SouthWest Bank: A Secure Online Banking

CSE 545: Software Security, Final Project Report – Group 15

Aneesh Umesh Shastry
ASU ID: 1207047955
MCS CS, CIDSE
Ira A. Fulton Schools of Engineering
Arizona State University
ashastry@asu.edu

Mahathi Shakthidharan
ASU ID: 1207066610
MCS CS, CIDSE
Ira A. Fulton Schools of Engineering
Arizona State University
mshakthi@asu.edu

Rajesh Surana
ASU ID: 1207633202
MCS CS, CIDSE
Ira A. Fulton Schools of Engineering
Arizona State University
rajesh.surana@asu.edu

Shankar Krishnamurthy
ASU ID: 1208465371
MCS CS, CIDSE
Ira A. Fulton Schools of Engineering
Arizona State University
shankar.krishnamurthy@asu.edu

Sagar Mukesh Sangani
ASU ID: 1205141934
MS CE, CIDSE
Ira A. Fulton Schools of Engineering
Arizona State University
smsangan@asu.edu

Tanvi Apte
ASU ID: 1207666430
MCS CS, CIDSE
Ira A. Fulton Schools of Engineering
Arizona State University
tapte@asu.edu

Varun Kamath Burde
ASU ID: 1207688062
MCS CS, CIDSE
Ira A. Fulton Schools of Engineering
Arizona State University
vkamathb@asu.edu

Karthik Narayanan Sockalingam
ASU ID: 1207626559
MCS CS, CIDSE
Ira A. Fulton Schools of Engineering
Arizona State University
ksockali@asu.edu

***Abstract*— This document is the final project group report for the Secure Online banking project. It contains the design for a secure banking application. Complete details on the functional and non-functional features are provided. All security features implemented for the application security has been discussed in detail. The roles and responsibilities of each of the team members listed above has been discussed. In the end, a discussion on the vulnerabilities reported by testers have been discussed.**

I. INTRODUCTION

The SouthWest Bank application is a secure online banking system (SBS) that facilitates the working of any common bank. The need of automating everything and making it available over the internet for 24 hours has led to constructing systems like this one. It has all features of a normal bank. It is demonstrated for 5 types of users, Internal Employees, Managers, Customers, Banking Administrator and Merchants. The application allows external user to login and perform tasks as he would do in normal world banking application. Users can access their statements, transaction history, transfer funds to another user and debit or credit funds in their account and between their accounts. Opening a secondary bank account online and managing it is made easy using this application. The system has made it easy even for the bank employees to manage accounts of user. The government employee is also involved in the system as it's required to access and release person's personal information. Banking administrator is responsible for creating and managing all important accounts like regular employee, bank manager and external user and to manage the system logs.

The system is made highly secure by implementing different security features. As, the system deals with people's personal information as well as their funds, security is the basic necessity in this case. A complete and comprehensive all round security has been implemented in the system. This paper will talk in detail about which security features are used and how they are implemented. Some of the implemented security features are PKI, password hashing, captcha and OTP. All the required user communication is done through E-mail. All keys required to implement PKI are also enclosed in an email and sent to the user. The system is successfully implemented over the network and is functional.

II. SYSTEM DESIGN

A. Product perspective

The Secure Online Banking system has been designed to allow customers – internal and external users, of a banking organization to perform transactions and bank operations online in a secured manner. Security to such an application is of utmost importance and any defect in the software can lead to huge financial losses.

The internal users- who are the bank employees and the bank manager, of the application use the system to perform banking

and manage critical transactions of all user accounts while external users – the individual customers and the merchants use the system to initiate transfers, payments etc. from their individual accounts. The functionalities and features are explained in detail in the below sections.

