# Experiment-7.2

JWT Authentication for Secure Banking API Endpoints

**Code-**

## Banking API Server (server.js)

This file contains the complete code for the Node.js banking server with JWT authentication middleware.

```
const express = require('express');
const jwt = require('jsonwebtoken');
const app = express();

// Middleware to parse JSON
app.use(express.json());

// Secret key for JWT (In production, use environment
   variables!)
const JWT_SECRET = 'your-secret-key-change-in-production';

// --- 1. Hardcoded User Credentials (for demonstration only)
   ---
/**
 * In a real application, users would be stored in a database
 * with hashed passwords using bcrypt.
 */
const USERS = {
  'john': { password: 'password123', accountId: 'ACC001' },
  'jane': { password: 'securepass', accountId: 'ACC002' }
};

// --- 2. Mock Database for Account Balances ---
/**
 * In production, this would be replaced with a real database
 * like MongoDB, PostgreSQL, or MySQL.
 */
const accounts = {
  'ACC001': { balance: 5000, owner: 'john' },
  'ACC002': { balance: 3000, owner: 'jane' }
};

// --- 3. JWT Authentication Middleware ---
/**
 * This middleware function verifies the JWT token.
 * It will be applied to all protected routes.
 */
const authenticateToken = (req, res, next) => {
```

```
37    // Get token from Authorization header
38    const authHeader = req.headers['authorization'];
39    const token = authHeader && authHeader.split(' ')[1];
40    // Format: "Bearer TOKEN"
41
42    if (!token) {
43      return res.status(401).json({
44        error: 'Access denied. No token provided.'
45      });
46    }
47
48    try {
49      // Verify token
50      const decoded = jwt.verify(token, JWT_SECRET);
51      req.user = decoded; // Attach user info to request
52      next(); // Pass control to the next middleware/route
53    } catch (err) {
54      return res.status(403).json({
55        error: 'Invalid or expired token.'
56      });
57    }
58  };
59
60  // --- 4. Routes ---
61
62  /**
63   * Public Route: Login
64   * This route accepts username and password, validates
       credentials,
65   * and returns a signed JWT token upon successful
       authentication.
66   */
67  app.post('/login', (req, res) => {
68    const { username, password } = req.body;
69
70    // Validate input
71    if (!username || !password) {
72      return res.status(400).json({
73        error: 'Username and password are required.'
74      });
75    }
76
77    // Check credentials
78    const user = USERS[username];
79    if (!user || user.password !== password) {
80      return res.status(401).json({
81        error: 'Invalid username or password.'
82      });
83    }
84
85    // Generate JWT token
```

```
86    const token = jwt.sign(
87      {
88        username: username,
89        accountId: user.accountId
90      },
91      JWT_SECRET,
92      { expiresIn: '1h' } // Token expires in 1 hour
93    );
94
95    res.json({
96      message: 'Login successful',
97      token: token,
98      accountId: user.accountId
99    });
100 });
101
102 /**
103  * Protected Route: Get Account Balance
104  * This route has the `authenticateToken` middleware.
105  * 1. `authenticateToken` runs first to verify JWT.
106  * 2. The route handler runs if authentication succeeds.
107  */
108 app.get('/balance', authenticateToken, (req, res) => {
109   const accountId = req.user.accountId;
110   const account = accounts[accountId];
111
112   if (!account) {
113     return res.status(404).json({
114       error: 'Account not found.'
115     });
116   }
117
118   res.json({
119     accountId: accountId,
120     balance: account.balance,
121     owner: account.owner
122   });
123 });
124
125 /**
126  * Protected Route: Deposit Money
127  * This route requires JWT authentication via middleware.
128  */
129 app.post('/deposit', authenticateToken, (req, res) => {
130   const { amount } = req.body;
131   const accountId = req.user.accountId;
132   const account = accounts[accountId];
133
134   // Validate amount
135   if (!amount || amount <= 0) {
136     return res.status(400).json({
```

```
137         error: 'Invalid deposit amount. Must be greater than 0.'
138       });
139     }
140
141     if (!account) {
142       return res.status(404).json({
143         error: 'Account not found.'
144       });
145     }
146
147     // Process deposit
148     account.balance += amount;
149
150     res.json({
151       message: 'Deposit successful',
152       accountId: accountId,
153       amount: amount,
154       newBalance: account.balance
155     });
156 });
157
158 /**
159  * Protected Route: Withdraw Money
160  * This route requires JWT authentication via middleware.
161  * It also checks for sufficient balance before processing.
162  */
163 app.post('/withdraw', authenticateToken, (req, res) => {
164     const { amount } = req.body;
165     const accountId = req.user.accountId;
166     const account = accounts[accountId];
167
168     // Validate amount
169     if (!amount || amount <= 0) {
170       return res.status(400).json({
171         error: 'Invalid withdrawal amount. Must be greater than
      0.'
172       });
173     }
174
175     if (!account) {
176       return res.status(404).json({
177         error: 'Account not found.'
178       });
179     }
180
181     // Check sufficient balance
182     if (account.balance < amount) {
183       return res.status(400).json({
184         error: 'Insufficient balance.',
185         currentBalance: account.balance,
186         requestedAmount: amount
```

```
    });
  }

  // Process withdrawal
  account.balance -= amount;

  res.json({
    message: 'Withdrawal successful',
    accountId: accountId,
    amount: amount,
    newBalance: account.balance
  });
});


// --- 5. Start Server ---
const PORT = 3000;
app.listen(PORT, () => {
  console.log(`Banking API server running on http://localhost:${PORT}`);
  console.log('---');
  console.log('Available routes:');
  console.log('1. POST http://localhost:3000/login (Public)');
  console.log('   - Body: {"username": "john", "password": "password123"}');
  console.log('2. GET http://localhost:3000/balance (Protected)');
  console.log('3. POST http://localhost:3000/deposit (Protected)');
  console.log('4. POST http://localhost:3000/withdraw (Protected)');
  console.log('---');
  console.log('Protected routes require Header:');
  console.log('   "Authorization: Bearer <your-jwt-token>"');
  console.log('---');
  console.log('Available test users:');
  console.log('- Username: john, Password: password123');
  console.log('- Username: jane, Password: securepass');
  console.log('---');
  console.log('Watching for requests...');
});
```

Listing 1: server.js - JWT Banking API Server