Configuration Manual

MSc Research Project
Cyber Security

Pradeep Kumar Reddy Elugoti
Student ID: X23192909

School of Computing
National College of Ireland

Supervisor: Joel Aleburu

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	PRADEEP KUMAR REDDY ELUGOTI		
Student ID:	X23192909		
Programme:	CYBER SECURITY	**Year:**	2024
Module:	Msc Cyber security Practicum 2		
Lecturer:	Joel Aleburu		
Submission Due Date:	12-12-2024		
Project Title:	Enhancing Security in Node.js Applications to Prevent SQL Injection		
Word Count:	745	**Page Count:**	6

Signature: Pradeep Kumar Reddy Elugoti

Date: 12-12-2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Pradeep Kumar Reddy Elugoti
Student ID: X23192909

1 System Requirements

1.1 Hardware Requirements:

- Minimum 4 GB RAM.
- At least 2 GHz CPU.
- 10 GB minimum of free disk space.

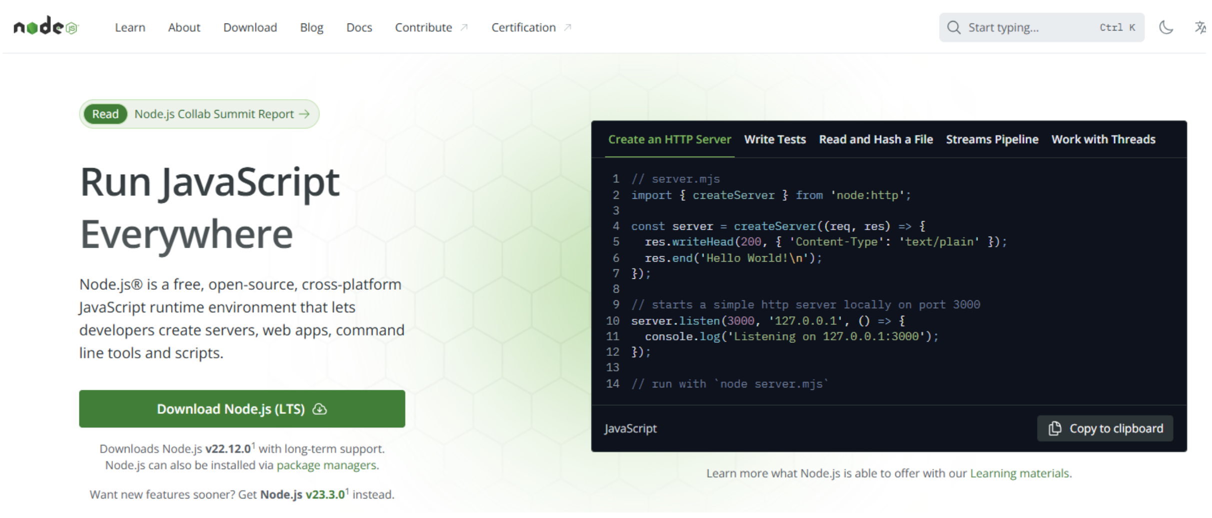
1.2 Software Requirements:

- Operating System: Windows 10/11, macOS, or Linux.
- Node.js: Version 16.0.0 or higher.
- MySQL Server: Version 8.0 or higher.
- OWASP ZAP: Latest stable version.
- IDE: Visual Studio Code or any equivalent code editor.
- Xampp Server

2 Application Setup

Installing Node.js:

- Download Node.js from Node.js Official Website.
- Install Node.js and npm (Node Package Manager).



Run JavaScript Everywhere

Node.js® is a free, open-source, cross-platform JavaScript runtime environment that lets developers create servers, web apps, command line tools and scripts.

Downloads Node.js **v22.12.0**[1] with long-term support.
Node.js can also be installed via package managers.

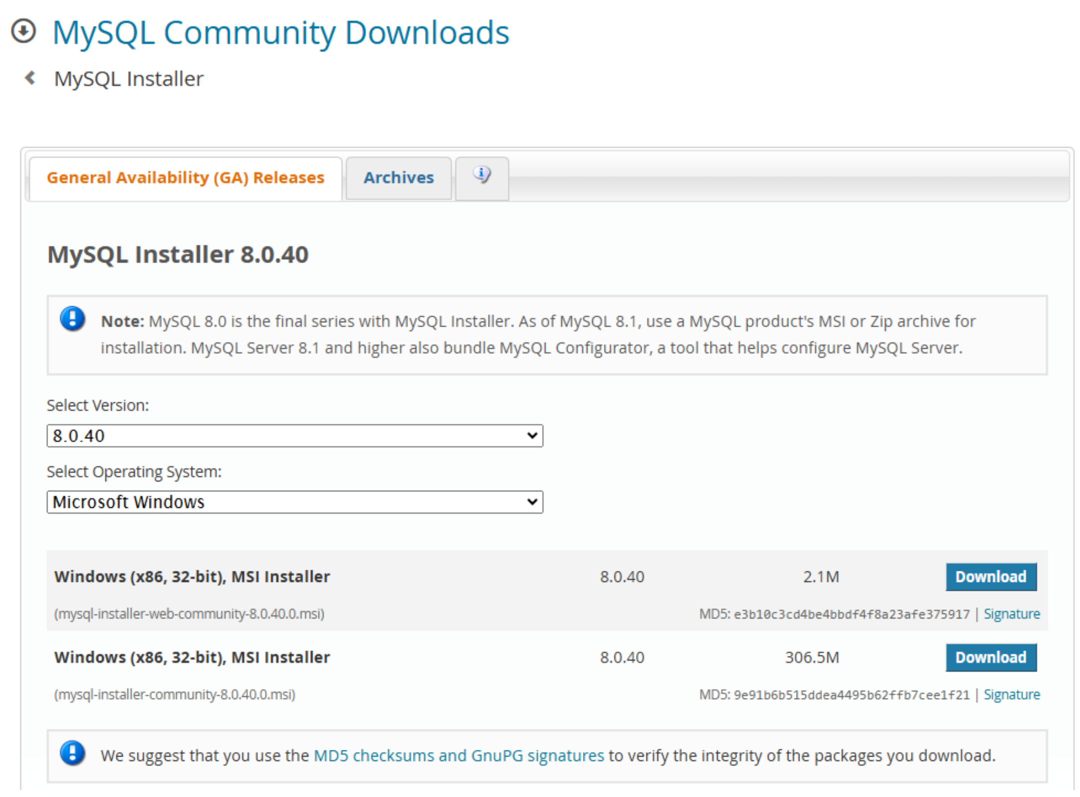
Want new features sooner? Get **Node.js v23.3.0**[1] instead.

```javascript
// server.mjs
import { createServer } from 'node:http';

const server = createServer((req, res) => {
  res.writeHead(200, { 'Content-Type': 'text/plain' });
  res.end('Hello World!\n');
});

// starts a simple http server locally on port 3000
server.listen(3000, '127.0.0.1', () => {
  console.log('Listening on 127.0.0.1:3000');
});

// run with `node server.mjs`
```

Learn more what Node.js is able to offer with our Learning materials.

Installing MySQL:

- Download MySQL Server from [MySQL Official Website](#).
- Configure the root user with a secure password.


MySQL Community Downloads

MySQL Installer

MySQL Installer 8.0.40

Note: MySQL 8.0 is the final series with MySQL Installer. As of MySQL 8.1, use a MySQL product's MSI or Zip archive for installation. MySQL Server 8.1 and higher also bundle MySQL Configurator, a tool that helps configure MySQL Server.

Select Version: 8.0.40

Select Operating System: Microsoft Windows

Windows (x86, 32-bit), MSI Installer	8.0.40	2.1M	Download
(mysql-installer-web-community-8.0.40.0.msi)		MD5: e3b10c3cd4be4bbdf4f8a23afe375917 | Signature	
Windows (x86, 32-bit), MSI Installer	8.0.40	306.5M	Download
(mysql-installer-community-8.0.40.0.msi)		MD5: 9e91b6b515ddea4495b62ffb7cee1f21 | Signature	

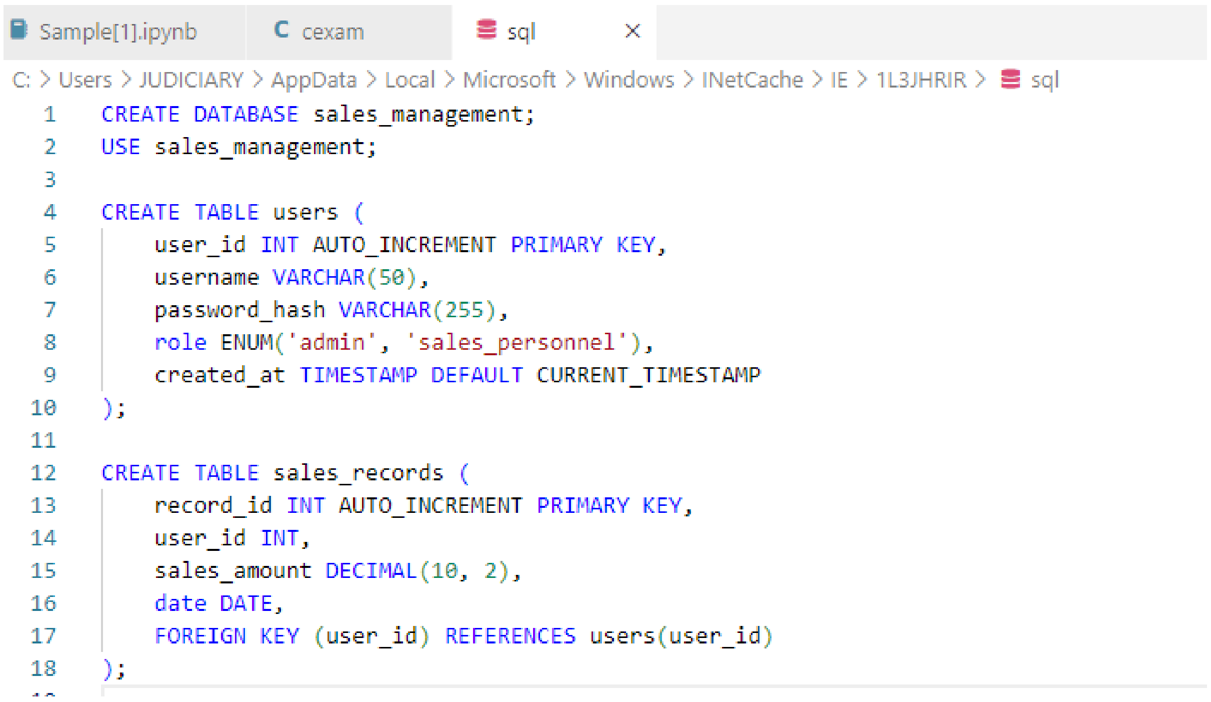
We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download.

Installing Dependencies:

- Open the terminal/command prompt in your project directory.
- Run npm install to download all dependencies from package.json.

Setting Up the Database:

- Use the provided SQL script to create the required tables


Sample[1].ipynb
cexam
sql
C: > Users > JUDICIARY > AppData > Local > Microsoft > Windows > INetCache > IE > 1L3JHRIR > sql

```sql
CREATE DATABASE sales_management;
USE sales_management;

CREATE TABLE users (
    user_id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50),
    password_hash VARCHAR(255),
    role ENUM('admin', 'sales_personnel'),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE sales_records (
    record_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    sales_amount DECIMAL(10, 2),
    date DATE,
    FOREIGN KEY (user_id) REFERENCES users(user_id)
);
```

3 Testing Procedures

3.1 Manual Testing:

- Test for SQL injection vulnerabilities using crafted payloads.
- Verify functionality of user authentication and sales record submission.

3.2 Automated Testing:

- Run OWASP ZAP scans to identify security vulnerabilities.
- Validate the absence of SQL injection vulnerabilities in the test results.

3.3 Performance Testing:

- Monitor application response times after applying security measures.

4 Deployment Instructions

4.1 Local Deployment

Conversely for the testing and running the application locally then the XAMPP was used to create the related server environment. And last but not least, XAMPP offers Apache, MySQL, PHP, and Perl integrated pack allowing to easily install local development environment. Once the installation of XAMPP is complete and once the application has been started the MySQL server has to be set up to host the application database. For purpose of creating and populating the required tables, the SQL script developed in the project is run through the XAMPP MySQL console or phpMyAdmin.

When the database is set the Node.js application can be initiated by typing the command npm start in the terminal from project directory. By default, the application will run on http://localhost:3000, that is, through any browser on the Internet, one can enter a system. As such, this kind of structure provides a proper local environment for testing new features such as user login, submission of sales records, and protection against SQL injection attacks.

4.2 Production Deployment

Production deployment can be made on hosting platforms like AWS, Heroku, or any other cloud services for scalability and reliability. In deploying to production, HTTPS should be set up so as to secure communication and protect sensitive data, such as user credentials and/or financial records. Besides, the.env file holding sensitive configurations.

The same is the case with the production environment, which should adhere to strict security guidelines as laid down by the database for parameterized queries and access control to prevent unauthorized access. Regular backups and replication setups can further safeguard the data.

4.3 Continuous Monitoring

It is highly recommended that continuous monitoring tools be integrated to ensure the integrity and performance of the application in real time. These tools monitor server performance metrics, identify anomalies, and detect potential security threats. For instance, monitoring tools such as New Relic or Datadog can provide insight into server health and application performance, while vulnerability-scan testing tools like OWASP ZAP can be run periodically in production.

The combination of XAMPP for local testing with a robust production deployment strategy and continuous monitoring will guarantee both secure and highly reliable operations of the Node.js app when used in real-world environments.