B. Product Architecture

The system has a three tier architecture with a database layer. It can be represented as follows

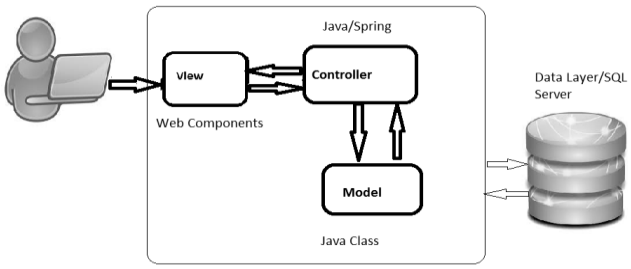


Fig. 1. *- Product Architecture*

Product Functionality/Features

1) User Login

- Here, the users are allowed to access the system features through a secure login mechanism.
- Authentication to first time login through a device is provided through OTP.
- The users are also provided with password recovery link.

2) External User Account Functionalities/Features

- Users can initiate a change in the account critical information through an approval mechanism wherein any change has to be approved by the bank employees before it is updated in the system.
- External Customers of the system can initiate transfers to other customers through the unique external customer identification ID.
- The customers can set preferences for alerts in various transactions in their accounts.
- The customers will also be provided with option to restrict the bank employees to view their information.
- External customers are also provided with option to initiate transfers to a list of registered merchants in the system.
- The External customers of the system are also provided with options to download statements.
- A special set of external users called the merchants will be able to accept the initiated payments by the individual customers and forward it to the bank to receive payments.

3) Bank User (Internal User) Functionalities/Features

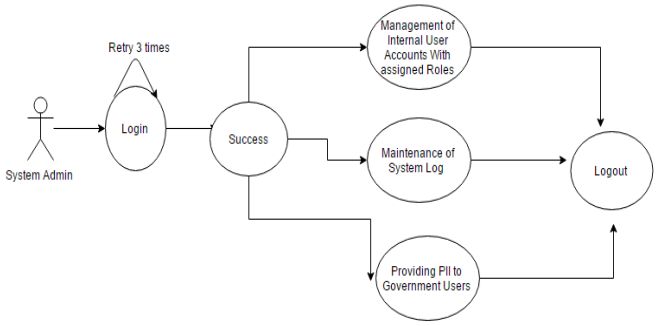
- The internal users of the system are classified into two groups – Bank employees and system admins.
- The regular employee will be creating the set of external user accounts and the manager will approve the creation of the accounts.
- The bank employee sub group are further categorized as regular bank employees and bank managers.
- The regular employees are provided permissions to review critical transactions by the manager and upon approval they can review the pending critical transaction pool and approve them to proceed.
- Upon approval from the individual users, the regular employees will be able to view, update and modify the individual user account information.
- The bank managers in addition to access the features and functionalities of the regular employees will have access to management of individual user accounts.
- The system admins have access to the various system resources and their functionality is to manage the internal (Bank employee) accounts. The management of the internal user accounts include creation, modification and deletion of the internal user accounts.
- The system admins can also access the encrypted PII information of the employees and forward it to the government employees on request.

C. User Characteristics

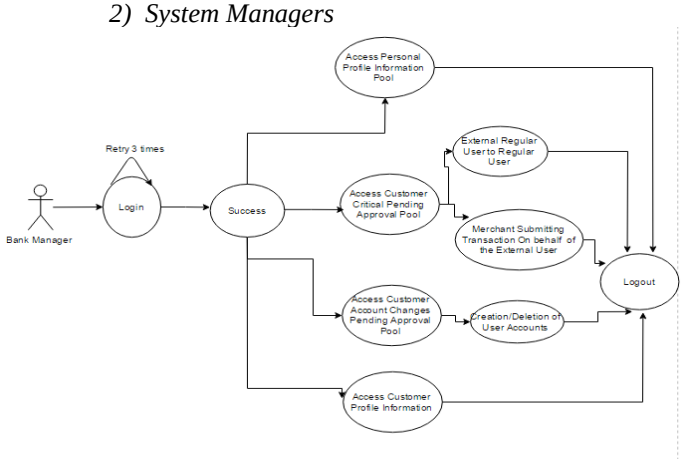
The users of the system can be broadly classified into two categories – Internal Users and External users

a) Internal Users – This group of users are further classified into three roles. The internal users of the system will be allowed to access different functionalities and features of the system as opposed to the external users. Here is a brief look of the various features that each category of internal users can access

