**Capstone Final Report**



Kalekidan Bekele, Devin Samuda

Bowie State University

Professor Appolo Tankeh

COSC 480: Senior Capstone

May 11, 2023

# Introduction

This report provides the final updates on the development of our banking application. The project aims to provide customers with a modern and secure banking experience through a user-friendly mobile application. The application will enable Bowie State University customers to perform various banking activities, including account management, account authentication, user registration, balance inquiry, and transactions made from deposits and withdrawals.

During the entire semester, our team has made significant progress toward achieving the project goals. We have completed several major milestones, including the styling of each page, which has been done, with only a few touches required to meet the standards of professional web applications. We have also added functionality to most of the pages, such as login, signup, reset, and transactions.

To build the website, we are currently utilizing a combination of several key tools, including JavaScript for dynamic actions, HTML such handlebars to use values from Node JS for web page content, CSS for styling, MySQL to store and retrieve user’s data, and Node JS for server-side operations. These tools allow us to create a fully functional and visually appealing website that meets our needs and those of our users.

# Services

The project aimed to showcase the potential of NodeJS in developing a banking webpage as part of a software team. Throughout the semester, we gained experience creating registration forms for sign-up and sign-in, setting up databases, and connecting them through Sequelize. We completed the necessary tasks to achieve our goal, but due to time constraints and incomplete understanding, the project may not have turned out as polished as we had hoped. However, despite the challenges, we managed to incorporate all necessary functionalities into the starting file, index.js and successfully hosted the application with all requirements met.

To use the application, users must first sign up by creating an account. They will need to enter their first and last names, email, and password to complete the form. One potential error they may encounter is if the password and confirm password inputs do not match. This results in a flash warning stating that the passwords do not match. Additionally, if the user's email already exists within the database, they will receive a flash message stating: "email already exists." Once the account is successfully created, the user will be redirected to the sign-in form to officially log in. They will receive a default balance of $100 to get started.

We have added a new feature that enables users to reset their forgotten passwords by filling out a form. This form has a similar error handling system as the sign-up form. However, instead of showing an error message when the email address already exists, it will display an error message when the email address doesn't exist.

To access your account, you'll need to fill out a sign-in form built using a handlebar template file. This allows for dynamic data display not possible with HTML. You'll need to input your email (which serves as your username) and password. If you enter incorrect login details, a warning message will appear. This will let you know that the username or password you entered doesn't match any records in the MySQL ClearDB database.

Upon logging into the account, the user will be greeted with a welcome message displaying their email and current balance. The interface contains a single input field and two buttons that allow the user to either deposit or withdraw funds from their balance. To ensure successful transactions, the inputted amount must meet the following requirements: it must be a numerical value, greater than $0.00, and less than $10,000. If these requirements are met, the balance will be adjusted accordingly, and a flash message will appear indicating the amount corresponding to the button clicked.

There are additional pages that allow users to exchange specific amounts with other users who have an account, as well as view transaction history. However, time constraints hindered the development of these features. If an error occurs that is not accurately described, additional error handling is implemented. Overall, the application works well for sign-in, sign-up, deposit, withdrawal, and error display when used correctly.

# Node JS

At first, our application didn't utilize NodeJS because we were unfamiliar with it. Instead, we utilized PHP and the phpMyAdmin database, which linked well with HTML and PHP. However, as the project progressed, we needed to incorporate NodeJS, so we adapted the application's functionalities to suit the platform. Once we successfully transitioned to NodeJS, the application was closer to completion. While the sign-in and sign-up forms worked, the database only connected with the phpMyAdmin database. After several weeks of experimentation and learning, we discovered that we needed Sequelize to simplify connecting to the MySQL workbench and reduce the need to write full MySQL statements.

# Hosting

We tried using AWS to host our application, but we ran into issues with the database created in EC2 not connecting to the application. Since AWS requires payment, we decided to switch to Heroku. However, this transition was not seamless because Heroku requires the use of a remote database. We looked for solutions and found that Heroku offers various add-on databases. We also discovered that Heroku's solutions are better suited for macOS than Windows, so we had to look for external sources for the right solutions. After we established a database that resembles MySQL, we were able to store user data in the application.

# Setting Up the Application

To initiate the application from the beginning, you must clone it from the GitHub page: <https://github.com/KalekidanBekele/COSC480-Bank.git>. Once it is cloned into the desired directory, you must run a command to install all the needed packages into the node\_modules folder. Next, you must rewrite the connection to the MySQL workbench from the route/connection.js file since the information already present connects to the Heroku Add-on database, ClearDB.

For example, by default the setting are:

const sequelize = new Sequelize(

   'bankproject', //the database name

   'root',// the username

   '', // the password

    {

      host: 'localhost', // the host

      dialect: 'mysql',

      createDatabaseIfNotExist: true

    }

  );

You have to reconfigure the lines that have comments stating what the lines means.

Once the changes are saved and the connection to the database is secure, you can run node index.js. The website will run locally using localhost:3000 as the domain.

To run the application on a hosted website, you can use Heroku. It will be an easy transition since Heroku can link to the GitHub repository where the application is stored. Once linked and connecting the ClearDB, you’ll have to push the cloned application into Heroku’s master repository to officially run the website. At the end of pushing the application it will provide the website link to access the application.

# Conclusion

In conclusion, our collaborative efforts have resulted in the successful creation of a basic banking application using NodeJS, MySQL, and Sequelize. Throughout the development process, we encountered challenges as we transitioned platforms from PHP to NodeJS. This shift required us to adapt our coding practices and familiarize ourselves with new frameworks and libraries. Additionally, hosting our application proved to be another hurdle we faced. By leveraging Heroku, we navigated the complexities of deploying our project to the cloud and configuring the necessary environment variables. This experience not only expanded our technical expertise but also taught us valuable lessons in managing web applications in a production environment.

# Pictures

A screenshot of a login screen

Description automatically generated with medium confidenceA screenshot of a login form

Description automatically generated with low confidenceA screenshot of a login form

Description automatically generated with medium confidenceA screenshot of a login screen

Description automatically generated with medium confidenceA screenshot of a login form

Description automatically generated with medium confidenceA screen shot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with low confidenceA screenshot of a website

Description automatically generated with medium confidenceA screenshot of a phone

Description automatically generated with low confidenceA screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidence