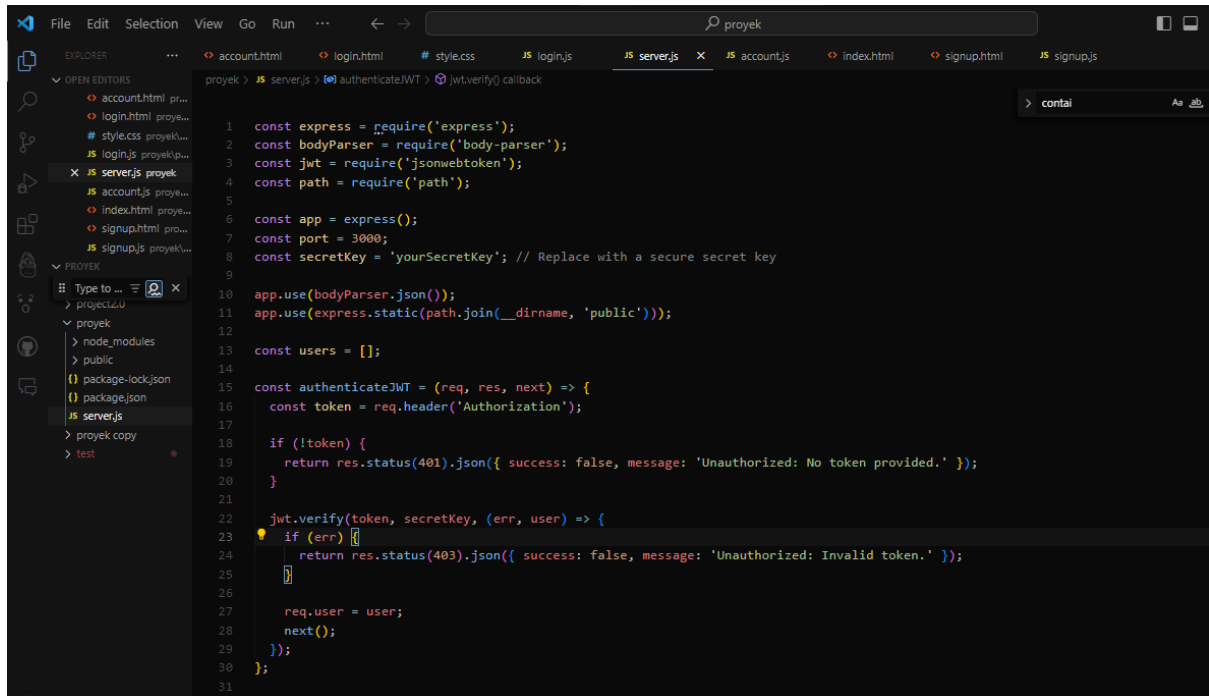


Nama : - Junaedi Samandias  
- Mozart Felix L

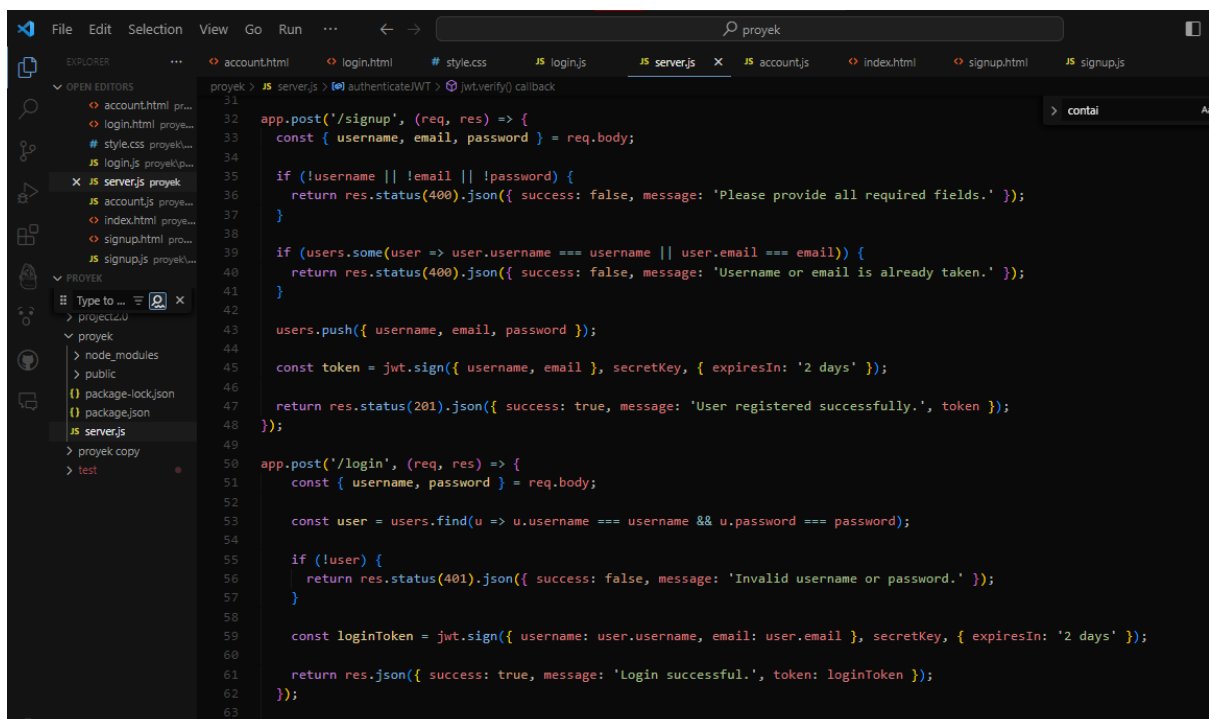
Kelas : 3IA11

API



The screenshot shows a Visual Studio Code editor window with a project named 'proyek'. The Explorer sidebar on the left shows the file structure, including 'server.js' which is currently selected. The main editor area displays the code for 'server.js'. The code includes imports for 'express', 'body-parser', 'jsonwebtoken', and 'path'. It sets up an Express app with static file serving and a JWT secret key. A 'users' array is initialized. The 'authenticateJWT' function checks for a token in the 'Authorization' header, verifies it with 'jwt.verify', and if successful, sets 'req.user' and calls 'next()'. If there's an error, it returns a 403 status with an 'Unauthorized: Invalid token' message.

```
1 const express = require('express');
2 const bodyParser = require('body-parser');
3 const jwt = require('jsonwebtoken');
4 const path = require('path');
5
6 const app = express();
7 const port = 3000;
8 const secretKey = 'yourSecretKey'; // Replace with a secure secret key
9
10 app.use(bodyParser.json());
11 app.use(express.static(path.join(__dirname, 'public')));
12
13 const users = [];
14
15 const authenticateJWT = (req, res, next) => {
16   const token = req.header('Authorization');
17
18   if (!token) {
19     return res.status(401).json({ success: false, message: 'Unauthorized: No token provided.' });
20   }
21
22   jwt.verify(token, secretKey, (err, user) => {
23     if (err) {
24       return res.status(403).json({ success: false, message: 'Unauthorized: Invalid token.' });
25     }
26
27     req.user = user;
28     next();
29   });
30 };
31
```



This screenshot continues the 'server.js' code from the previous image. It defines two POST endpoints: '/signup' and '/login'. The '/signup' endpoint checks if all required fields (username, email, password) are provided. It then checks if the username or email is already in the 'users' array. If not, it pushes the new user to the array, generates a JWT token with an expiration of 2 days, and returns a 201 status with a success message and the token. The '/login' endpoint checks if the provided username and password match an entry in the 'users' array. If they do, it generates a JWT token and returns a 200 status with a success message and the token. If not, it returns a 401 status with an 'Invalid username or password' message.

```
31
32 app.post('/signup', (req, res) => {
33   const { username, email, password } = req.body;
34
35   if (!username || !email || !password) {
36     return res.status(400).json({ success: false, message: 'Please provide all required fields.' });
37   }
38
39   if (users.some(user => user.username === username || user.email === email)) {
40     return res.status(400).json({ success: false, message: 'Username or email is already taken.' });
41   }
42
43   users.push({ username, email, password });
44
45   const token = jwt.sign({ username, email }, secretKey, { expiresIn: '2 days' });
46
47   return res.status(201).json({ success: true, message: 'User registered successfully.', token });
48 });
49
50 app.post('/login', (req, res) => {
51   const { username, password } = req.body;
52
53   const user = users.find(u => u.username === username && u.password === password);
54
55   if (!user) {
56     return res.status(401).json({ success: false, message: 'Invalid username or password.' });
57   }
58
59   const loginToken = jwt.sign({ username: user.username, email: user.email }, secretKey, { expiresIn: '2 days' });
60
61   return res.json({ success: true, message: 'Login successful.', token: loginToken });
62 });
63
```

```
app.get('/users', authenticateJWT, (req, res) => {  
  res.json({ success: true, user: req.user });  
});  
  
app.listen(port, () => {  
  console.log(`Server is running on http://localhost:${port}`);  
});
```

Hasil

The screenshot shows a web browser window with the address bar displaying 'localhost:3000/signup.html'. The browser has three tabs: 'ChatGPT', 'Register Form', and 'Register Form'. The main content area shows a 'Register' form with the following elements:

- Register** (Title)
- 
- 
- (with an eye icon for toggling visibility)
- 
- Already have an account? [Login here.](#)