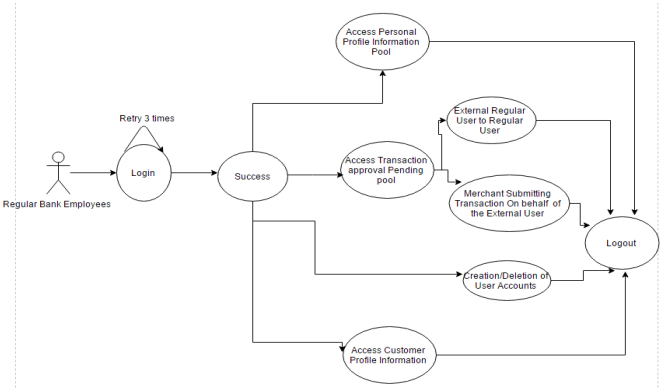
1) System Administrators



2) System Managers

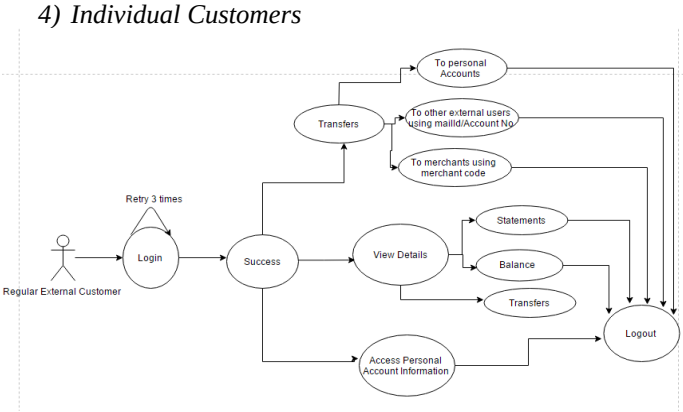


3) Regular Employees

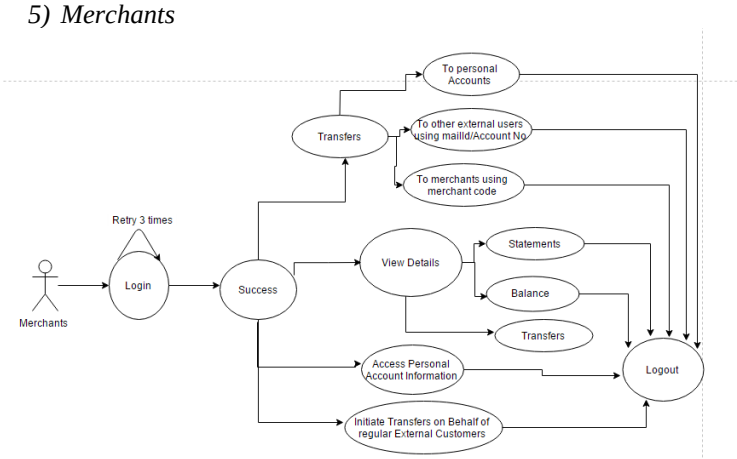


b) External Users – This group is further classified into two roles

4) Individual Customers



5) Merchants



III. Team Member Roles and Responsibilities

1. ***Mahathi Shakthidharan****:* Mahathi was responsible for the SRS preparation during the Design phase. Later on during the implementation phase, she worked hard on implementing all the vital components of the system. Setup of the base system and the login functionalities was first completed by Mahathi and the system was built around it. The other important features implemented were Internal Employee functionalities, Manager Functionalities, UI design and setup, Admin User functionalities, Government user functionalities and Controller level validations. Significant involvement in the implementation of PKI functionality. Mahathi was actively involved in the Application testing and User guide updates.

2. ***Sagar Sangani****:* Sagar was responsible for the preparation of database schema during the design phase of the project. He played a pivotal role in the rapid and reliable setup of Hibernate on the database earlier on in the project so that the rest of the application could be built around it. The other functionalities that were implemented by Sagar are as follows. Internal Employee functionalities, Manager Functionalities, Internal - External user workflow, Admin User functionalities, Controller level validations, PKI functionality, User guide updates. Sagar was involved in the Application testing and played an important role in the Server code deployment on the Virtual machine.

