CSE 545 – A Secure Banking System
Group 5 – C55 Bank

Abhishek Zambre, Aditya Tangirala, Dhanashree Adhikari, Dilip Kasargod, Gautham Adhithya Thirunavukkarasu, Durga Prasad Reddy Kothinti, Hanumantha Narayana Harish Pendela, Mayank Khullar, Pankaj Singh, Prabhat Racherla, Saurabh Singh, Srinivas Puranam, Vidya Sridhar, Vineet Mishra, Vishaka Bhaskaran, Vishwak Thatikonda,
School of Computing, Informatics, and Decision Systems Engineering
Arizona State University

***Abstract*—This is the final group project report for designing and creating a secure online banking application as a part of the course project. It contains the high level application description of the banking system. Functional, Nonfunctional features and their corresponding details have been listed and explained. The implemented security features of the system and its vulnerabilities have been discussed in detail. The roles and responsibilities played by team members listed above have been summarized.**

I. Introduction

The C55 banking application is designed to be extremely user friendly. For a naïve user it is as easy to use as a flip of a switch. The banking application closely matches up to the working of a real world, highly reliable banking application. The banking system allows usage of 5 different types of users namely, Admin, Manager, Internal Employee, External User(Customer), Merchant. The application facilitates the external user to login and perform online transactions, money transfer, view/download transaction statements, change personal details and debit/credit funds to other user accounts. The customers can perform transactions by payment of existing dues towards a merchant account. The internal users look over the critical transactions in order to approve or decline, and check for dubious transactions. The managers can use their capabilities to perform all of the internal user's duties and also create new employees. System administrator is responsible for maintaining the system health. He is also authorized to access the system logs which are required to keep suspicious activity at bay. Another important functionality of admin is the internal user account management. The merchants can perform all the functions of a customer and has a higher upper limit for the transactions compared to a customer account, as the amount of inflow and outflow of funds is expected to be comparatively higher.

The banking application matches up to the security expectations of a customer by incorporating the latest security features, making the banking system a virtual fort knox. The security features like OTP, password hashing, Public Key encryption are a few of the highly sophisticated features that have been implemented in the application. The system is successfully implemented and available for users.

II. System Design

A. Product Perspective

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Our Secure Banking Application - "C55 Bank" has been designed and developed to assist users to perform all major banking functionalities online in a secure manner. Major actors in the system include external users, internal users and system administrator. External user can perform operations like fund transfers, credit card operations etc. Internal users, which includes regular bank employees and managers have a bunch of functionalities that they can perform - like creating customer accounts, initiating fund transfers, approving critical fund transfers, initiating customer profile updates etc. The system also has another actor - System Admin: whose functionalities are focused around employee account management and system management. Feature and functionalities of each of the actors are explained in detail in this document in the 'User Characteristics' section.

Given the criticality of a banking application, special emphasis was given to security related features during our product development. Certain extremely important security features like proper session management, PKI implementation, virtual keyboard, access control mechanisms, end-point user authentication etc were appropriately incorporated. We also obtained a signed SSL certificate from an authentic external CA - COMODO instead of just demonstrating this feature by having a self signed certificate.

B. Product Architecture

Our product is developed using the spring MVC framework. The Spring Web model-view-controller (MVC) framework is designed around a DispatcherServlet that dispatches requests to handlers, with configurable handler mappings. The default handler is based on the @Controller and @RequestMapping annotations, offering a wide range of flexible handling methods. With the introduction of Spring 3.0, the @Controller mechanism also allows you to create RESTful Web sites and applications, through the @PathVariable annotation and other features [1].

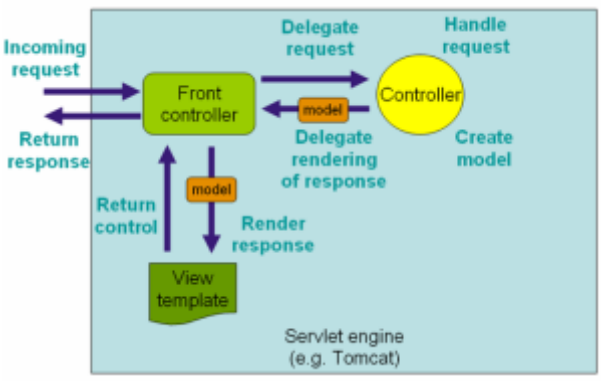

Figure 1. Servlet Engine

One of the main reasons for considering the use of Spring framework to build our application was the extensive set of security features this framework has on offer. A few key security aspects Spring MVC framework provides us with include:

