Capstone Final Report

Kalekidan Bekele, Devin Samuda

Bowie State University

Professor Appolo Tankeh

COSC 480: Senior Capstone

May 11, 2023

Table of Contents

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, which is 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

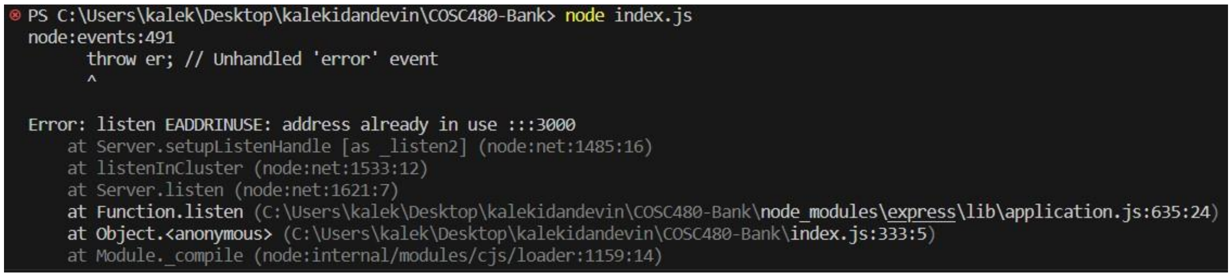
At first, our application didn't utilize NodeJS because we were unfamiliar with it. Instead, we utilized PHP and the phpMyAdmin database, which linked well with HTML and PHP. However, as the project progressed, we needed to incorporate NodeJS, so we adapted the application's functionalities to suit the 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.

The index.js file is crucial to the application's functionality. It utilizes several packages, such as express, session, body-parser, handlebars, flash, and path, to connect the application to the database and website. Middleware is then configured to modify the response and request objects, ensuring smooth execution of the code. Sequelize is used to authenticate the database connection. Lastly, routes are defined to properly display handlebar files and allow for the manipulation of user-inputted data through GET and POST methods.

An example of how the login route works are that it allows the handlebar file to be displayed as an HTML due to an app engine added in the middleware. When using the get login route, it can redirect to the page that contains the /login within the action parameters. This also enables flash messages to appear if the if statement conditions are true. For the post login route, it retrieves the data inputted on the website and checks if the email and password are valid. If they are not, then the if statement condition becomes true and outputs one of the error messages. If the account does exist, the email is searched and saved in a value that redirects to the get home route to view the greeting message and balance.

Hosting

When the application starts, it first checks whether port 3000 is available or not. If the port is already in use, an error message will appear. However, if the port is available, the server will start, and a console message will confirm that the server has started on port 3000.

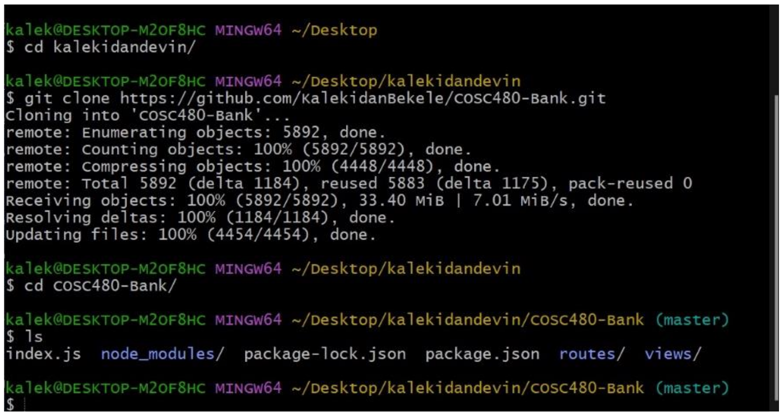


```
⊗ PS C:\Users\kalek\Desktop\kalekidandevin\COSC480-Bank> node index.js
  node:events:491
        throw er; // Unhandled 'error' event
        ^

  Error: listen EADDRINUSE: address already in use :::3000
      at Server.setupListenHandle [as _listen2] (node:net:1485:16)
      at listenInCluster (node:net:1533:12)
      at Server.listen (node:net:1621:7)
      at Function.listen (C:\Users\kalek\Desktop\kalekidandevin\COSC480-Bank\node_modules\express\lib\application.js:635:24)
      at Object.<anonymous> (C:\Users\kalek\Desktop\kalekidandevin\COSC480-Bank\index.js:333:5)
      at Module._compile (node:internal/modules/cjs/loader:1159:14)
```

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.

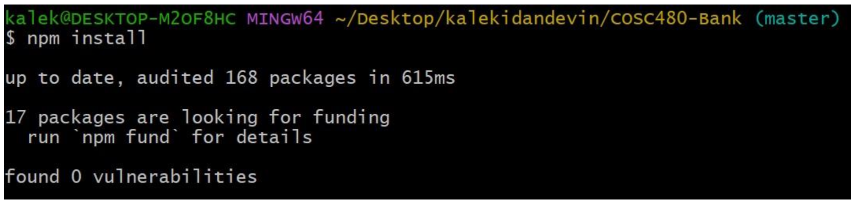
Setting Up the Application

To initiate the application from the beginning, you must clone it from the GitHub page: https://github.com/KalekidanBekele/COSC480-Bank.git.



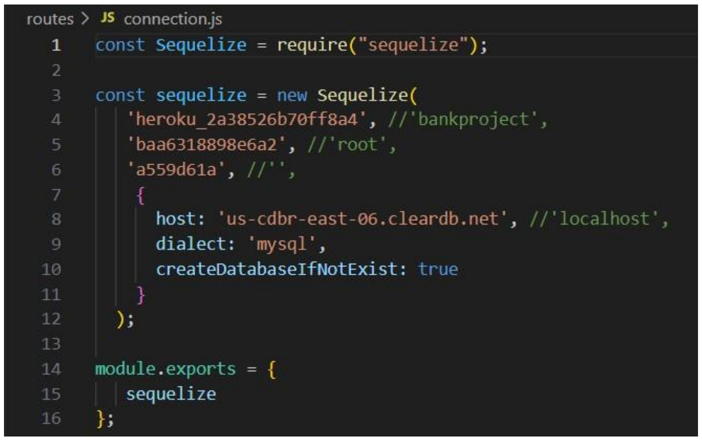
```
kalek@DESKTOP-M2OF8HC MINGW64 ~/Desktop
$ cd kalekidandevin/

kalek@DESKTOP-M2OF8HC MINGW64 ~/Desktop/kalekidandevin
$ git clone https://github.com/KalekidanBekele/COSC480-Bank.git
Cloning into 'COSC480-Bank'...
remote: Enumerating objects: 5892, done.
remote: Counting objects: 100% (5892/5892), done.
remote: Compressing objects: 100% (4448/4448), done.
remote: Total 5892 (delta 1184), reused 5883 (delta 1175), pack-reused 0
Receiving objects: 100% (5892/5892), 33.40 MiB | 7.01 MiB/s, done.
Resolving deltas: 100% (1184/1184), done.
Updating files: 100% (4454/4454), done.

kalek@DESKTOP-M2OF8HC MINGW64 ~/Desktop/kalekidandevin
$ cd COSC480-Bank/

kalek@DESKTOP-M2OF8HC MINGW64 ~/Desktop/kalekidandevin/COSC480-Bank (master)
$ ls
index.js  node_modules/  package-lock.json  package.json  routes/  views/

kalek@DESKTOP-M2OF8HC MINGW64 ~/Desktop/kalekidandevin/COSC480-Bank (master)
$
```

After cloning the git with the command "git clone" into the desired directory, it is necessary to execute a command that installs all the required packages into the "node_modules" folder.



```
kalek@DESKTOP-M2OF8HC MINGW64 ~/Desktop/kalekidandevin/COSC480-Bank (master)
$ npm install

up to date, audited 168 packages in 615ms

17 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

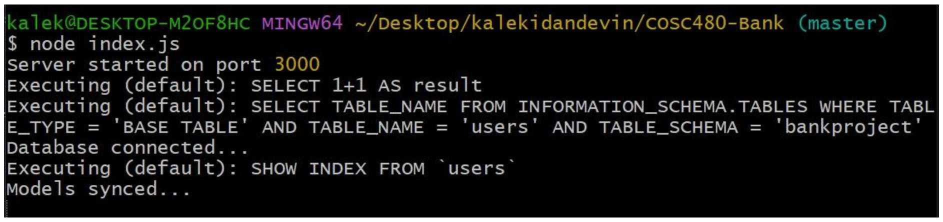
To proceed, it is necessary to make some adjustments to the connection between the MySQL workbench and the route/connection.js file. This is because the current information in use is intended for connecting to the Heroku Add-on database, ClearDB. Therefore, rewriting the connection is required to ensure everything works as intended.



routes > JS connection.js

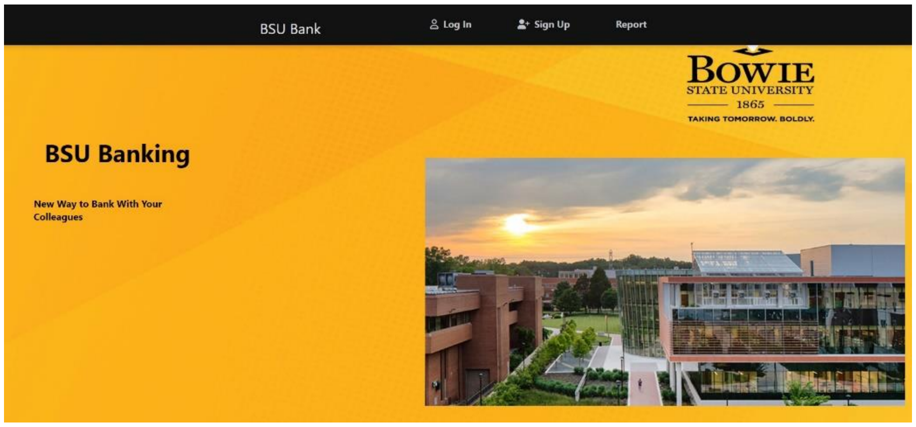
```
 1  const Sequelize = require("sequelize");
 2
 3  const sequelize = new Sequelize(
 4      'heroku_2a38526b70ff8a4', //'bankproject',
 5      'baa6318898e6a2', //'root',
 6      'a559d61a', //'',
 7      {
 8        host: 'us-cdbr-east-06.cleardb.net', //'localhost',
 9        dialect: 'mysql',
10        createDatabaseIfNotExist: true
11      }
12    );
13
14  module.exports = {
15      sequelize
16  };
```

You must reconfigure lines 4, 5, 6, and 8 depending on how you've set up your MySQL Workbench. For example, if the database name is 'banking', you'll need to replace 'heroku_2a38526b70ff8a4' with 'banking'. To replace the username and password, you need to change 'baa6318898e6a2' and 'a559d61a' respectively. Additionally, you need to replace the hostname that fits the database.



```
kalek@DESKTOP-M2OF8HC MINGW64 ~/Desktop/kalekidandevin/COSC480-Bank (master)
$ node index.js
Server started on port 3000
Executing (default): SELECT 1+1 AS result
Executing (default): SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABL
E_TYPE = 'BASE TABLE' AND TABLE_NAME = 'users' AND TABLE_SCHEMA = 'bankproject'
Database connected...
Executing (default): SHOW INDEX FROM `users`
Models synced...
```

After saving the changes and ensuring the database connection is secure, run "node index.js" to launch the website. It will be accessible locally at "localhost:3000".



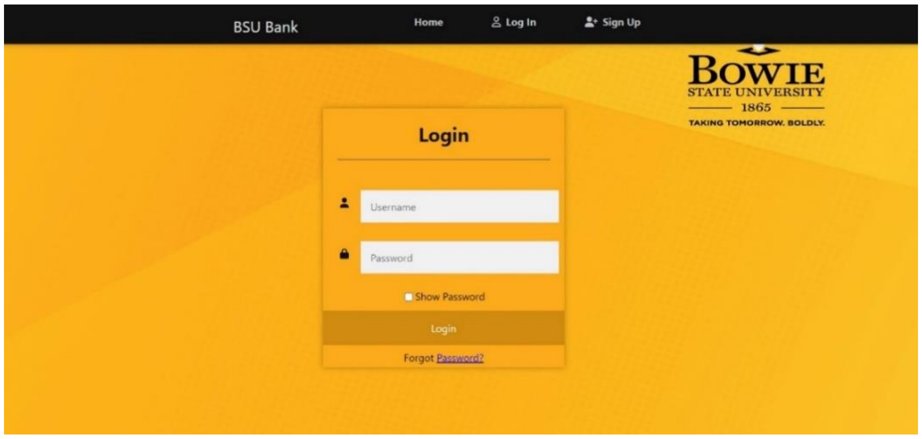