3. ***Rajesh Surana****:* Rajesh played an important role during the SRS preparation during design phase. With the maximum number of commits to the codebase, he played an instrumental role during implementation. He successfully worked on implementing the secure login functionalities. Worked on the Captcha and the system lockout functionalities by implementing novel lockout functionalities. Other features implemented were Front end validations, External User functionalities, Merchant functionalities, External - Internal user workflow, Application testing, Controller level validations, Virtual

keyboard at login page, Captcha for login page, Role based authentication for each page, PDF generation for external user bank statement, forget password functionality. Also implemented all the essential front end validations.

4. ***Shankar Krishnamurthy**:* Shankar worked on the SRS during the design phase of the project. He explored different ways by which the whole system can be secured on a variety of attacks on the network. Shankar played an important role in the development of the system session management functionality. Various techniques using the system hardware and software context were used to devise the session management features which proved to be successful. The other functionalities implemented were External User functionalities, Merchant functionalities, External - Internal user workflow, Controller level validations. He was also involved in the application testing.

5. ***Varun Kamath Burde**:* Varun played a very important role during the system design phase by working on producing a well-defined and detailed User guide. He was responsible in the design of UI layouts and the different user interfaces required for each of the different user roles in the application. He also worked on the implementation of other features like Bootstrap setup, Front end validation, Controller level validations. Worked on the implementation of the PKI functionality, updating UI pages with the required data and page elements. He played a vital role in the preparation of updated user guide and during the testing of the application.

6. ***Tanvi Apte**:* Tanvi was responsible for the UML and class diagram preparation during the project design phase. In the implementation phase, she was responsible to implement the Bootstrap elements on the UI of the application. By applying bootstrap to all the .jsp pages, the responsive web design was implemented on the application view. Also worked on the view design for the different pages and user roles in the application. Also worked on the controller level validations during the implementation phase. Played an important role in updating the user guide.

7. ***Karthik Sockalingam**:* Karthik worked on the test case preparation during the design phase of the project. Test cases covering all the important aspects of the application functionality were designed to ensure completeness. All the requirements were taken care of during the test design. In the implementation phase, Karthik assisted in the development of the Government user functionalities which could be used to approve the PII functionalities for the system admin. He was involved in the application testing and while updating the user guide.

8. ***Aneesh Shastry**:* Aneesh was involved in preparing the Test Plan and Test cases during the design phase. Detailed Test plan was prepared with multiple rounds of testing spanning both functional and non-functional requirements. Was involved in the UI and interface design of the application. He was responsible for the setup of SSL certificates on the server and on the virtual machines. Responsible for implementing the exception handling in the application with the controller advice and the setup of the generic error page in the application. Also worked on the controller level validation for different functionalities. Was actively involved in the testing of the application. In addition to this, Aneesh was also responsible for the project lead activities.

IV. Implementation

A. External Interface Implementation

External interface of the SBS is in the form of web pages opened in browser. The front page has a login window which requires email as username, password and captcha to log into account. Unauthorized user won't be able to pass beyond this page. After logging in each user will have user interface based on his role.

1. *Login View*
   - Username text field
   - Password text field [pops up virtual keyboard on clicking inside.
   - Login button
   - Forget password button

2. *Regular Employee*
   - List of pending transaction pool in the form of table
   - View and edit external user and merchant's account information. Some of the fields will be editable based on the external user authorization
   - List of pending account update requests (from external user and merchants) in the form of table.
   - New bank account creation request option for an external user (with manager's approval)
   - Logout button

3. *System Manager*
   - List of pending transaction pool in the form of table
   - Filter button to filter out non-critical transactions
   - View and edit external user and merchant's account information. Some of the fields will be editable based on the external user authorization.
   - A list of pending external users' bank accounts to be created in the form of table
   - A search box to pull up any user and linked account
   - List of pending account update requests(from external user and merchants) in the form of table
   - Logout button

