```
;('const express = require('express
;('const cors = require('cors
;('const bcrypt = require('bcryptjs
;('const jwt = require('jsonwebtoken
;()const sqlite3 = require('sqlite3').verbose
;('const path = require('path

;()const app = express

Middleware //
;(())app.use(cors
;(())app.use(express.json
;((('/..' ,app.use(express.static(path.join(__dirname

// تكوين قاعدة البيانات
} <= (const db = new sqlite3.Database('./database/rescue.db', (err
    } (if (err
        ;(console.error('Error opening database:', err
    } else {
        ;('console.log('✅ Connected to SQLite database
        ;()initializeDatabase
    {
;({

// تهيئة الجداول
} ()function initializeDatabase {
    ) db.run(`CREATE TABLE IF NOT EXISTS users (
        ,id INTEGER PRIMARY KEY AUTOINCREMENT
        ,name TEXT NOT NULL
        ,email TEXT UNIQUE NOT NULL
        ,password TEXT NOT NULL
        ,role TEXT CHECK(role IN ('user', 'police', 'volunteer')) NOT NULL
        created_at DATETIME DEFAULT CURRENT_TIMESTAMP
    )`;

    ) db.run(`CREATE TABLE IF NOT EXISTS reports (
        ,id INTEGER PRIMARY KEY AUTOINCREMENT
        ,user_id INTEGER NOT NULL
        ,missing_name TEXT NOT NULL
        ,missing_age TEXT NOT NULL
        ,missing_type TEXT CHECK(missing_type IN ('ذكر', 'أنثى')) NOT NULL,
        ,last_location TEXT NOT NULL
        ,missing_description TEXT
        ,status TEXT CHECK(status IN ('جديد', 'قيد العمل', 'تم الحل')) DEFAULT 'جديد',
        ,created_at DATETIME DEFAULT CURRENT_TIMESTAMP
        (FOREIGN KEY (user_id) REFERENCES users (id
    )`;
```

```
// إضافة بيانات تجريبية
addSampleData();
{

} function addSampleData()
// التحقق من وجود بيانات تجريبية
db.get("SELECT COUNT(*) as count FROM users", (err, row) => {
  if (row.count === 0) {
    const hashedPassword = bcrypt.hashSync('123456', 10);

    db.run(`INSERT INTO users (name, email, password, role) VALUES (?, ?, ?, ?)`,
      ['أحمد محمد', 'user@example.com', hashedPassword, 'user']);

    db.run(`INSERT INTO users (name, email, password, role) VALUES (?, ?, ?, ?)`,
      ['شرطة الرياض', 'police@example.com', hashedPassword, 'police']);

    db.run(`INSERT INTO users (name, email, password, role) VALUES (?, ?, ?, ?)`,
      ['محمد المتطوع', 'volunteer@example.com', hashedPassword, 'volunteer']);

    db.run(`INSERT INTO reports (user_id, missing_name, missing_age, missing_type,
      last_location, missing_description) VALUES (?, ?, ?, ?, ?, ?, ?)`,
      [1, 'سارة أحمد', '15 سنة', 'أنثى', 'مركز المدينة التجاري', 'ترتدي فستان أزرق وتحمل حقيبة سوداء']);

    console.log('✅ Sample data added successfully');
  }
});

// Middleware المصادقة
const authenticateToken = (req, res, next) => {
  const authHeader = req.headers['authorization'];
  const token = authHeader && authHeader.split(' ')[1];

  if (!token) {
    return res.status(401).json({ error: 'الوصول مرفوض - توكن مطلوب' });
  }

  jwt.verify(token, 'your-secret-key', (err, user) => {
    if (err) return res.status(403).json({ error: 'توكن غير صالح' });
    req.user = user;
    next();
  });
};

// 🔐 routes المصادقة
app.post('/api/auth/register', async (req, res) => {
  try {
    const { name, email, password, role } = req.body;
```

```
    if (!name || !email || !password || !role) {
      return res.status(400).json({ error: 'جميع الحقول مطلوبة' });
    }

    // التحقق من وجود المستخدم
    db.get("SELECT * FROM users WHERE email = ?", [email], async (err, user) => {
      if (err) return res.status(500).json({ error: 'خطأ في الخادم' });
      if (user) return res.status(400).json({ error: 'البريد الإلكتروني مسجل مسبقاً' });

      // تشفير كلمة المرور
      const hashedPassword = await bcrypt.hash(password, 10);

      // إضافة المستخدم
      db.run(
        `INSERT INTO users (name, email, password, role) VALUES (?, ?, ?, ?)`,
        [name, email, hashedPassword, role],
        function(err) {
          if (err) return res.status(500).json({ error: 'خطأ في إنشاء الحساب' });

          res.status(201).json({
            message: 'تم إنشاء الحساب بنجاح',
            userId: this.lastID
          });
        }
      );
    });

  } catch (error) {
    res.status(500).json({ error: 'خطأ في الخادم' });
  }
});

app.post('/api/auth/login', async (req, res) => {
  try {
    const { email, password, role } = req.body;

    if (!email || !password || !role) {
      return res.status(400).json({ error: 'جميع الحقول مطلوبة' });
    }

    // البحث عن المستخدم
    db.get("SELECT * FROM users WHERE email = ? AND role = ?", [email, role], async
(err, user) => {
      if (err) return res.status(500).json({ error: 'خطأ في الخادم' });
      if (!user) return res.status(401).json({ error: 'المستخدم غير موجود' });

      // التحقق من كلمة المرور
```

```
;(const validPassword = await bcrypt.compare(password, user.password
;({ error: 'كلمة المرور غير صحيحة' }).if (!validPassword) return res.status(401).json

// إنشاء توكن
)const token = jwt.sign
,{ userId: user.id, role: user.role }
,'your-secret-key'
{ 'expiresIn: '24h }
;(

})res.json
,token
} :user {
,id: user.id
,name: user.name
,email: user.email
role: user.role
{
;({})
;({})

} (catch (error {
;({ error: 'خطأ في الخادم' }).res.status(500).json
{
;({})

// 📋 routes البلاغات
} <= (app.get('/api/reports', authenticateToken, (req, res
} <= (db.all("SELECT * FROM reports ORDER BY created_at DESC", (err, reports
;({ error: 'خطأ في جلب البلاغات' }).if (err) return res.status(500).json
;(res.json(reports
;({})
;({})

} <= (app.get('/api/reports/my-reports', authenticateToken, (req, res
db.all("SELECT * FROM reports WHERE user_id = ? ORDER BY created_at DESC",
} <= ([req.user.userId], (err, reports
;({ error: 'خطأ في جلب البلاغات' }).if (err) return res.status(500).json
;(res.json(reports
;({})
;({})

} <= (app.post('/api/reports', authenticateToken, (req, res
const { missingName, missingAge, missingType, lastLocation, missingDescription } =
;req.body

} (if (!missingName || !missingAge || !missingType || !lastLocation
;({ error: 'جميع الحقول المطلوبة' }).return res.status(400).json
```

```
{

)db.run
INSERT INTO reports (user_id, missing_name, missing_age, missing_type,`
,`(?, ?, ?, ?, ?, ?) last_location, missing_description) VALUES
req.user.userId, missingName, missingAge, missingType, lastLocation,]
,[missingDescription
} (function(err
;({ 'خطأ في إضافة البلاغ' :if (err) return res.status(500).json({ error

})res.json
,success: true
,'تم إضافة البلاغ بنجاح' :message
reportId: this.lastID
;({
{
;(
;({

// 👮 routes الشرطة
} <= (app.get('/api/police/reports', authenticateToken, (req, res
} ('if (req.user.role !== 'police
;({ 'غير مصرح بالوصول' :return res.status(403).json({ error
{

} <= (db.all("SELECT * FROM reports ORDER BY created_at DESC", (err, reports
;({ 'خطأ في جلب البلاغات' :if (err) return res.status(500).json({ error
;(res.json(reports
;({
;({

} <= (app.put('/api/police/reports/:id/status', authenticateToken, (req, res
} ('if (req.user.role !== 'police
;({ 'غير مصرح بالوصول' :return res.status(403).json({ error
{

;const { id } = req.params
;const { status } = req.body

} ((includes(status.['تم الحل' ,'قيد العمل' ,'جديد']!) if
;({ 'حالة غير صالحة' :return res.status(400).json({ error
{

} (db.run("UPDATE reports SET status = ? WHERE id = ?", [status, id], function(err
;({ 'خطأ في تحديث الحالة' :if (err) return res.status(500).json({ error

})res.json
,success: true
```

```
        message: `تم تحديث حالة البلاغ إلى: ${status}`
    });
  });
});

// routes 👥 المتطوعين
app.get('/api/volunteer/reports', authenticateToken, (req, res) => {
  if (req.user.role !== 'volunteer') {
    return res.status(403).json({ error: 'غير مصرح بالوصول' });
  }

  db.all("SELECT * FROM reports ORDER BY created_at DESC", (err, reports) => {
    if (err) return res.status(500).json({ error: 'خطأ في جلب البلاغات' });
    res.json(reports);
  });
});

// تشغيل السيرفر
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`🚀 السيرفر شغال على http://localhost:${PORT}`);
});
```