If you want to run your application on a website that is hosted, you can utilize Heroku. The transition will be smooth because Heroku can connect to the GitHub repository where your application is stored. After connecting ClearDB, you just need to push the cloned application into Heroku's master repository to officially launch the website. Once the application is pushed, you'll receive a website link to access it. However, if you are using Windows, you'll need to take additional steps and install an extra application (Heroku CLI). This may cause some issues.

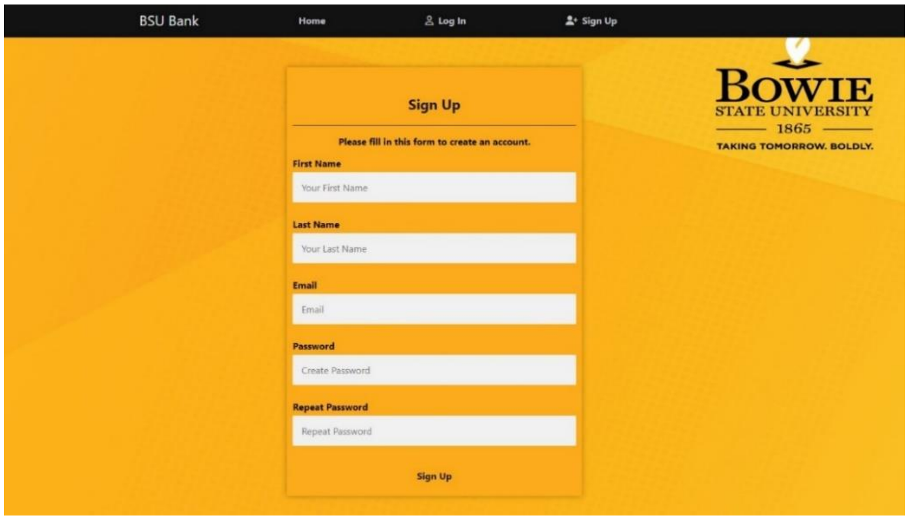
Conclusion

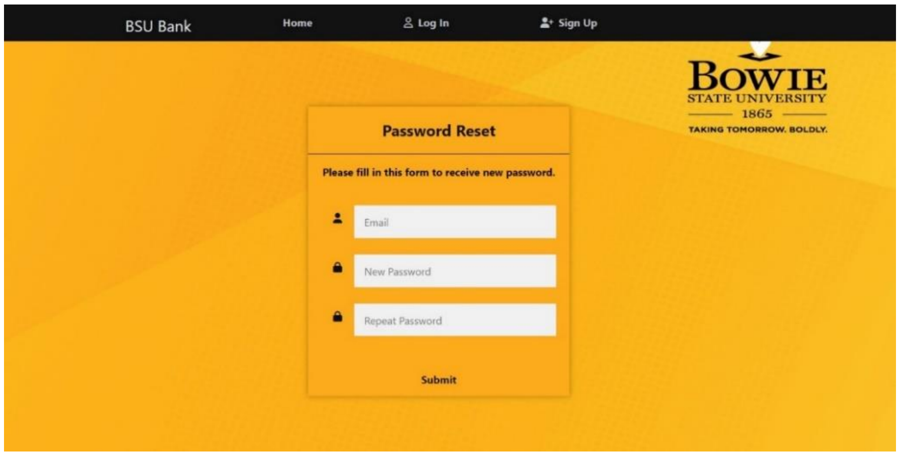
In conclusion, our collaborative efforts have resulted in the successful creation of a basic banking application using NodeJS, MySQL, and Sequelize. Throughout the development process, we