- Require authentication to every URL in your application
- Support for managing User sessions. (This includes handling session timeouts).
- CSRF attack prevention
- Session Fixation protection
- Security Header integration
    - HTTP Strict Transport Security for secure requests
    - X-Content-Type-Options integration
    - Cache Control (can be overridden later by your application to allow caching of your static resources)
    - X-XSS-Protection integration

The class diagram for the system is as follows:

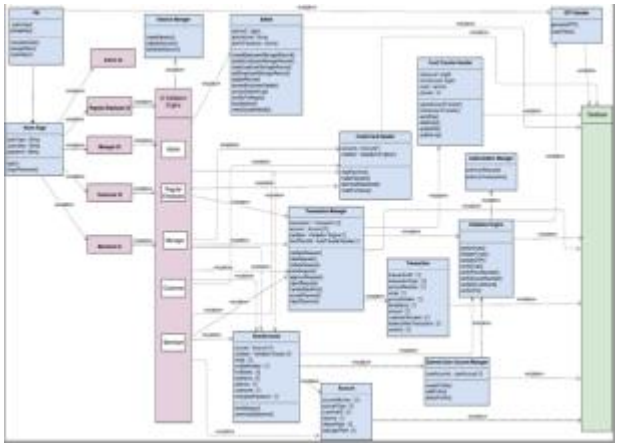

Figure 2: Class Diagram

III. USER CHARACTERISTICS

A. Internal Users

System Administrator:

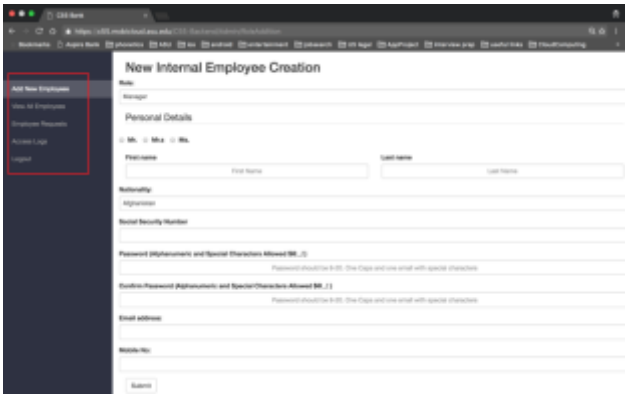

Figure 3: Admin page

1. Can verify internal user's requests
    a. System Admin is provided with an interface for updating internal users requests along with approval and reject button.
2. Can view, create, modify, and delete internal users' account.
    a. The admin can add new internal users (regular employee or manager).
    b. The role of the new employee can be chosen from the drop down list.
    c. The personal details of the new employee are filled accordingly which includes details like Nationality of the employee can be chosen from the drop down list and social security number (SSN) which must be 10 digits long.
3. Can access the system log file. (System log file is only accessible to the administrator)
    a. The admin is the only actor who can access logs.
    b. The admin provides the date for which he/she needs the log
    c. The "get log" button gets the log and displays it. The admin can view the log he needs.
    d. Only the admin can access logs and this conforms to the "Default deny"(5) step and "Adhere to the principle of least privilege"(6) step of the secure coding practices.
4. Can access PII. (PII is only accessible to the administrator)
    a. The admin can view all the employees in the bank.
    b. The "view PII" button displays the employee personal details like SSN.
    c. These can be updated/edited by the admin

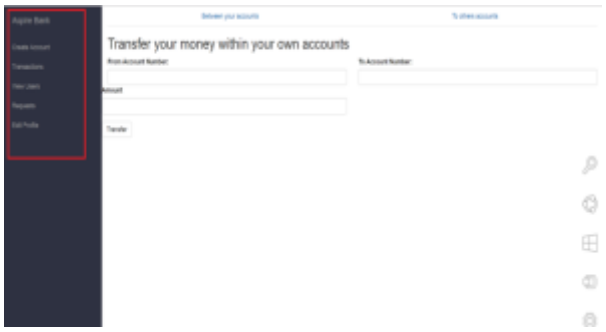

Figure 4: Transfer funds

System Manager

The role of a Manager is carried out by a bank manager, the role has higher responsibilities related to transactions approval and also has higher service privileges in the bank and is usually carried out by a single manager for a particular bank even though the code has provided functionality for multiple managers. The manager overlooks the transactions between the customers, services the customer's requests, creates accounts for customers and can create transactions on behalf of the customer as well as approve service request from a Regular Employee.

Figure 5: Account creation

1. Can view, create, modify, delete and authorize non-critical transactions
    a. Manager can create, modify a particular non-critical transaction. A non-critical transaction pertains to a transaction below the value of $5000. The Manager has the authority to modify, create and authorize the transaction but it has to be further approved by the customer to whom the account belongs to.
2. Can view, create, modify, and delete external users' accounts
    a. Manager can view, create, modify and delete the customer's account. The Regular Employee and the Manager are the only users who can create/delete an account for the customer.
3. Can authorize critical transactions, can initiate modification of personal account
    a. All critical transaction (Transactions above $5000) need to be approved by Manager. The transaction would be at a pending stage, in which the amount will be deducted from the sender's account but not credited to the receivers account till the approval is done.
4. Can verify external users' and verify regular employees' requests
    a. Manager is the sole authority to service a Regular Employee's request. Regular Employee's requests are subject to approval by the manager, and only after the approval can the Regular Employee's tasks can be implemented.

Regular Employee

The role of a Regular Employee is carried out by a normal bank employee, the role has higher privileges compared to the customer as it is an internal employee but considerably lower privileges that a Manager. The regular employee overlooks the transactions between the customers, services the customers' requests, creates accounts for customers and can create transactions on behalf of the customer.

1. Can view, create, modify, delete and authorize non-critical transactions
    a. Regular Employee can create, modify a particular non-critical transaction. A non-critical transaction pertains to a transaction below the value of $5000. The Employee has the authority to modify, create and authorize the transaction but it has to be further approved by the customer to whom the account belongs to.
2. Can view, create, modify, and delete external users' accounts
    a. Regular Employee can view, create, modify and delete the customer's account. The Regular Employee and the Manager are the only users that can create an account for the customer.

B. External Users

Customer

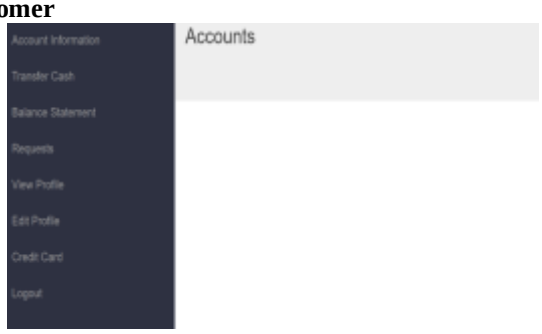

Figure 5: Customer profile

1. Can view, debit, credit, and transfer money from his/her personal bank account
    a. Customers can perform banking activities like transferring personal funds.
    b. They have limited privileges and require additional permissions for all the critical transactions
2. Can initiate modification of personal account
    a. They can perform trivial profile updates but cannot update phone numbers/email.
3. Can view, authorize and decline internal users' requests
    a. The changes like updating email/phone numbers are to performed by the internal users upon request from external users.
    b. Internal users make the necessary changes and these changes are reflected in the Request->Service Request tab of the external user.
    c. Upon approval from the external user, the changes are updated.
4. Account information tab shows the list of accounts of the customer in the bank. It lists out both savings and checking accounts.
5. In the transfer cash tab, the customer can transfer funds upon entering the correct OTPs sent to their mails/phones.
6. The Balance Statement tab shows the transactions of the customer at a given interval of time.
7. Request tabs are used to approve transaction/service requests raised by the internal employee upon request by the external customer.
8. View/Edit profile tabs are used for viewing the customer details and editing them if necessary.
9. Credit Card tab is used to make purchases using a credit card. Also used to settle the credit card bill.