4. *System Administrator*
   - View and edit internal user account details in the form of table
   - List of system logs in the form of table
   - Search box to pull up internal users PII and ability to update it

- Logout button

5. *Individual Customer*
   - Retrieve and download Balance
   - Button to download statements in the specified date range
   - Transfer button which pull up empty transfer form
   - Transfer form to transfer money with send button
   - Account information with some of the field editable
   - List of changes done by bank employees to user's account in the form of table with approve or deny button for each change
   - Newsfeed section listing history of account changes
   - History of past transactions in the form of table
   - Limit field to set daily transaction upper cap
   - Logout button

6. *Merchant/Organization*
   - All of the interface items listed for individual user
   - Special transfer button to transfer amount to another merchant
   - List of transactions for received money with accept/decline buttons
   - Logout button

B. *Internal Interface Requirements*

System's internal interface will follow following architecture -

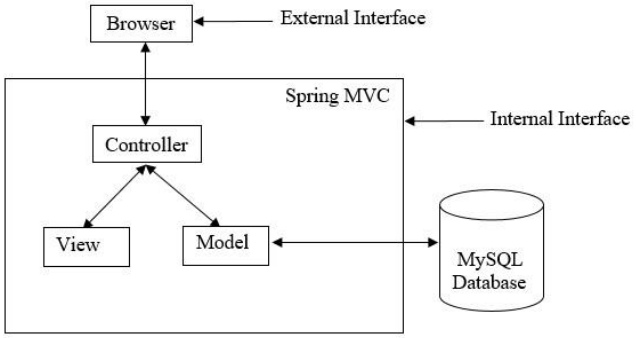


Fig: System Architecture diagram

1. *Internal Data Implementation*

All data in SBS is stored in the MySQL database. Data transfer over internet will be done securely using public key infrastructure and https protocol.

2. *Design and Implementation Constraints*
   - OTP is valid for 20 minutes only after generation.
   - External individual user will be able to do transaction up to a maximum of $10000 each day.
   - User can set limit for max transaction per day.
   - User should maintain a minimum balance of $2000.
   - After 3 unsuccessful attempts user won't be able to login for 1 hour.
   - Regular employee will not be able to view/edit external user's account information without proper authorization.
   - Only system admin can view and update internal users PII.

3. *Other Requirements*
   - This system software will be written in Java (Spring MVC).
   - Apache will be used for web hosting.
   - MySQL database will be used to store data.
   - Basic security features will be implemented using spring and some of them could be accomplished using third party product.

C. *Constraints*

The following are the constraints of the software application designed for the users of the system.

- Each user of the system will be able to perform critical transaction using the system only upon approval from the bank employees.
- The regular bank employees will be able to perform modify the user accounts only upon permissions from the individual customers.

D. *Assumptions and Dependencies*

- Implementation of certain functionalities like e-mailing is dependent on third party tools and any bug found with respect to these should be resolved by communicating with the owner of the tools.
- The developed application was deployed in the virtual machine provided by vlab.asu.edu and it was assumed that the machine is available at all times.

V. SECURITY FEATURE IMPLEMENTATION

1. *One Time Password*

One time password is randomly generated and used for each new transaction. As compared to static login password, it is dynamic and valid for very limited time. In our application, we use OTP for Debit/Credit transactions as well as forget password functionality. OTP is not replacement for static password but should be used in combination with it to add additional level of security.

2. *Virtual Keyboard*

We have used Keithwood's [1] library to implement virtual keyboard on login page. Virtual keyboard is a software interface for entering sensitive information such as password into the online form. It protects user's credentials from key logger installed on unsecure/public machines. In addition, we use virtual keyboard to enter OTP for various critical transactions.

3. *Secure Socket Layer*

Secure Socket Layer (SSL) is a standard security technology for establishing an encrypted link between server and client. In our implementation, we establish an http connection over SSL. This ensured that our entire transaction is encrypted and prevented man in the middle attacks.

4. Public Key Infrastructure

A public key infrastructure(PKI)[2] is a set of hardware, software, people, policies and procedures needed to create, manage, distribute, use and store and revoke digital certificates and manage public-key encryption.
In our implementation, we have used Rivest-Shamir-Adleman (RSA) [3] to generate private and public key pair. We have used email communication as form of reliable transport to provide these keys to the client. In order to make sure that server is talking to correct client, the server uses the private key to authenticate the client. Similarly, the client uses the public key to authenticate the server.

5. Captcha

Captcha [4] is an acronym for Completely Automated Public Turing test to tell Computer and Human Apart. We implemented Captcha by running a randomized algorithm which generated a random string each time and this string was converted into an image. We would then wait for user input to match the same string. If captcha entered was right along with the login details, the system would allow access to the user.

6. Role based Authentication/Access

Role based Access ensures that each user can access functionalities/web pages only according to his privilege level. We ensured that Role based access was maintain across our system and no user was able to navigate to a feature he wasn't authenticated for.

7. *Single Logging in one session*

This security feature ensured that from a single machine only a single logging was possible at a time.

8. Hashed Password Storage

The passwords were hashed while storing in the database. This was implemented using bcrypt [5] library which is based on blowfish cipher. This ensured that even if a malicious user gained access into the database, he/she can't view the passwords.

9. Form Validations: Front End & Back End

We have added validations to each transaction input from using jQuery and html pattern tag. On backend we again do the same validation to add additional layer of validation.

Validation checks include transaction amount cannot be negative, account balance cannot go below minimum required balance, etc.

10. Right Click disabled

We have disabled right click functionality on important pages to prevent copying of sensitive information.

11. Refresh disabled

We have disabled refresh functionality on important pages to prevent resubmission of post requests.

12. Back navigation disabled

We have disabled backward navigation functionality on each page to prevent access to state of completed transactions as well as resubmission of transactions.

13. Account locking after 3 unsuccessful attempts

After 3 failed login attempts for a username we lock that account for one hour. This prevents brute-force attack.

14. Handling DoS Attacks in the form of TCP Syn Floods

This was done by reducing number of half open TCP connections and rate-limiting on Apache Tomcat server. With the help of this we prevented the server running out of memory, when attacker tries to do a DoS attack.

VI. VULNERABILITIES

After testing, the following vulnerabilities were reported for the project. Only a few were accepted as vulnerabilities while the rest of them were decided to be classified as not a vulnerability. More details have been provided for each of the reported issue.

A. Accepted Vulnerabilities

1. Home Page redirection

Issue: During forgot password workflow, when a user is on the OTP page, on clicking the Home button, the user is redirected to the home page.
Resolution: This issue was investigated thoroughly and was found that a coding error has caused this issue. It can be easily resolved by updating the code. The investigation also confirmed that the issue does not have any side effects on the system and all the other security features continue to work as expected.

B. Rejected Vulnerabilities

1. Login form values not clear after failed login
Reported Issue: After login fails and redirected back to login page, the form values aren't cleared
Justification: This is a design implementation to make login page more user friendly. We don't reset login form after failed login. If for any scenario, login fails due to incorrect username/password or wrong captcha then user doesn't have to retype those fields. However, these form values aren't stored and are cleared if you refresh the page or close the browser.

2. PKI not *working*
Reported Issue: PKI authentication fails
Justification: Our PKI implementation does certain computations to verify public and private key on client side which is a bit time consuming process. We found that our assigned VM didn't have enough resources to scale well to our application, specifically our PKI implementation part and timed out the request before operation on client side was completed. We had developed the application locally as working with VM seemed near impossible due to random crashes and glitches and didn't encounter any issues locally with our implementation which led us to believe it would work properly on VM. We tried various ways to mitigate this issue but couldn't find a quick feasible solution which didn't involve any major design changes.

3. *Captcha not working*

Reported Issue: It was reported that captcha wasn't working for someone

Justification: Captcha is case sensitive which I guess many people might have issue with considering 1, l and I look pretty similar. However we couldn't reproduce this issue and found it to be working as designed.

4. *JavaScript pages can be accessed from website*

Reported Issue: Some users noted that .js files are easily accessible from website and can be tampered with to disable scripts which disable refresh, back button and right click

Justification: The JavaScript pages accessible through website normal support scripts which are used to disable back-button, refresh and right-click on website and other similar minor computational operations to be done on client side. They do not expose any vulnerable information to client, nor can we control behavior on client machine if these scripts are disabled.

5. *Regular Employee can add a customer*

Reported Issue: Some users mentioned that regular employees shouldn't be allowed to and customer/merchant accounts

Justification: This was design choice as we believe we didn't want to burden manager with all the operations each time a new customer/merchant account needed to be added to the system. We allow regular employees to add customers/merchant accounts to the system but it still a manager is still required to physically verify all the input information and approve the account, before account is activated and ready for use.

6. *Server leakage info*

Reported Issue: Server information could be found by appending /docs/setup/html to the website URL and some users reported that it can used to access the server.

Justification: Navigating to the page only reveals the server being used (viz. in our case Tomcat 7) which isn't a highly sensitive information. Moreover, no users or default users are set on server which means it cannot be accessed remotely.

7. *Forms can execute scripts*

Reported Issue: Some users mentioned that certain forms accepted strings like "<script></script>" can be exploited to execute malicious scripts on website

Justification: All accepted strings are sanitized and made sure that they are only used as strings and couldn't be misused in any way. A user can pick username like "root" but that doesn't mean it would give him root access to the system.

8. *"Multiple sessions not allowed" message*

Reported Issue: It was reported that people were receiving this message on login

Justification: This is a security feature implemented to ensure that older session needs to be closed before opening new session for another user on same browser. Many times, people log into the website, close the tab but don't necessarily close the browser. While the session automatically expires in 30 minutes but it is still active and needs to be properly closed by logging out or restarting the browser.

9. *Ability to access webpage using URL*

Reported Issue: Some users reported that after logging into the website and closing the tab, they could use URL to access a webpage

Justification: This is expected as per design. A session remains active for 30 minutes, closing the browser or logout action, whichever occurs earlier. Closing a tab or closing window doesn't mean same as closing the browser entirely.

10. *Future dates are accepted*

Reported Issue: Some users reported that future dates are being accepted in date of birth field

Justification: Not a security issue but design oversight. In any case, we expect manager to verify all information before approving a user account for merchant/customer.

11. *Session Hijacking*

Reported Issue: Many users reported that session hijacking is possible and illustrated by copying session cookie and reinserting it in another browser using firebug and cookie manager

Justification: The reported method requires attackers to access to machine to acquire session cookies (which are valid for 30 minutes maximum) physically or using malware. This behavior is not currently covered in scope of this project. The data available on network or transport layers is encrypted by SSL and session variables cannot be acquired by attacker using this method

12. *Form/Page refresh is not working*

Justification: Refresh was disabled as a security measure

13. *SSN is visible and accepts characters*

Justification: SSN is visible only during input or approval process in which case manager needs to verify the SSN and do background checks before approving user access to the system. Also we allow SSN to have characters as well since SSN can contain hyphens and we leave SSN validation and verification up to manager

14. *SQL Injection*

Reported Issue: Some users reported that SQL injection is possible as users can input values such as "a OR 1 == 1"

Justification: We use Hibernate Query Language with parameterized statements to format SQL scripts and such strings are sanitized

15. *Email not working*

Justification: Couldn't reproduce the issue

16. *Couldn't add new customer*

Justification: Couldn't reproduce the issue

17. *Regular employee can see critical transactions*

Reported Issue: Some users reported that regular employees shouldn't be allowed to view critical transactions

Justification: We felt allowing regular employees to view transactions was a better design choice as they could confirm on customer's behalf the status of a transaction, in case

customer is unable to access the system for some reason. But, approval/denial of critical transaction still requires a manager.

18. Error message for Manager update
Reported Issue: Some users experienced an error message when updating information from manager account
Justification: Couldn't reproduce the issue. Most likely cause of behavior would be improper data being entered which doesn't pass the controller level validations

19. No email message when a new user is added to the system
Reported Issue: It was reported that an email wasn't sent when a new user was added to the system
Justification: When a new user is added to the system, it is required to be approved by a manager and the manager assigned for the approval process is picked at random. If system has multiple managers, the account pending approval could be assigned to anyone and until the account is approved by the assigned manager, the account isn't activated. The email message with temporary password isn't sent out until the account is activated. This is the implemented system.

20. Password reset issue
Reported Issue: OTP is sent to wrong email
Justification: Unable to reproduce this issue

21. CSRF attack
Justification: An attacker requires highly critical knowledge of the system such entire form structure which is only possible if attacker has infected client machine and gathering all such information and relaying it back. Dealing with infected client machines isn't really within scope of our project.

22. Refreshing captcha refreshes page
Justification: This is a design implementation

23. Not enough input validation
Reported Issue: Some users reported that there wasn't enough input validation
Justification: We have good combination of backend and frontend validation and since front-end validation isn't very reliable, we focused more on backend validation.

24. Cannot transfer amount
Reported Issue: One user reported not being able to transfer a small amount even though having positive balance
Justification: System requires to maintain a minimum balance which was outlined in the user guide. If the account do not meet the minimum balance criteria, transactions are declined

25. Infinite redirects
Reported Issue: Error page redirects for verify PKI failure
Justification: Couldn't reproduce the issue. Also clicking on 'click here' redirects to home page

26. Man in the middle attack
Reported Issue: On user reported that certain session variables like username and password are visible in developer tools and can be accessed using man in the middle attack
Justification: All the network and transport layer data is secured by SSL and cannot be sniffed by malicious routers. To access session variables, one needs to access machine physically or by means of malware which isn't covered in current scope of project

27. Invalid date of birth
Reported Issue: One user reported that form accepts invalid date of birth like 15/15/2015
Justification: In each part wherever dates need to be inserted, a date picker UI is available to pick valid dates. If user still decides to input malicious data, dates are sanitized in backend and automatically converted to 15/3/2016 in given case.

28. No users available for approval
Reported Issue: After adding a user and checking from manager account, no users were available for approval
Justification: This is an expected design behavior. Whenever a new user is added to the system, a random manager is assigned for approval process and only visible from assigned manager's account. We couldn't reproduce the issue where no manager was assigned to a newly added customer account.

29. Logout option before login
Reported Issue: One user reported that the logout option is available before login
Justification: This is a usability concern. This does not in any way affect the security

30. Session does not expire after some time
Justification: Session timeout works as expected.

VII. CONCLUSION

During the project, the whole team contributed towards the project design, implementation, testing and presentation. All the assigned tasks were completed on time and as per the schedule. All the required functional and non-functional requirements were implemented with utmost care to ensure a robust and secure system. Additional security features were implemented to make sure that the system foolproof. Once the vulnerabilities were reported by external testers, the team skillfully investigated each of the issue and found that only one of the reported vulnerabilities was a genuine. The rest of them have been classified as non-issues. By working on this project, all team members have gained immense knowledge on Spring MVC, hands on experience in designing and implementing the security features and to work in a large project group.

VIII. ACKNOWLEDGEMENT

We would like to thank Professor SS Yau and the Teaching Assistant Yaozhong Song for their immense support during the entire course of the project and the coursework.

IX. REFERENCES

[1] JQuery virtual keyboard, [online].
Source: http://keith-wood.name/keypad.html.
[2] Public Key Infrastructure, [online].
Source: https://en.wikipedia.org/wiki/Public_key_infrastructure.
[3] RSA, [online].
Source: https://en.wikipedia.org/wiki/RSA_(cryptosystem).
[4] CAPTCHA, [online]. Available: https://en.wikipedia.org/wiki/CAPTCHA.
[5] bcrypt, [online]. Source: https://en.wikipedia.org/wiki/Bcrypt.