encountered challenges as we transitioned platforms from PHP to NodeJS. This shift required us to adapt our coding 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.

Images from the Application

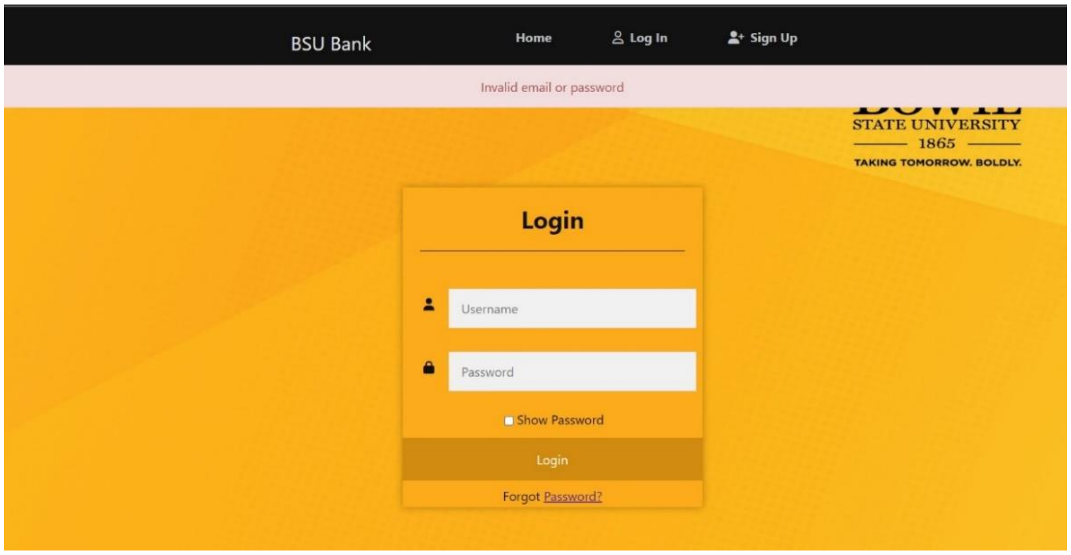
These images are the forms that the app offers for signing in, signing up, or resetting the user's password.

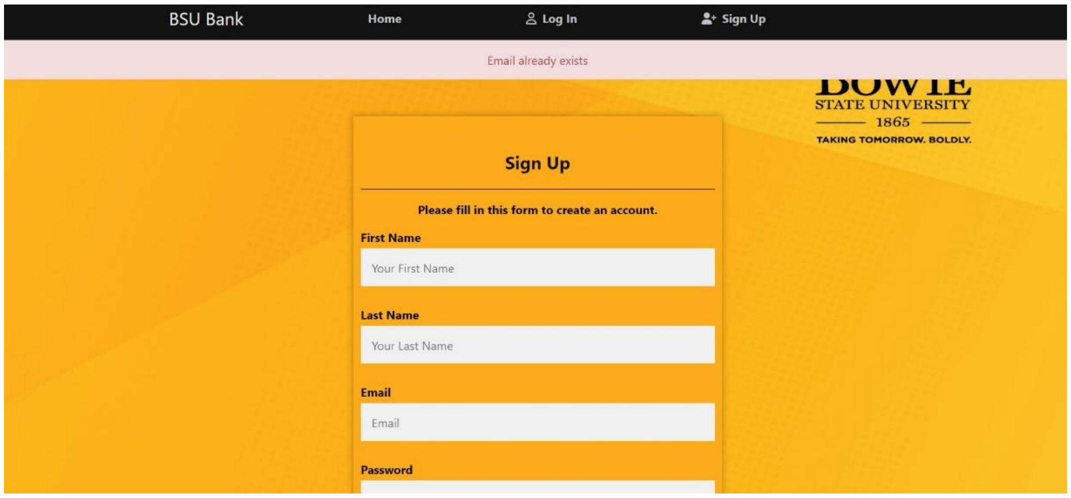


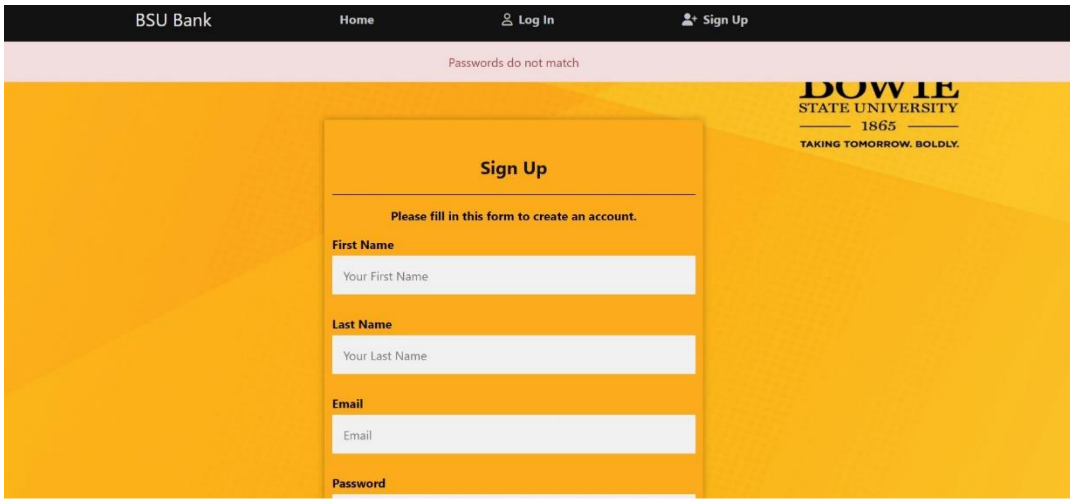



BSU Bank
Home
Log In
Sign Up
BOWIE STATE UNIVERSITY 1865 TAKING TOMORROW. BOLDLY.
Password Reset
Please fill in this form to receive new password.
Email
New Password
Repeat Password
Submit

The following pictures demonstrate how the application manages errors related to invalid email or passwords, existing and non-existing accounts, and mismatched passwords.






BSU Bank Home Log In Sign Up

Passwords do not match

BOWIE STATE UNIVERSITY 1865 TAKING TOMORROW. BOLDLY.

Sign Up

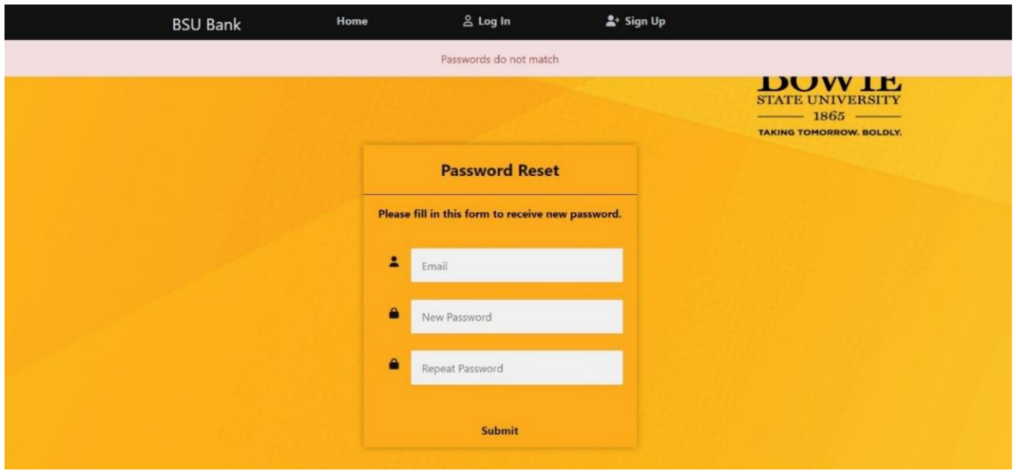
Please fill in this form to create an account.

First Name: Your First Name

Last Name: Your Last Name

Email: Email

Password


BSU Bank Home Log In Sign Up

Passwords do not match

BOWIE STATE UNIVERSITY 1865 TAKING TOMORROW. BOLDLY.

Password Reset

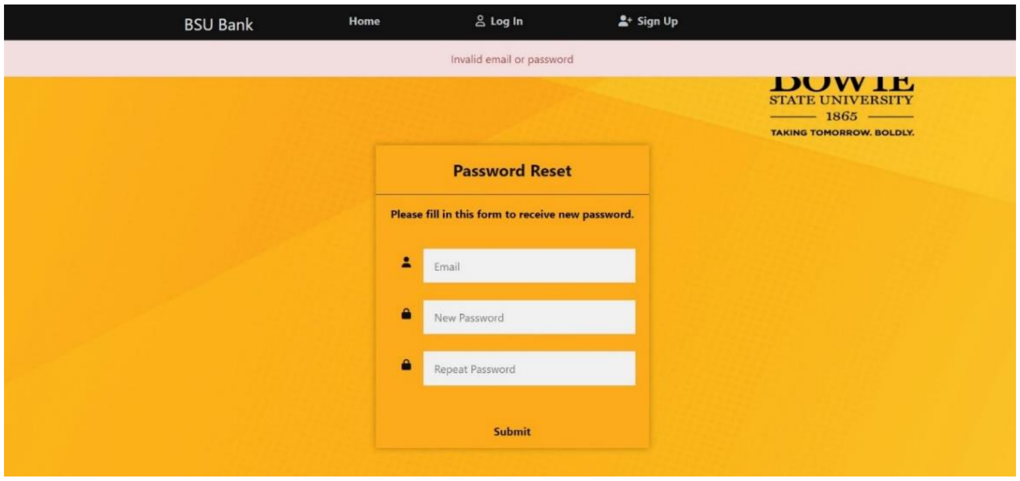
Please fill in this form to receive new password.

Email

New Password

Repeat Password

Submit


BSU Bank Home Log In Sign Up

Invalid email or password

BOWIE STATE UNIVERSITY 1865 TAKING TOMORROW. BOLDLY.

Password Reset

Please fill in this form to receive new password.

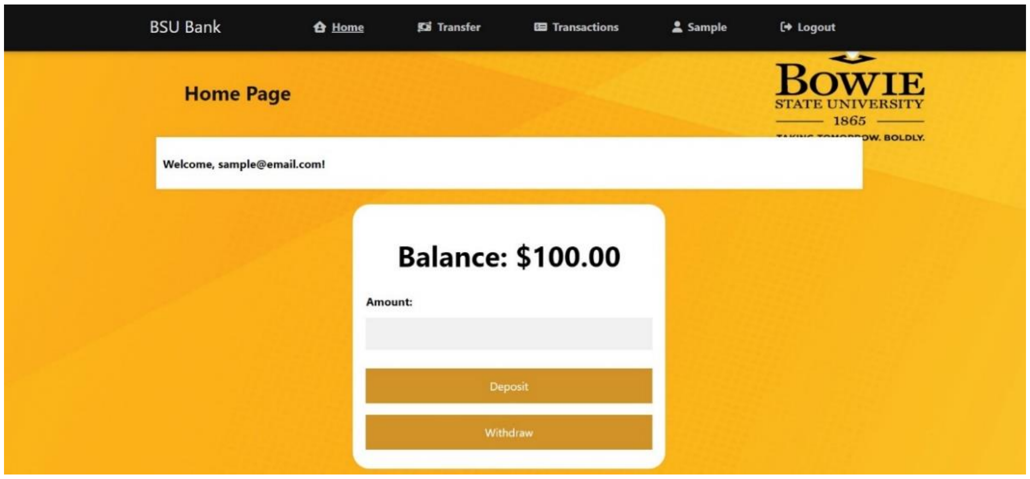
Email

New Password

Repeat Password

Submit

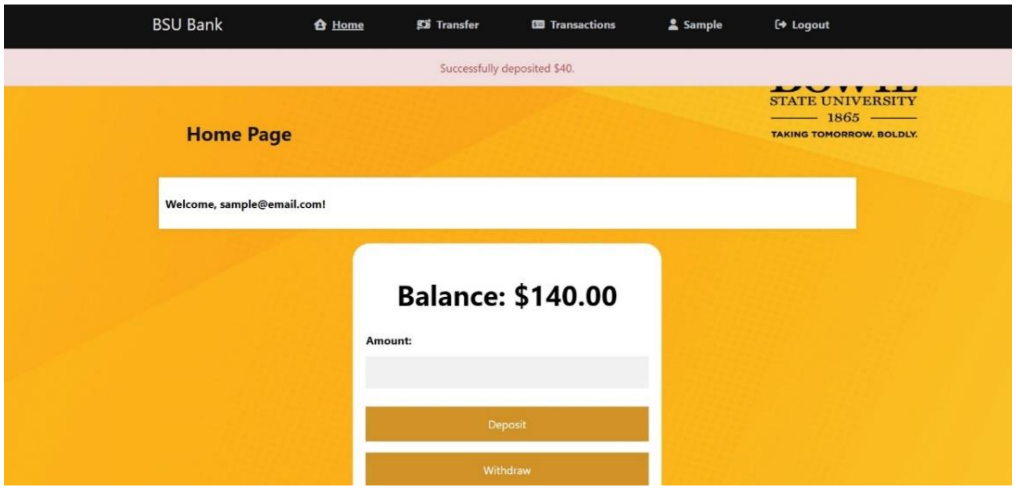
After logging in, the user will see their email address along with a default balance of $100. If the user wishes to deposit money, a message will appear indicating the amount that has been added to their balance. The same process applies to withdrawing money from the balance.



Welcome, sample@email.com!

Balance: $100.00

Amount:

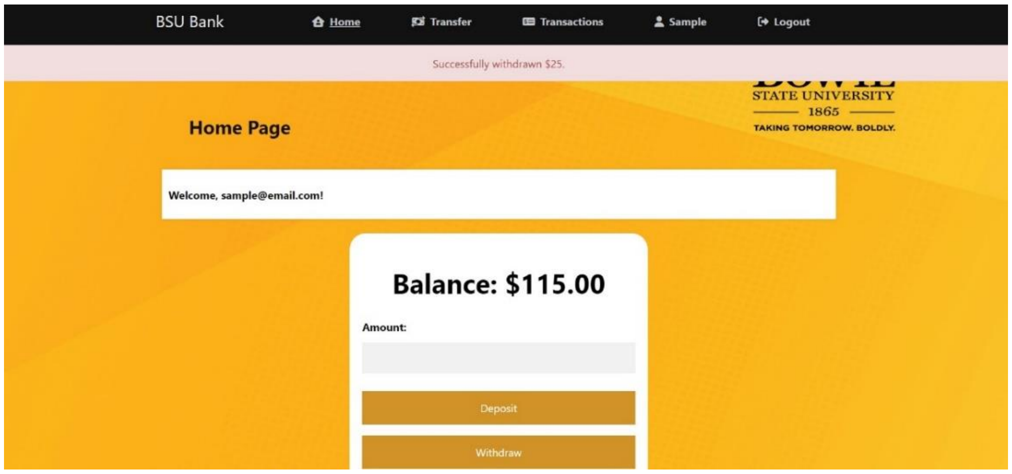


Successfully deposited $40.

Welcome, sample@email.com!

Balance: $140.00

Amount:



Successfully withdrawn $25.

Welcome, sample@email.com!

Balance: $115.00

Amount: