

**Bank of Insecurities**

**Table of Contents**

[**Project Description**](#h.gjdgxs)**…………………………………………………………………………………………………….……...….0**

[**Technology Stack Used**](#h.30j0zll)**……………………………………………………………………………………………………...….0**

[**Functionality**](#h.1fob9te)

[**1)** **Login Page**](#h.3znysh7)**………………………………………………………………………………………………………………………………1**

**2) Forgot Password Page……………………………………………………………………………………….……………………2**

**3) Editable Profile Page………………………………………………………………………………………………………..…….4**

**4) Admin Page - General……………………………………………………………………………………………………………..5**

**5) Admin Page – Password Editor……………………………………………………………………………………………....6**

**6) Admin Page – Inbox………………………………………………………………………………………………………………..7**

**7) Contact Page…………………………………………………………………………………………………………………………..8**

**8) Account Summary Page………………………………………………………………………………………………………….9**

**9) Payment History Page………………………………………………………………………………………………………..…11**

**10) Statement Page…………………………………………………………………………………………………………………….12**

# **Project Description**

* Bank of Insecurities is a web-based banking application which has vulnerabilities forcefully injected in it. It is named Bank of Insecurities because it is an intentionally insecure banking application.

# **Technology Stack Used**

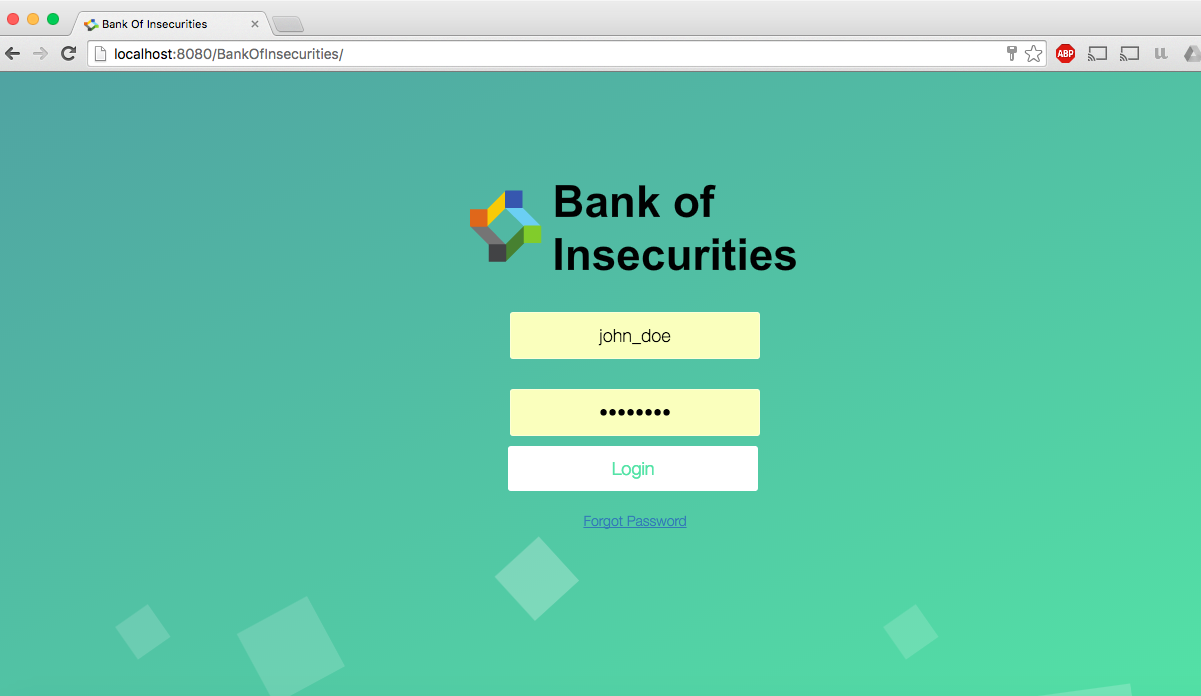
* Front End – Html5, CSS, JavaScript
* Middleware – Java
* Framework – MVC (Spring), Hibernate
* Backend – PostgreSQL

# 

# **Functionality**

## Login Page

Screenshot:



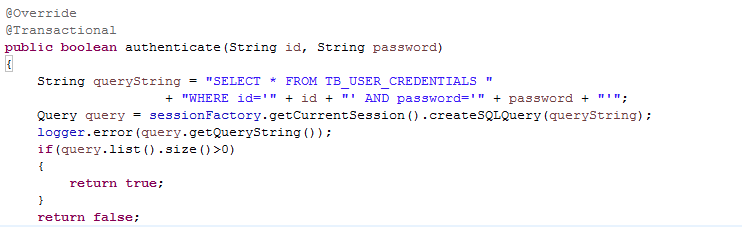
**Figure 1.1**

Features

* Enable User Login with username and password.
* Forgot Password link.

Vulnerabilities

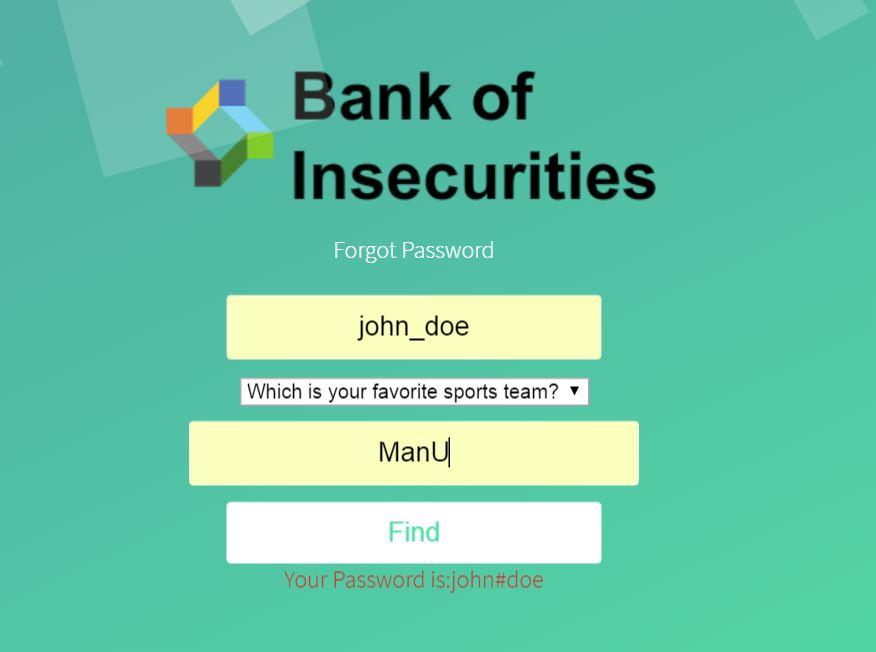
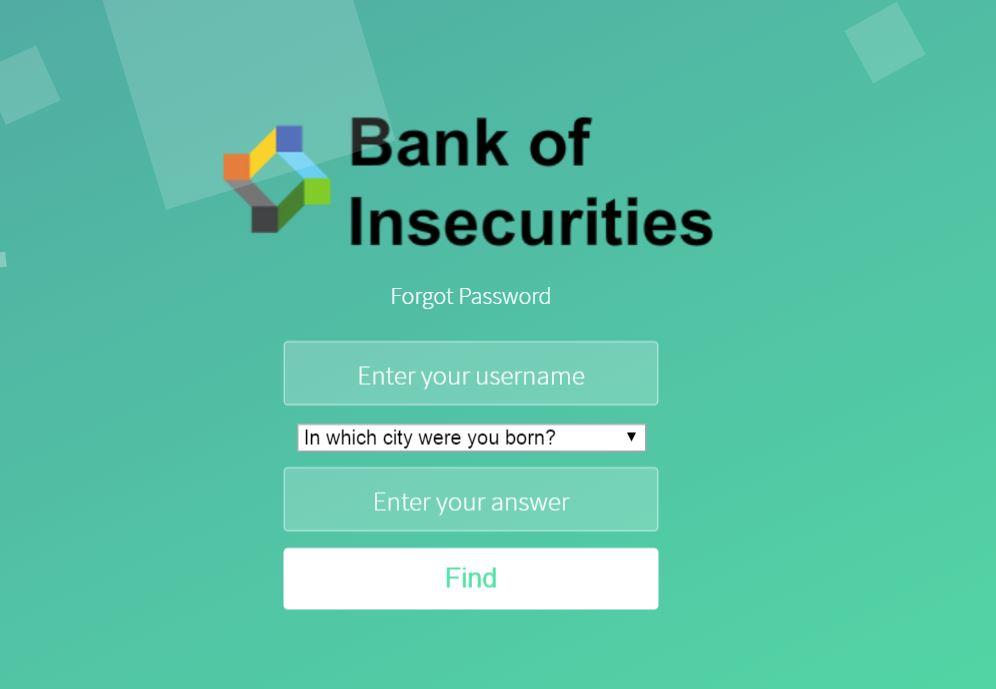
* SQL Injection (A1 - OWASP)
  + The queryString in Figure 1.2 will directly take the id inserted by the user. The user could easily insert something that will always be true to log into the first user’s account.
* On logout, users are redirected to a logout page. The logout page just takes a parameter of where to redirect the user to.
  + Session fixation can be done on the login page. Send a link with the modelParam “?sessionID=xyz” appended to the URL.
  + Video Demo: <https://youtu.be/7LvwmLhG8zY>



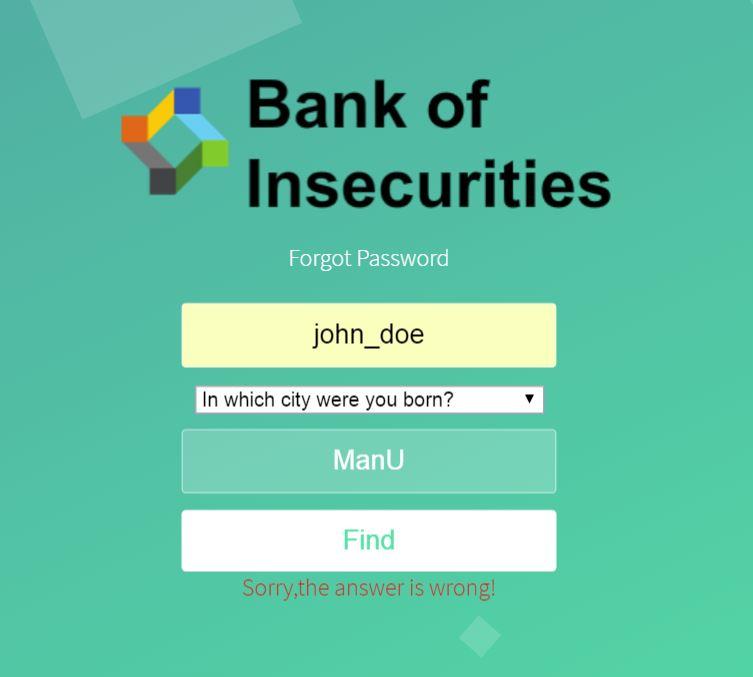
**Figure 1.2**

## Forgot Password Page

Screenshot:



**Figure 2.1 Figure 2.2**



**Figure 2.3**

Features

* Display security questions for users to retrieve password (Figure 2.1).
* Display the user’s password when answers are correct (Figure 2.2).
* Return error message when answers are incorrect (Figure 2.3).

Vulnerabilities

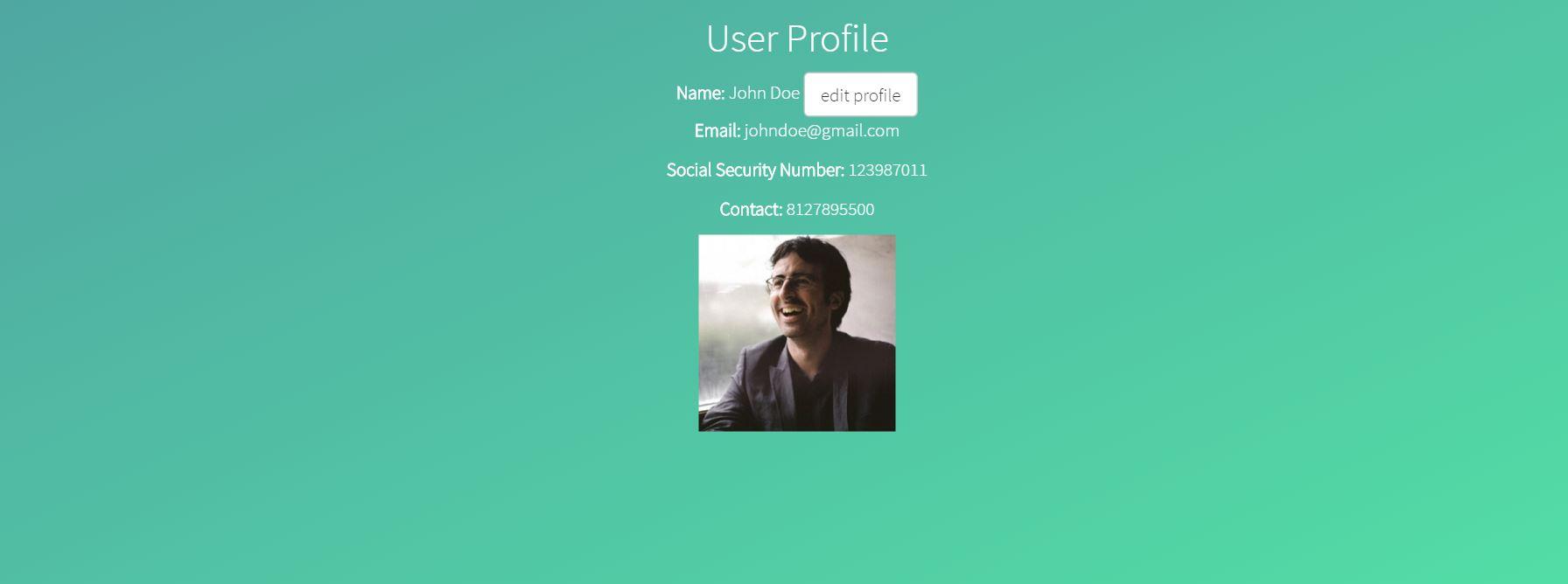
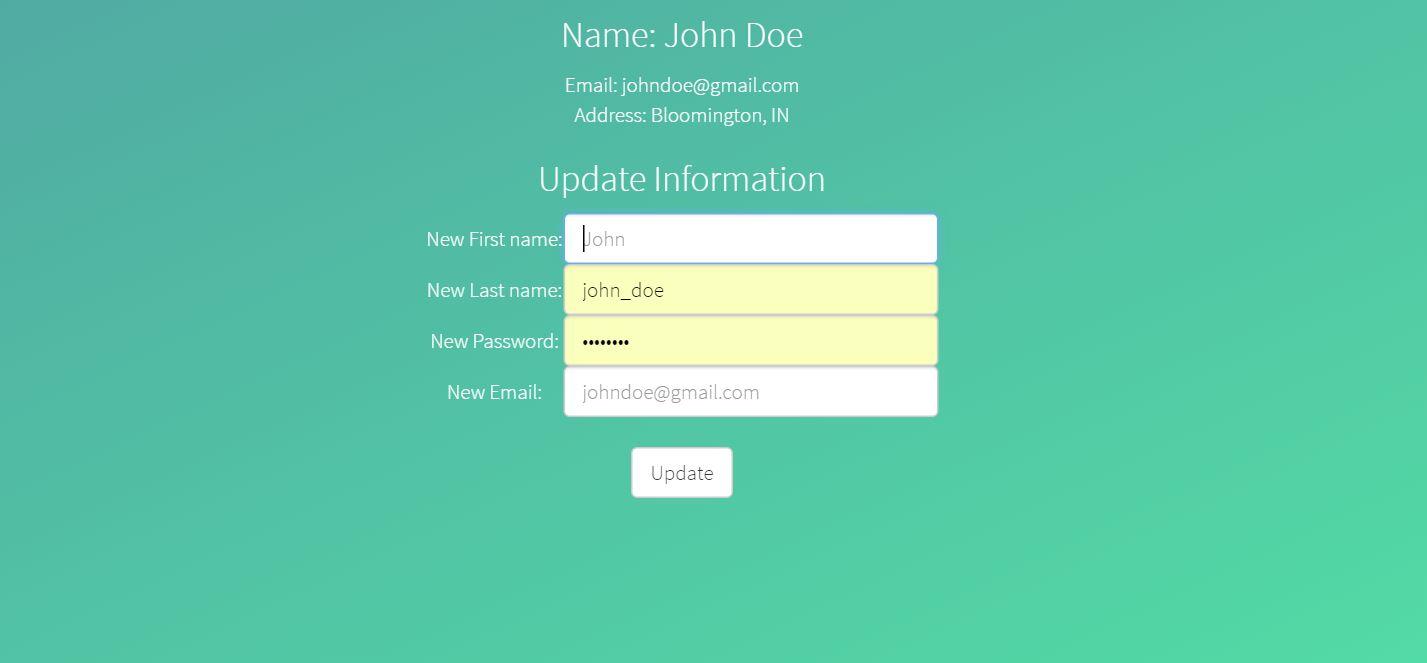
* Web Browser will store answers to security questions if not closed.
* Infinite attempts are allowed.

Relevant Classes

* /BankOfInsecurities/src/com/controller/ForgotPasswordController.java
  + This is where the controller logic is implemented.

## Editable Profile Page

Screenshot:

**Figure 3.1 Figure 3.2**

Features

* Show user profile - Figure 3.1
* Enable change credential functionality - Figure 3.2
* Click ‘edit Profile’ in Figure 3.1 to access the editing page on Figure 3.2.

Vulnerabilities

* Sensitive Data Exposure (A6 - OWASP)
  + The starred out password shown in Figure 3.2 next to ‘New Password’, is using the password text type, so all the characters and letters are replaced with “•”; however, if user choose to view the source code on the web page, then the plaintext password will be shown.
* Cross-Site Scripting
  + You can modify any of the fields to be a script.

Relevant Classes

* /BankOfInsecurities/src/com/controller/SingleProfileController.java
* This is where the controller logic is implemented.

## Admin Page - General

Screenshot:



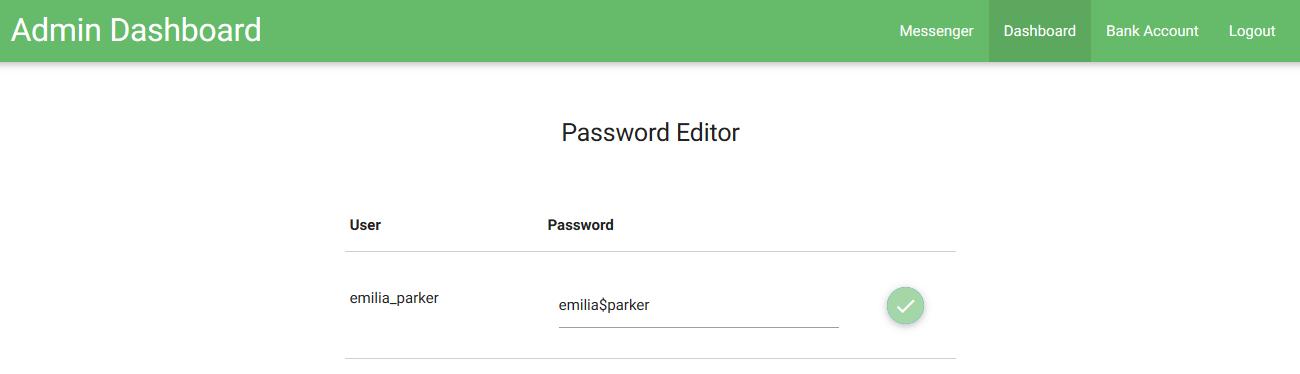
**Figure 4.1**



**Figure 4.2**



**Figure 4.3**



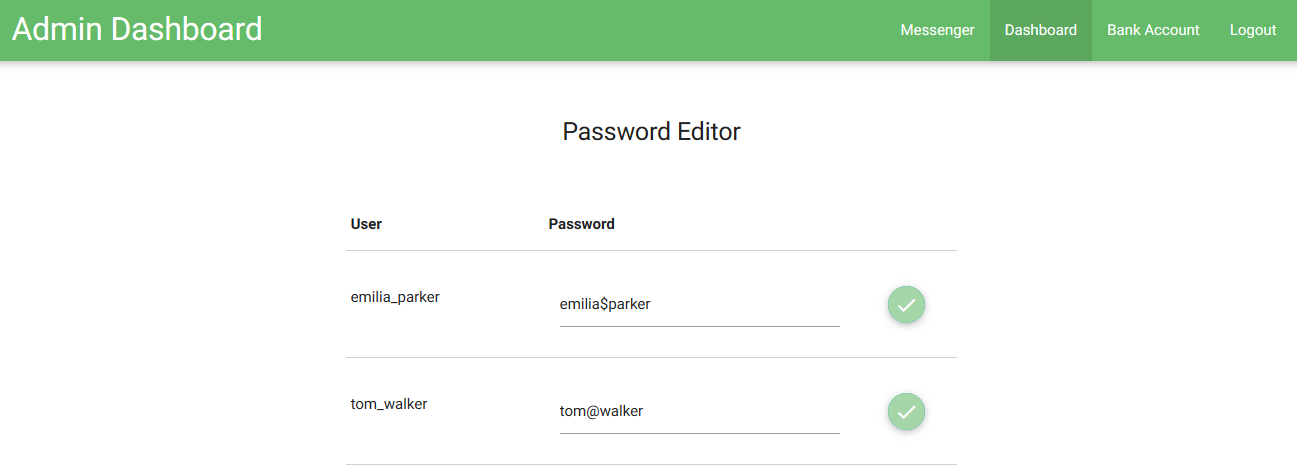
**Figure 4.4**

Features

* For a user to access the admin dashboard, he or she must be an admin in the database. His privileges are determined in the ‘roleid’ field of the ‘tb\_user\_details’ table pictured in Figure 4.3. ‘N’ means that the user is normal, ‘A’ indicates an admin.
* In order to access the dashboard, click the ‘Admin’ button that will be displayed on every page if the user is an admin. The button is pictured in Figure 4.1. Figure 4.2 is what a non-admin user will see in the navigation toolbar.
* If you reach the admin dashboard, it should look like what is pictured in Figure 4.4.

## Admin Page - Password Editor

Screenshot:



**Figure 5.1**

Features

* Edit any non-admin user’s password by typing the new password in the corresponding text field in the ‘Password’ column and clicking the green checkmark in Figure 4.3.

Vulnerabilities

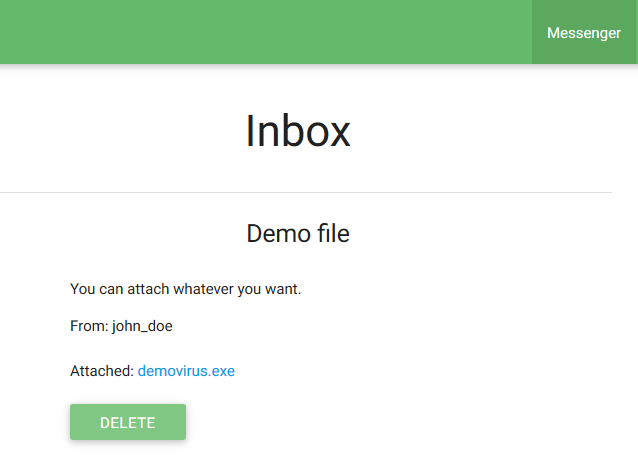
* The malicious transferring is toggled on and off on this page. Malicious transferring causes any transaction that any user has to transfer money to the admin john\_doe’s account. This means if a transaction is for $8.2112, $.0012 is transferred into john\_doe’s account. This functionality is toggled on and off by appending the following: “?funTransferring=<true/false>” to the end of the URL.

Relevant Classes

* /BankOfInsecurities/src/com/controller/AdminDashboardController.java
  + This is where the controller logic is implemented.
* /BankOfInsecurities/src/com/command/TransferFundsCommand.java
  + The malicious transfer is done around lines 72-83.

## Admin Page - Admin Inbox

Screenshot:



**Figure 6.1**

Features

* View or delete the messages that a user sent to the admin
  + Send these messages using the ‘Contact’ page in the regular banking application.
  + The delete button removes the message from the database and the corresponding attachment from the server.
* Click the hyperlink next to ‘Attached:’ in Figure 6.1 to download the attachment that was saved to the server.

Vulnerabilities

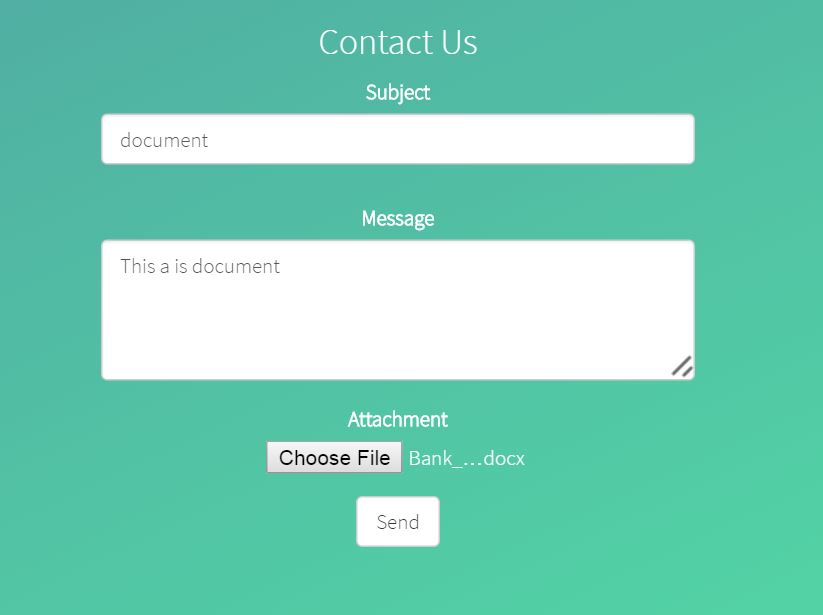
* The attached files can be anything. They are not verified at all. The files are also saved directly to the tomcat server that the application is deployed on.
* Cross-Site Scripting
  + The messages are not verified, anybody could insert a script into them.
* No limit on the amount of messages that can be sent or the time between messages.

Relevant Classes

* /BankOfInsecurities/src/com/controller/AdminMessengerController.java
  + This is where the controller logic is implemented.

## Contact Page

Screenshot:



**Figure 7.1**

Features

* Users are able to send messages with a subject and attach one file the administrator. The messages and the file attached will be received on the Admin page.
* Press ‘Choose File’ to attach file and press ‘Send’ to send the message.

Vulnerabilities

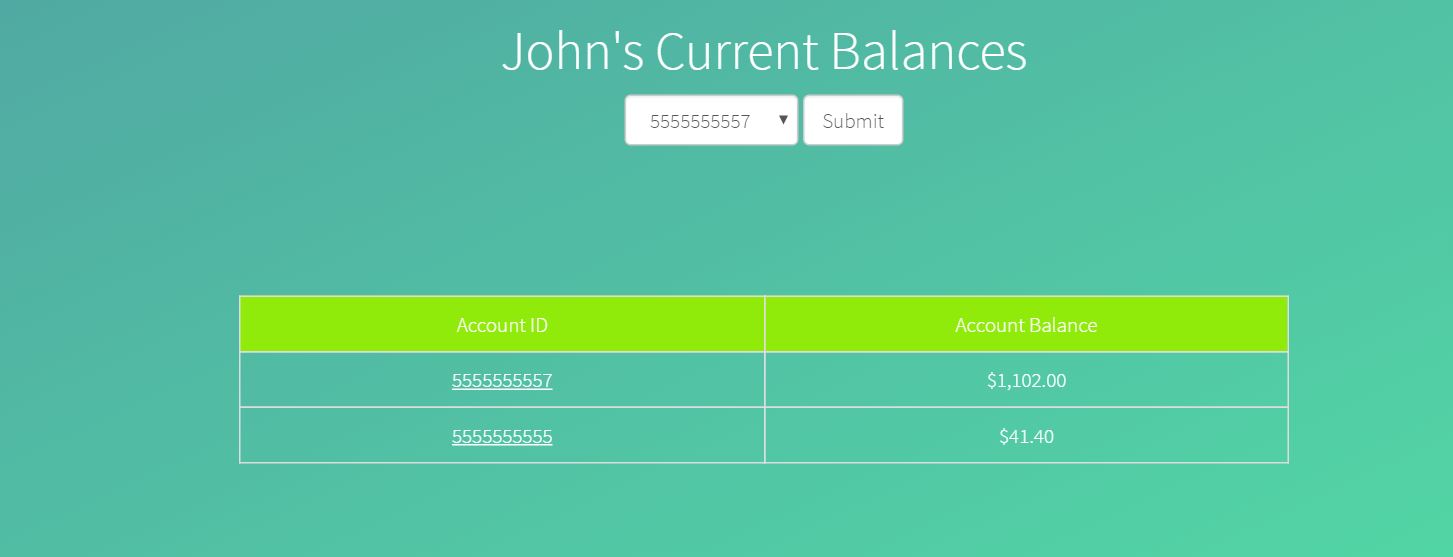
* The file can be anything, there is no check on the file attached.
* Users can only upload one file at a time and no notification whether the message delivered or not.

Relevant Classes

* /BankOfInsecurities/src/com/controller/MessagePageController.java
* /BankOfInsecurities/src/com/controller/model/MessageToAdminModel.java

## Account Summary Page

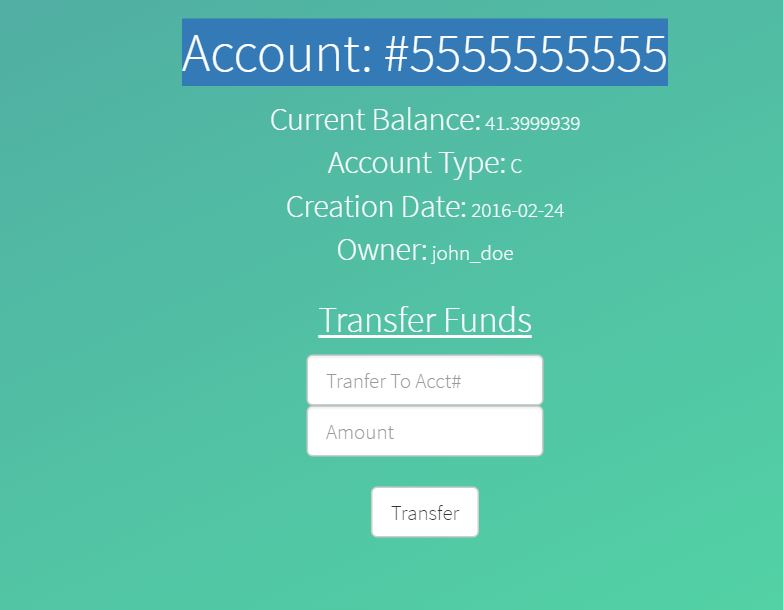
Screenshot:



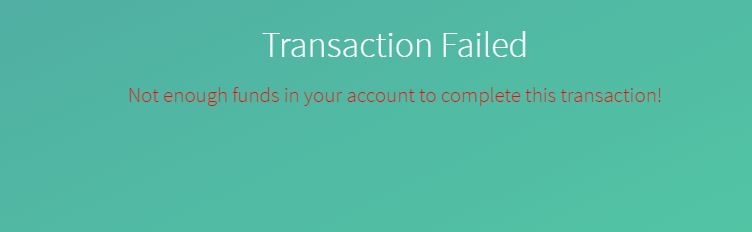
**Figure 8.1**

****

**Figure 8.2A**

****

**Figure 8.2B**

** **

**Figure 8.3A Figure 8.3B**

Features

* The typical account management page is shown in Figure 8.1 displaying the current balances of the user’s accounts. The user will have two types of accounts (saving and checking).
* Users have two ways to view the details of each account. First, the user can select account number in the drop-down menu in Figure 8.1, the other way is to click the hyperlink in the bottom of the page.
* In each account detail page (Figure 8.2A), the details include account number, current balance, account type, creation date, and name of owner.
* On checking accounts, there is a “Transfer” button, which enable user to transfer money from the current account to any other account. Transfers are not allowed on a saving accounts. An invalid transfer is shown in Figure 8.3B.

Vulnerabilities

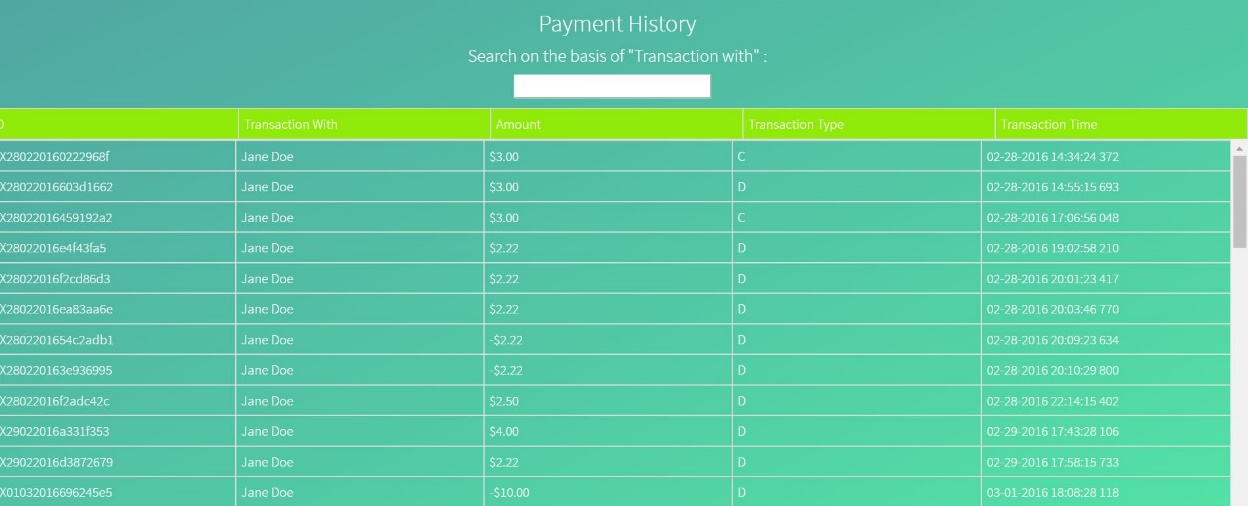
* The money transferred can be a negative number, which violates basic business logic. In this project, it enables the user to maliciously steal money from another account.

Relevant Classes

* /BankOfInsecurities/src/com/controller/AccountManagementController.java
  + This is where the controller logic is implemented for the account summary
* /BankOfInsecurities/src/com/controller/model/SingleAccountController.java
* Details of a specific account’s controller Figure 8.2A.

## Payment History Page

Screenshot:



**Figure 9.1**

Features

* This page shows detailed payment history of a user, the content includes user name money was transferred to/from, the amount of money transferred, transaction type and the time transaction happened.
* There is a search field in this page which enables the user to insert character or number to better retrieve a specific one among all the transactions.

Vulnerabilities

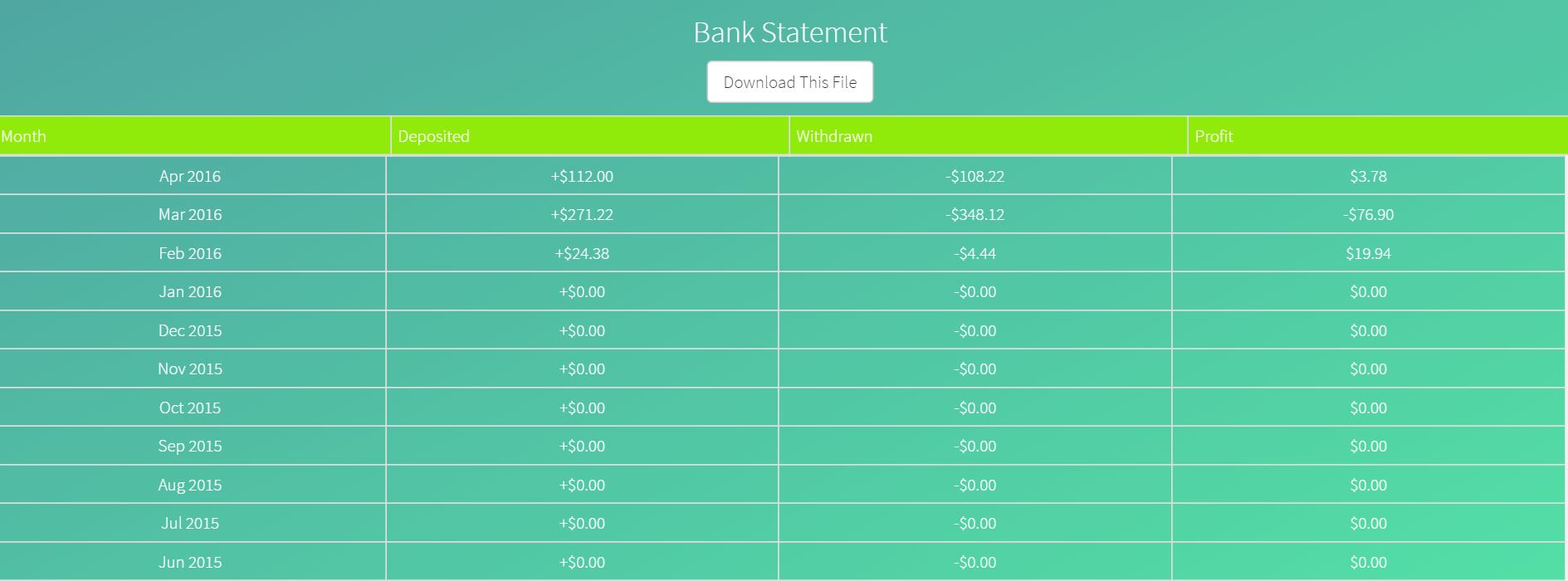
* DOM-based Cross-site Scripting is a vulnerability in this page’s text box.
* The user can go to accountManagement/paymentHistory/admin to any user’s payments.
  + Video Demo: <https://youtu.be/HqQG44pbciE>

Relevant Classes

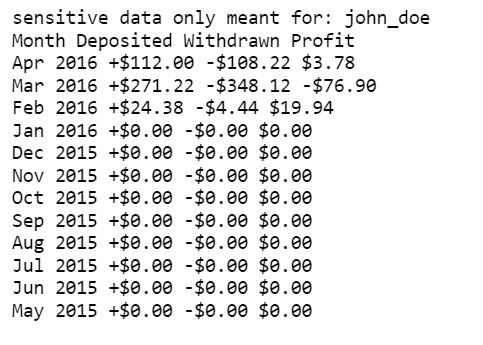
* /BankOfInsecurities/src/com/controller/TransactionHistoryController.java
* This is where the controller logic is implemented for the Payment History Page.

## Statement Page

Screenshot:

****

**Figure 10.1**

****

**Figure 10.2**

Features

* This Page shows a monthly statement for a certain user’s account. The detail includes the month, the total amount of the deposited money, the total amount of the withdrawn money, and the profit of the account.
* The statement is downloadable. Press the ‘Download This File’, a txt version of statement will pop up as shown in Figure 10.2.

Vulnerabilities

* While log in, a user can edit the URL to retrieve other users’ statement to retrieve their statement.
  + EX: www.Base\_URL/statement/<username>/downloads/statement.txt
  + Video Demo: <https://youtu.be/w8WlOXntIZg>

Relevant Classes

* /BankOfInsecurities/src/com/controller/StatementController.java
* This is where the controller logic is implemented for the Statement Page.