**Secure Banking Application Project:**

**Web Application (65%)**

Department of Computing, Atlantic Technological University Letterkenny, Donegal

by

Michelle Ogunade

L00176523

Lecturer: Maria Griffin

Table of Contents

[Introduction 3](#_Toc197099508)

[Creating the project 3](#_Toc197099509)

[Functionality 4](#_Toc197099510)

[Conclusion 9](#_Toc197099511)

[References 9](#_Toc197099512)

## Introduction

**Introduction to the report (6/7 lines), tell the reader what this report is about, explain that you are going to describe the application that you created and how you made sure it would be secure.**

This report outlines the development of a Secure Banking Web Application created as part of the CA3 assignment for the Secure Programming module. The application was developed to simulate a simple banking system that prioritises secure coding practices. I had an easier choice to approach this but I opted to challenge myself to develop this bank system as a web application.

In this report, I will describe how the application was created, what technologies were used, and how secure development principles such as authentication, password hashing, and secure data handling were implemented and tested.

## Creating the project

**Tell the reader how you created the project, did you use VS Code? MySQL Developer? Git Hub? XAMPP? What language did you choose, Java, C#, Python, explain why.**

This secure banking web application was developed using Visual Studio Code as the primary IDE and MySQL Server for managing the relational database.

**Languages & Tools Used:**

* JavaScript, Java (partially), HTML, CSS, Bootstrap
* MySQL Server, SQL queries, Node.js, Express.js
* GitHub for version control, GitHub Desktop

I used GitHub for version control throughout the development of this project. It allowed me to efficiently track changes and roll back to previous versions when necessary.

I chose JavaScript as my primary programming language for this system because I’ve had more experience with it and really enjoy developing frontend interfaces.

For the backend logic, I initially tried to use only plain JavaScript with Live Server, avoiding node\_modules to reduce setup complexity. However, due to the limitations of this approach, I later reintroduced Node.js (v20.4.0) and Express.js, which significantly improved development efficiency.

**Any issues you had with your set up. Did you use your own laptop, ATU machines, Virtual Machine, Hard Drive. You can be specific with the versions on the language that you used and the spec of your laptop etc.**

To create my banking system, I had to plan what pages I wanted to include and display to the user and the workflow logic.

I primarily used the ATU Dell laptop for development, although I occasionally used ATU lab machines. Compatibility issues on lab machines, particularly related to running JavaScript code in Visual Studio Code, posed minor setbacks.

The system specifications were:

* **Processor:** 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz
* **Installed RAM:** 16.0 GB (15.4 GB usable)
* **System type:** 64-bit operating system, x64-based processor

**Users**

I had to identify what users and their associated roles within the system would be. However, decided to take this out.

## Functionality

**Overview of how your project works. Details the technologies that you used. What version. Explain the basic features of your application…. Login, Account creation, balance checking, deposits, and withdrawals etc.**

On arriving on the web application, the user has the option of logging in (if they have already created an account previously), Or on the Register Page, the user can create an account.

On the Account Information page, would have displayed all the users details, and if they wanted to view or edit them. The customer could’ve viewed their account details, check their balance, deposits and withdrawals. Was’t an important feature.

**Main Features:**

* **User Registration & Login**: Users can securely register and log into their accounts.
* **Transaction Handling**: Users can perform deposits and withdrawals, with updates reflected in real-time.

**How robust is it?**

**Talk about how your application handles errors gracefully, maintains consistent behaviour under different conditions, and recovers from failures.**

**Include screen shots…**

* Prepared Statements
* Error handling and validation
* Alerts

**How have you included Secure Coding Practices, show where you used prepared statements, password hashing, input validation, and data sanitisation. Details with the screenshots the Static Analysis tool(s) that you used. How did static analysis help you with developing your code?**

**Detail how you implemented Security Features such as password hashing and multi-factor authentication. 2FA does not have to work but should be implemented in the code and you need to show how you would do it and explain how it works. Detail the vulnerabilities of the system, identify potential security vulnerabilities and how you would mitigate against them**

Prepared Statements: Used to prevent SQL Injection attacks.

Password Hashing: Implemented using bcryptjs to securely store user passwords.

Input Validation & Sanitization: All user input is validated on both client and server sides to avoid XSS and data corruption.