Merchant

1. Can view, debit, credit, and transfer money from his/her personal bank account
    a. Merchants can perform banking activities like transferring funds to other account.
    b. However merchants should turn to Regular Employee if they require additional permissions for all the critical transactions
2. Merchants are the external users who should be able to accept payments in exchange of goods and services.
3. As a minimal requirement of this system five merchants have been created who can authorize payments on behalf of the user.
4. Can view, authorize and decline internal users' requests
    a. If the internal user raises a request to change an address or a phone number then the merchant should be able to accept or decline those requests.
    b. If the internal user makes any changes to the services they should reflect in the service request tab to the external user.
    c. Once the external user approves, the changes are updated in the database.
5. Can modify their personal information
6. They should be able to initiate a change to the account information such as address, phone number and so on.
7. Can perform payments on behalf of the customers
    a. Apart from these functionalities merchants can authorize payment by accepting debit or credit card credentials
    b. Then producing them to complete the payment process after the goods and services are delivered.

IV. Implementation

A. External Interface Implementation

1. External interface consists of web pages that allow the user to access the banking system.
2. Login page
    a. Login prompt
    b. Username field
    c. Password fields
    d. Role user is logging in as(Admin, Customer, Merchant, Employee, Manager)
3. Customer
    a. Account information
    b. Transfer Cash
        i. Between 2 different accounts
        ii. Between customer's accounts
        iii. Quick Pay
        iv. Debit/Credit cash
    c. Balance statement (Enter account number and start and end date)
    d. Requests
        i. Page containing requests raised by internal employees on customer behalf
    e. View Profile (Displays customer's information(PII))
    f. Edit Profile
        i. Change customer's first name
        ii. Change customer's last name
        iii. Change customer's Password
        iv. Change customer's Address
    g. Credit Card(Make Payment/Purchase)
    h. Logout
4. Administrator
    a. Add New Employees
    b. View Employee Details
    c. Approve Employee Requests
    d. Access System Logs
    e. Logout
5. Internal Employee
    a. Create Customer Account
    b. Initiate transaction on customer behalf
    c. Edit customer's profile based on account details
    d. Delete Existing customer accounts
    e. Approve transaction raised by customer (Large transactions)
    f. Logout
6. Merchant

a. All the interface items of the customer page
b. Page to approve payments by customer to merchant

B. Internal Interface Requirements

1. Internal Data Requirements
    a. All data stored in MySQL databaseUsed Hibernate for safe and secure querying
2. Design Interface
    a. User has maximum of 9 unsuccessful login attempts (Account will be locked after 10th unsuccessful one
    b. Employee PII accessible only by system administrator
    c. Needs to have and provide valid mobile provider and number
3. Other Requirements
    a. Uses Spring MVC and Rest API
    b. Apache Tomcat 8.0 used to host web services

C. Constraints

1. The website is also verified and signed by Trust Secure Seal. A Trust Webseal communicates that a third party verified by a business is legitimate and can be trusted.
2. Internal employees should login to the system using email ID. They do not have a username.
3. There is only one try for OTP. If wrong OTP is entered, no error message is displayed. A new OTP has to be requested again.
4. Self debit and credit function has no limit on critical amount. Every such request should be approved by regular employe and not by manager.
5. Our Secure Banking System(SBS) supports only 5 domestic mobile carriers.
6. SBS has 3 merchants created by default - Amazon, BestBuy and Ebay. There is no provision to create additional merchants.

D. Assumptions and Dependencies

1. It is assumed that the system developed shall be compatible with the given hardware requirements.
2. All system downtime issues that don't occur due to the hardware malfunctions, shall be fixed with the highest priority.
3. It is assumed that the email IDs and the phone numbers along with the carrier details provided by the external users are valid. No additional validation is done by the system in this regard, apart from the syntax checks.
4. Every external employee is provided with a Credit Card by default. Every credit card holder should have an account with the bank.
5. SBS allows transactions with 3 merchants. New merchants cannot be created/added by the user for purchases.
6. If any fund transfer exceeds $5000, it is considered as a critical transaction and internal employee approval is required.
7. Password is alphanumeric and must contain one special character.
8. No error message is displayed when incorrect login information is entered. The user gets redirected to the login page.
9. The regular employee is not provided with a *Forgot Password* functionality due to security policy.
10. A customer is allowed to have a maximum of 5 accounts.
11. When creating a new password, the required password specifications must be followed - minimum 8 alphanumeric characters with 1 special character.

E. System Setup and Deployment

For this project, we used Ubuntu Linux 16.04 LTS (Xenial Xerus) as an application server and Apache Tomcat (Version 8.5.8) for hosting our application which is an open source web server developed by Apache software foundation. It generally runs JSP, and Servlet among others. There is a built in web container called Catalina in the tomcat bin directory. It loads all http related requests and has privileges to instantiate the GET and POST method's object. It also uses Cynote i.e a http connector through network layer of the computer. All the execution is managed by JSP engine.

We have used Java Development Kit 1.8 for application development along with Spring Framework 4.3.4. For build generation and deployment, we have used Apache Maven 3.3.9. We have used Git as version control system for our code.

F. Maintenance and Troubleshooting

Once Apache Tomcat is installed, we need to deploy war file generated by build into Tomcat's webapps directory (Tomcat's default location is usually /var/lib/tomcat8 on Ubuntu). Once war file is deployed we can access application locally at *http://localhost:8080*.

In order deploy system into production we need to change conf/server.xml and have to add Domain URL at appropriate location. Tomcat can be maintained using "systemctl start|stop|restart tomcat8". For application troubleshooting logs can be found at *var/log/tomcat8*.

V. Security Feature Iimplementation

A. One Time Password (OTP)

One Time Password(OTP) is generated for any mode of transaction (a fund transfer to an account owned by the user, fund transfer to an account owned by another user, a fund transfer to an account owned by another user using his/her mobile phone or email ID). OTP is sent as an e-mail OR text message to the assigned user to their mobile via their registered mobile numbers depending on the option selected by the user. Every OTP is valid for 15 minutes since the time of generation.

B. Virtual Keyboard

Virtual keyboard is used in the project to ensure that it is not susceptible to keylogger attack as it uses a mouse instead of keyboard which is difficult to pinpoint using the key logger.

The key logger may be injected into the system using a malware or installed onto a public computer. Virtual keyboard is added to the Forgot Password page.

C. SSL

SSL Certificates are small data files that digitally bind a cryptographic key to an organization. When installed on a web server, it activates the padlock, and the https protocol and allows secure connections from a web server to a browser. Typically, SSL is used to secure transactions, data transfer and logins, and more recently, it is becoming the norm for secure browsing of social media sites.

When a certificate is successfully installed on your server, the application protocol (also known as HTTP) will change to HTTPs, where the 'S' stands for 'secure'. SSL certificates are usually provided by Certificate Authority (CA). For this project, we have used certificates provided by Comodo.

To ensure user that website is using end-to-end encryption, we configured TrustLogo provided by Namecheap.com.

D.Public Key Infrastructure (PKI)

Public Key Infrastructure is used for two purposes. One, for proving identity and two, for encrypting sensitive data. In our application PKI is implemented at two places. The first implementation is signed certificate. Whenever a website is entered in any user's browser, in order to prove the website's identity, the browser requests for the website's certificate. The browser checks the signature of the certificate and if it belongs to a well known certificate authority(CA), the website's source is identified as *trusted*.

For the second application of PKI, we look at the logs that are generated. As per the requirement only an admin user can view the logs. Each time the admin views logs, the generated logs for a particular date are hashed and signed. This encrypted value is stored separately. This helps the admin verify any unauthorized input in the log files.

E. CSRF Prevention

Cross-Site Request Forgery (CSRF) is a type of attack that occurs when a malicious website, email, blog, instant message, or program causes a user's web browser to perform an unwanted action on a trusted site for which the user is currently authenticated. Some frameworks handle invalid CSRF tokens by invaliding the user's session, but this causes its own problems. Instead by default Spring Security's CSRF protection produces an HTTP 403 access denied. This was customized by configuring the AccessDeniedHandler to process "InvalidCsrfTokenException".

F. Role Based Authentication Access

This is feature in the system that redirects to the user's homepage depending on the type of the user. The type of the user must be selected from a dropdown in the login panel and all the three fields the username, password and the user type are sent to different backend services that is responsible for the redirection depending on if the user. If the user is an admin the service login/admin is called, and if the user is a manager or an employee the service login/internal user is called, and if the user is a customer or a merchant the service login/externaluser is called. Role based authentication is very crucial in isolating the functionalities and hence provides a robust way of dealing with the escalated privileges.

G. Single Logging in One Session

When a user that is already authenticated tries to authenticate again, the application can deal with that event in one of a few ways. It can either invalidate the active session of the user and authenticate the user again with a new session, or allow both sessions to exist concurrently. This application makes sure that the Spring Security session registry is notified when the session is destroyed. To enable the scenario which allows multiple concurrent sessions for the same user the <session-management> element is used in the XML configuration where the maximum number of session is set to one.

H. Hashed Password Storage

Most attacks that take place in any web application are through forms. Because that is one of the ways in which you can send data to server. So it becomes crucial that you validate each and every input that you are sending to the backend. For the project we incorporate HTML and JavaScript form validation. We have taken care that any form is not submitted before it's properly validated. Numbers, text fields, drop down, radio button, file download, everything was validated. In order to secure the account of a person and prevent it from brute force attack we have ensured that, the password fields follow the standard conventions that is one small letter, one capital letter, one number and one special character, and min of 8 length characters. Like this we have followed a few more standard form validation techniques. Thus thwarting any attempt to send any malicious data to the backend.

I. Account locking

User Account will be locked after ten wrong attempts which can be unlocked by raising an unlocking service request to the employee.

VI. Vulnerability Report

A. Accepted Vulnerabilities

1. Improper parameter validation at function level
2. Clickjacking

B. Rejected Vulnerabilities

1. Bruteforce Password guessing
    a. We have 10 attempts for unsuccessful logins
2. Autocomplete enabled
    a. Not our web application vulnerability, it's a client side browser feature for user convenience. As a general guideline, user is discouraged to save passwords for critical applications like Bank accounts on browser.Eg: Chase Bank allows autocomplete feature.
3. Improper session management

    a. We have Spring framework in place which is facilitating and enforcing session management in our web application
4. No CAPTCHA for login
    a. Not a requirement. Can be bypassed, doesn't offer a reliable defense.
5. HTTPOnly flag not set for cookie
    a. We are not using cookie for session management and the cookie does not hold any specific information about the web application which can be used by an attacker for impersonation
6. Multiple accounts having same email
    a. Design decision. Same person can have multiple accounts linked to same email id(Saving, Checking, CC). Race condition not possible as all DB operations are atomic. Eg: Bank of Punjab and SBI
7. Logs show URLs
    a. It was a design decision. It was in accordance to provide Admin troubleshooting capabilities. And moreover, logs are accessible to Admin only who is a legitimate user entitled to access them.
8. StackTrace
    a. Exception handling on few parameters of different modules are not handled because of regression. Any significant information about codebase or database is not leaked out in form of output
9. Negative numbers allowed
    a. It was a design decision to facilitate any future inter-bank transactions,
10. Denial of Service
    a. This couldn't be reproduced on the application. There was no reported unintentional downtime of the application
11. Admin can modify details of internal employee
    a. It was a design decision. It was in accordance to the given requirements of the project.
12. Back/Forward button works after logout
    a. This couldn't be reproduced on the application.
13. Able to create blank records in DB
    a. This couldn't be reproduced on the application.
14. Multiple logins not allowed
    a. This was a design decision.
15. No unique field for phone number
    a. This was a design decision. Many accounts can be linked to same phone number.

C. Accepted Functionality Errors

Forgot password script not working properly.

We have found some serious vulnerability in our application which we didn't address during our development. The vulnerabilities found by others were: Improper parameter validation at function level, which is also a part of one of the OWASP top 10, and Clickjacking, this vulnerability can lead to compromise of credentials and potential identity theft. We will be adding strong sanitization and validation for all fields to nullify this risk. On top of that, we would be adding a function level parameter check for all variables. For clickjacking, we would be setting the X Frame header to DENY to counter the vulnerability.

VII. INDIVIDUAL REPORT

Abhishek Zambre

- Performed Linux server system setup with all the necessary tools such as JDK 1.8, Maven etc. Performed Apache Tomcat installation and configuration. Performed security sanitization of Tomcat (using Apache guidelines) and MySQL (using mysql_secure_installation script).
- Raised CSR (Certificate Signing Request) with public key to Comodo CA (Certificate Authority). Configured keystore with private key/public key pair.
- Worked with Comodo CA and TA/Lab staff for configuration of SSL/TLS certificate on proxy server in order to override ASU technology office's wild-card certificate.
- Configured SSL/TLS certificate provided by CA on Linux server and added the same into Tomcat configuration and helped Namecheap.com to verify system in order to get the TrustLogo.
- Installation of Database Server (MySQL). Also created and managed application database. Configuration of GitHub and Server, for automated build and deployment.

Aditya Tangirala

- Worked as a part of the front end team handling the design part of the different front end
- Implemented the PKI security certificate as a part of the PKI security team
- Was responsible for the creation of merchant UI and integrating it with the backend services
- Implemented the backend functionalities to download balance statement with the user
- Collaborated with the backend team to synchronize the front end, backend request response formats and also performed form validations

Dhanashree Adhikari

- Gathered analyzed and documented the requirements of regular employee.
- As a part of the security team finalised the requirements and PKI applications of the project.
- Worked with the vulnerability and functionality testing team to identify the functional and security flaws in the application.
- Implemented the second application of PKI which involved hashing and public key encryption of the system logs.
- Was responsible for project documentation at various stages of the project.

Dilip Kasargod (Sub group leader)

- Sub group leader and point of contact for user interface related queries or problems.
- Gather use cases for Employee and Manager
- Design flows and front end requirements for Employee and Manager roles
- Designed and created a universal framework for front end.
- Implemented all flows of Regular Employee and Manager in front end
- Worked on integrating several backend-frontend flows for several REST APIs with the back-end team
- Coordinated, facilitated and supervised front-end functionalities, technical and functional issue as a front-end subgroup leader.

Gautham Adhithya Thirunavukkarasu

- Gautham participated and contributed to software requirement specifications, backend and DB design discussions.
- He was also part of team that built prototypes to test Spring, Hibernate and REST APIs.
- In the development phase he worked on credit card functionalities such as credit card creation, purchases using credit card, deletion and payment.
- Gautham helped ensure that critical functionalities worked by testing and fixing bugs.

Durga Prasad Reddy Kothinti (Sub Group leader)

- Requirement Analysis and Design phase: He gathered, analysed and documented requirements related to Admin. Keeping security as a major goal, he played a key role in designing, presenting and documenting the product architecture. He also lead the team that developed and documented the class diagrams which served as guideline throughout the product development
- Implementations: He implemented user account management functionalities from backend which include user creation, deletion, profile updates, accounts creation, deletion etc. He implemented self debit and self credit functionalities of the customer. Also, he implemented password handling logic that includes hashing.
- Post-deployment: He lead a small team that resolved certain issues post deployment by co-ordinating with TA and making changes in the DB on server. He also contributed in the verifying and validating vulnerabilities reported by other students and also, in defending them in the class discussion.
- As a subgroup leader: As the backend team leader, he supervised and coordinated backend team's activities. Worked closely with each individual member of the backend team to actively resolve the nitty-gritties of features he/she was developing whenever they were blocked. He acted as a single point of contact for all backend activities.
- Other important activities
    - He took technical training sessions to help a few of the team members who had no prior experience on a few of the technologies used in the project.
    - He developed initial project setup with all dependencies and frameworks like spring, hibernate etc configured which was passed on to the team members to work on.
    - Also, he setup, owned and maintained the private Github repository making sure that the product's health was maintained after code commits from individual developers.

Harish Pendela

- He gathered, analyzed, and documented the requirements for database and designed, owned, and maintained the *schema/database* throughout the product life cycle.
- He designed, and developed *Transaction Manager* to support all fund transfer activities. He also designed, and developed the *quick-pay* module of fund transfer that supports fund transfer through mobile number and email. He was also responsible for maintaining entire system wide *user transactions* which are required to generate a bank statement.
- He designed, and maintained entire back-end *REST API*s i.e., identify the request/response format, the URL format, and ensure the consistency of response format across all modules.
- He contributed and reviewed SRS document. He also reviewed Test plan, Class Diagram, and User Guide. He also gathered, analyzed, and documented the requirements of external user. Documented the use cases for external user.

Mayank Khullar

- As a member of front-end Development sub group gathered, analyzed and documented requirements related to Manager.
- Developed the Spring framework on which the frontend and Backend code was integrated and explained the Spring MVC and Spring Securities flow to the team.
- Implemented Session Management using "SessionManagementFilter" of Spring Framework and the Security features such as preventing CSRF attack by configuring the AccessDeniedHandler to process "InvalidCsrfTokenException", implementing session timeout, advanced web features such as HTTPS channel security, port mapping, concurrency control, prevention of session tracking using Cookies, prevention of session fixation attacks using Spring Security.
- Implemented the Login and Logout functionality, worked on Credit Functionality of Customer and Merchant and Implemented the front-end of OTP functionality.
- Wrote Ajax calls made to Rest-Controllers and Spring Forms for calling Controllers with CSRF protection.
- During integration phase combined backend and frontend team's code into a single workflow of the application.

