuration
public class FirebaseConfig {

    @Bean
    public FirebaseApp initializeFirebase() throws IOException {
        File file = new File("src/main/resources/firebasekey.json");
        FileInputStream serviceAccount = new FileInputStream(file);

        FirebaseOptions options = new FirebaseOptions.Builder()
                .setCredentials(GoogleCredentials.fromStream(serviceAccount))
                .build();

        return FirebaseApp.initializeApp(options);
    }
}
```

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

```java
package com.example.demofb.security;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;
import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

@Configuration
@EnableWebSecurity
public class SecurityConfig {

    @Autowired
    private CustomUserDetailsService customUserDetailsService;

    @Autowired
    private FirebaseTokenFilter firebaseTokenFilter;

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http
                .authorizeRequests(authorizeRequests -> authorizeRequests
                        .requestMatchers("/api/auth/**").permitAll()
                        .requestMatchers("/api/users/**").permitAll()
                        .requestMatchers("/api/task/**").authenticated()
                        .anyRequest().permitAll())
                .csrf(csrf -> csrf.disable())
                .sessionManagement(
                        sessionManagement -> sessionManagement.sessionCreationPolicy(SessionCreationPolicy.STATELESS));

        http.addFilterBefore(firebaseTokenFilter, UsernamePasswordAuthenticationFilter.class);
        return http.build();
    }

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

```

Service

Penambahan Firebase Auth Service

```java
@Service
public class FirebaseAuthService {

    @Value("${firebase.api-key}")
    private String firebaseApiKey;

    private final RestTemplate restTemplate = new RestTemplate();

    public LoginResponse signup(LoginRequest signupRequest) {
        String url = "https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=" + firebaseApiKey;

        Map<String, Object> request = new HashMap<>();
        request.put("email", signupRequest.getEmail());
        request.put("password", signupRequest.getPassword());
        request.put("returnSecureToken", true);

        Map<String, Object> response = restTemplate.postForObject(url, request, Map.class);
        String idToken = (String) response.get("idToken");

        return new LoginResponse(idToken);
    }

    public LoginResponse login(LoginRequest loginRequest) {
        String url = "https://identitytoolkit.googleapis.com/v1/accounts:signInWithPassword?key=" + firebaseApiKey;

        Map<String, Object> request = new HashMap<>();
        request.put("email", loginRequest.getEmail());
        request.put("password", loginRequest.getPassword());
        request.put("returnSecureToken", true);

        Map<String, Object> response = restTemplate.postForObject(url, request, Map.class);
        String idToken = (String) response.get("idToken");

        return new LoginResponse(idToken);
    }

    public UserRecord getUserByEmail(String email) throws FirebaseAuthException {
        return FirebaseAuth.getInstance().getUserByEmail(email);
    }

    public UserRecord updateUser(String uid, String email, String password)
            throws FirebaseAuthException {
        UpdateRequest request = new UpdateRequest(uid)
                .setEmail(email)
                .setPassword(password);

        return FirebaseAuth.getInstance().updateUser(request);
    }

    public void deleteUser(String uid) throws FirebaseAuthException {
        FirebaseAuth.getInstance().deleteUser(uid);
    }

    public FirebaseToken verifyIdToken(String idToken) throws FirebaseAuthException {
        return FirebaseAuth.getInstance().verifyIdToken(idToken);
    }
}
```

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

Controller

Penambahan Auth Firebase Controller

```java
package com.example.demo.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.authentication.BadCredentialsException;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.web.bind.annotation.*;

import com.example.demo.model.LoginRequest;
import com.example.demo.model.LoginResponse;
import com.example.demo.service.FirebaseAuthService;

@RestController
@RequestMapping("/api/auth/firebase")
public class AuthFirebaseController {

    @Autowired
    private FirebaseAuthService firebaseAuthService;

    @PostMapping("/signup")
    public LoginResponse signup(@RequestBody LoginRequest signupRequest) {
        return firebaseAuthService.signup(signupRequest);
    }

    @PostMapping("/login")
    public LoginResponse authenticateUser(@RequestBody LoginRequest loginRequest) {
        try {
            return firebaseAuthService.login(loginRequest);
        } catch (BadCredentialsException e) {
            throw new RuntimeException("Invalid email or password");
        }

    }

    @PostMapping("/logout")
    public String logoutUser() {
        SecurityContextHolder.clearContext();
        return "User logged out successfully";
    }
}
```

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

Config

Penambahan Firebase Token Filter

```java
package com.example.demo.security;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseAuthException;
import com.google.firebase.auth.FirebaseToken;

import jakarta.servlet.FilterChain;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;
import org.springframework.stereotype.Component;
import org.springframework.web.filter.OncePerRequestFilter;

import java.io.IOException;

@Component
public class FirebaseTokenFilter extends OncePerRequestFilter {

    @Override
    protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain)
            throws ServletException, IOException {
        String header = request.getHeader("Authorization");
        if (header != null && header.startsWith("Bearer ")) {
            String idToken = header.substring(7);
            try {
                FirebaseToken decodedToken = FirebaseAuth.getInstance().verifyIdToken(idToken);
                FirebaseAuthenticationToken authentication = new FirebaseAuthenticationToken(decodedToken);
                authentication.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));
                SecurityContextHolder.getContext().setAuthentication(authentication);
            } catch (FirebaseAuthException e) {
                e.printStackTrace();
            }
        }
        filterChain.doFilter(request, response);
    }
}
```

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

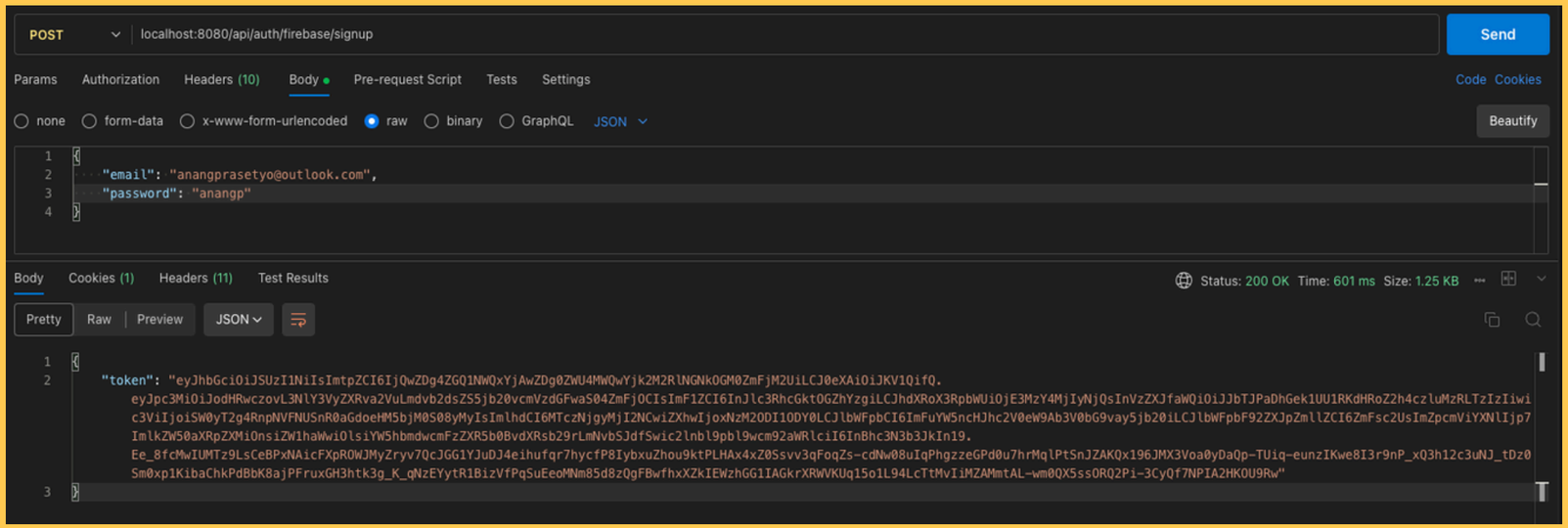
Penambahan Firebase Authentication Token

```java
package com.example.demo.security;

import com.google.firebase.auth.FirebaseToken;
import org.springframework.security.authentication.AbstractAuthenticationToken;

public class FirebaseAuthenticationToken extends AbstractAuthenticationToken {

    private final FirebaseToken firebaseToken;

    public FirebaseAuthenticationToken(FirebaseToken firebaseToken) {
        super(null);
        this.firebaseToken = firebaseToken;
        setAuthenticated(true);
    }

    @Override
    public Object getCredentials() {
        return null;
    }

    @Override
    public Object getPrincipal() {
        return this.firebaseToken;
    }

    public FirebaseToken getFirebaseToken() {
        return firebaseToken;
    }
}
```

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

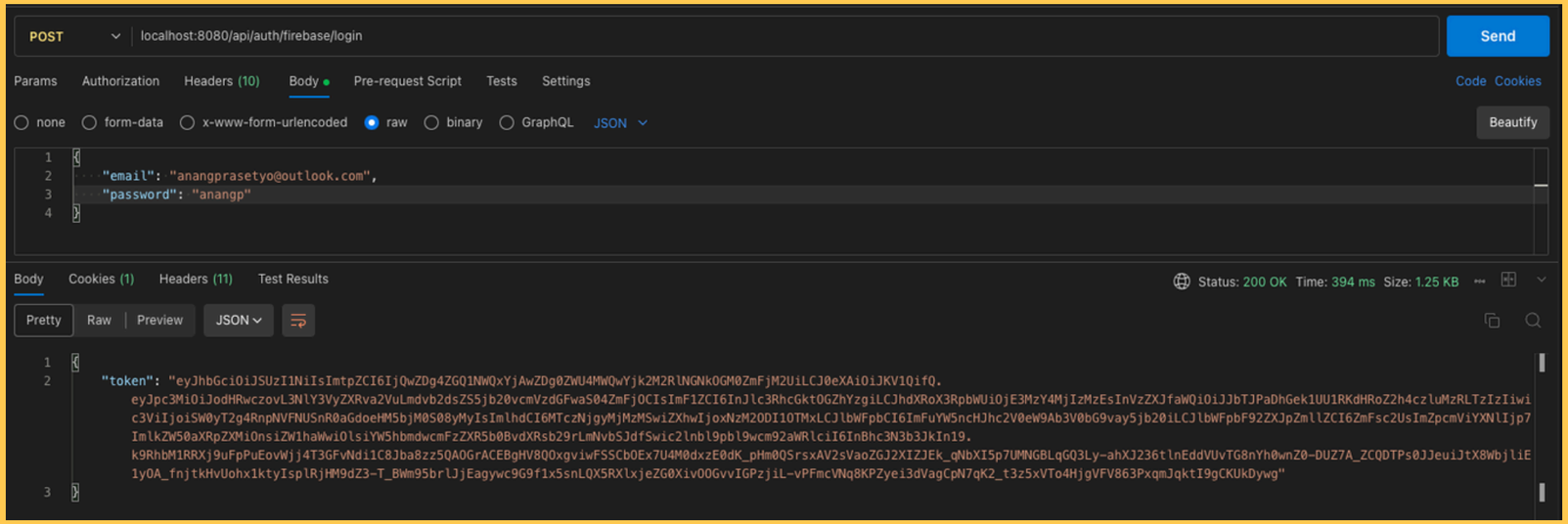
Testing

Endpoint Firebase Create User



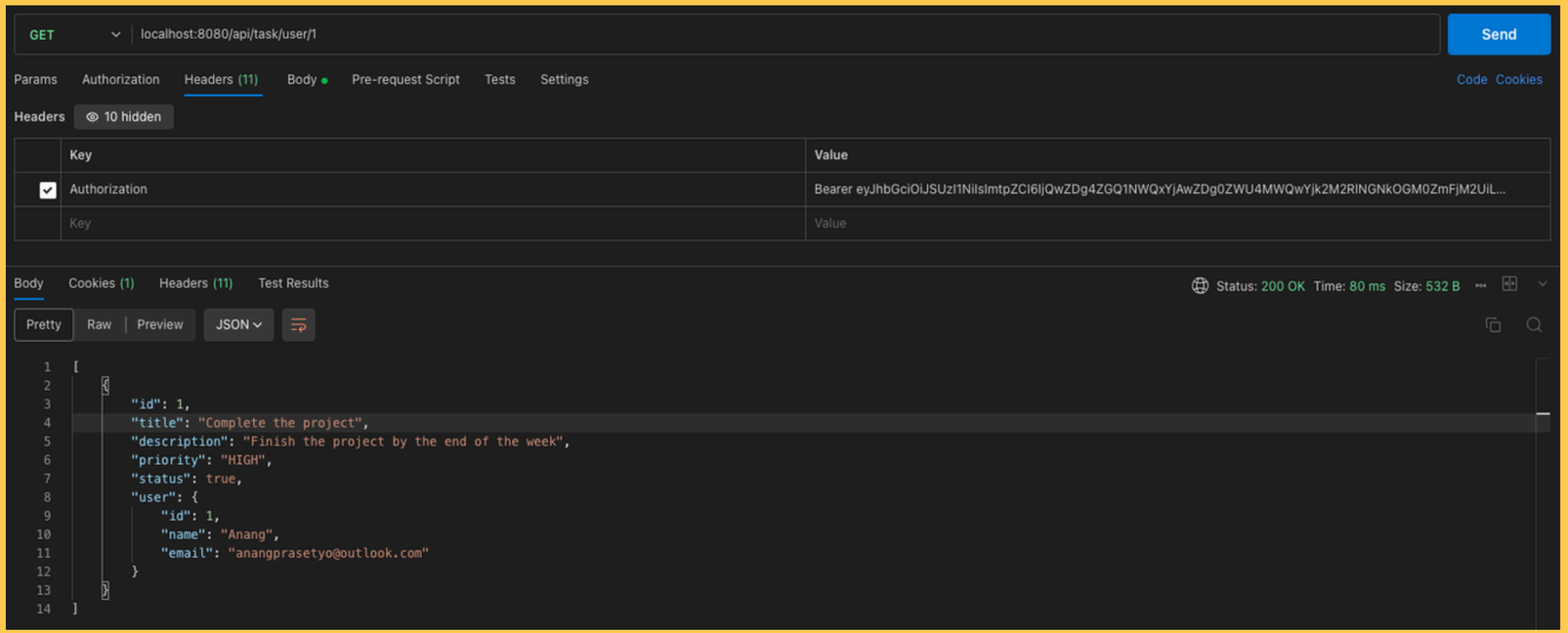
Testing

Endpoint Firebase User Login



Testing

Endpoint Task User



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/

BINUS CENTER

THANK YOU

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