A screen shot of a computer program

AI-generated content may be incorrect.

Two-Factor Authentication (2FA): Implemented using a placeholder function and UI input field to simulate 2FA logic.

**Vulnerabilities Identified & Mitigation:**

* I discovered delayed registration due to synchronous JavaScript and resolved by integrating async Node.js logic.
* Another thing, I identified lack of session timeout, so I proposed implementing inactivity-based logout.
* Didn’t notice that the customer was able to withdraw more than was in the current balance on time, I would’ve fixed that immediately.

**Show how you tested the application, explain how you tested the security features *e.g. secure login***

**Manual Testing:** Verified all routes, forms, and database interactions manually using test users.

**A close-up of a computer code

AI-generated content may be incorrect.**

Developed basic unit tests for registration and login logic using Junit5, but didn’t have enough time to develop it any further.

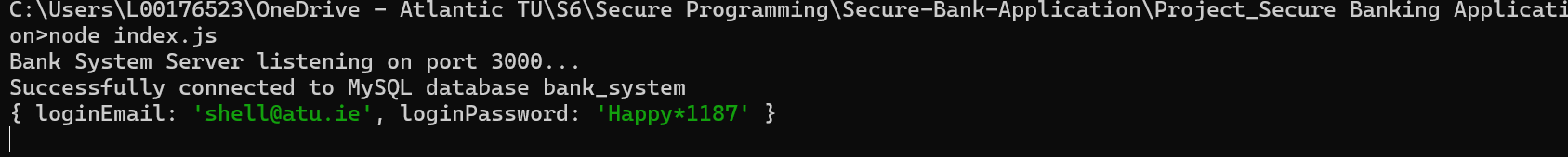
**Security Feature Testing:**

* I attempted SQL injection on login, was blocked by prepared statements
* I checked password hash format in database which confirmed bcrypt was working and tested it in the terminal, I uncommented it out as not to send out information

A screenshot of a login form

AI-generated content may be incorrect.

A computer screen shot of a code

AI-generated content may be incorrect.A screenshot of a login screen

AI-generated content may be incorrect.A screenshot of a login form

AI-generated content may be incorrect.A screen shot of a computer program

AI-generated content may be incorrect.

**Detail your unit tests add screen shots**

A computer screen shot of a program

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.

**Sadly, didn’t have the best experience with the unit testing due to time and focus on getting the users in the database displaying, however, the project is mostly secure.**

**Also, could’ve learned about unit testing in JavaScript and read about it prior to this.**

**Did you find any bugs, did you find any issues/holes in your security?**

Yes, it wasn’t exactly easy using JavaScript to build it, it took some time for a user to be registered into the database, as I was avoiding using node\_modules to begin with for less stress. But I’ve had enough experience with node\_modules in the past with previous modules, I believed it would help

A screenshot of a bank account

AI-generated content may be incorrect.

Main issue was that the salt hashing was being created as I could display it in the terminal, and the customer information was sent but the details weren’t being displayed in the database – which took a long time working out, couldn’t understand why as there were no errors.

However, a new salt was generated for a password and used for each new customer that registered.

## Conclusion

**What did you learn from doing this report, how secure do you think your application was overall, what would you change/do differently?**

I believe that my web application is very secure overall. A lot of research and preparation advance needs to be done. I found the project both challenging and insightful overall. Testing especially wasn’t fully done to a good standard.

**What was the most challenging part. Did anything surprise you as being easier/harder, something that you didn’t think was needed, or anything that you did that you realised wasn’t necessary or could be done a different or better way?**

The most challenging part I think was understanding the workflow in chronological order. I was struggling to pick between Java and JavaScript as my main programming language to use for the project.

Although due to many assignments at the same time, with very close deadlines i.e. 2 or 3 on the same day, it was always hard knowing how much time should be given to this project, although I did enjoy it, but it was always frustrating switching between different modules.

I feel more comfortable writing code in JavaScript as I have not practiced Java in years so it felt like a whole new language again. At the same time, I was only a little bit familiar with Unit Testing in Javascript so that was also something I had to learn, but I enjoy the learning aspect of it. I wish I had more time to read more into things or if I could find examples that were similar to my implementation.

## References

Any references you might use should go here…