Pankaj Singh (Group leader)

- Contributed in clarifying and gathering user roles and functionalities in the product with TA and team.
- Coordinated, contributed in designing product architecture along with writing and reviewing related documents.
- Developed Proof of concepts with Spring Framework and sessions integrating UI pages with Ajax calls.
- Guided and mentored team members while writing UI code and helped in resolving technical problems during end to end integration with backend.
- Owned and implemented User Interface accompanied by integration of end to end flow of System Admin User functionalities.
- As the team leader, arranged meetings for discussions in collaborative environment; distributed responsibilities to team members and reviewed their work, helped them in resolving any technical, functional or behavioral difficulties.
- Submitted final work of the team; clarified issues and helped testers from other teams in understanding and accessing our website during vulnerability testing phase.

Prabhat Racherla

- Worked as a member of the front end development sub group.
- Designed and implemented all the screens for customer actor.
- Developed the code for handling form validation for most of the forms.
- Helped other team members to make their front end web pages.
- Came up with logic to download the account statement in PDF format.

Saurabh Singh

- Worked under UI team for the front end development of the Bank Website. Interacted with back end team for finalising the design for UI for Regular Employee.
- Implemented all the UI pages for Regular Employee actor.
- Implemented the Input validation for Regular Employee and Customer page.
- Interacted and cooperated with other team members for solving critical issues.
- Implemented and resolved Rest API calls issue while integrating front end UI to backend Rest Services.

Srinivas Puranam

- Srinivas Puranam played a major role throughout the lifecycle of the project. He was a part of the backend and security team.
- Played a crucial part in designing the High-Level design document. In the design phase of the project, he has collected and analyzed the requirements of the External User/Merchant.
- Played a major role in completing the class diagram. Documented the necessary use cases to implement an effective development plan.
- In the development phase of the project, he was responsible in completing External User/Merchant functionalities. He has tested all the necessary functionalities and resolved critical issues.
- Played a vital role in helping other team members resolve major issues and coordinated with the front-end team. He was also responsible for successful integration of the project.

Vidya Sridhar

- Formulated and documented the requirements for regular employee
- Participated and contributed to the overall functional design of the system
- Ownership of creating and updating the SRS and class diagram and coordinated in the creation of user guide
- In the testing phase, performed the final testing of internal and external user functionalities of the system.

Vineet Mishra (Sub Group leader)

- Subgroup leader and point of contact for all security related issues. Collaborated and communicated with other subgroup leaders and teams to ensure the security was considered throughout the development cycle
- Gathered analyzed and documented the requirements of merchant. Documented the use cases for merchant
- Helped in designing and implementing the security features of the project like PKI, Signed certificates, hashes, etc
- Worked with the vulnerability and functionality testing team to identify the functional and security flaws in the application.
- Was a part of the project design team and ensured all important security requirements were included in the design. Contributed to project documentation at various stages of the project.
- Was responsible for verifying and testing the vulnerabilities reported by other students. Gave a presentation and defended vulnerabilities which weren't valid, in the class and blackboard.

Vishaka Bhaskaran

- Vishaka gathered, analyzed and documented requirements for admin.
- She was a part of the backend team that came up with the application's schema.
- She played a key role during the implementation phase by taking up the ownership to implement external user functionalities.
- She was part of the final functional and non-functional testing of the application.

- She took the ownership of creating the user guide and lent a helping hand in creating the class diagram and SRS documents.

Vishwak Thatikonda

- Vishwak played a pivotal role during the project design and testing phases.
- He was a part of the database team that came up with a robust schema for the application.
- All the critical aspects of the application were designed keeping the usability factor in the forefront.
- He also played an important role during the implementation phase by taking up the ownership to implement the internal user functionalities.
- He also took ownership to complete the final functional and non-functional testing of the application. He was the first point of contact for all the testing related activities.

VIII. Conclusion

We were successful in implementing a functional and secure application. In the process we also learned a lot about software security. The course gave us a theoretical understanding of security principles and the project helped us understand them from a practical perspective. We did make some implementation mistakes but this was a great learning experience for us. This learning has ensured that we do not repeat the same mistakes in our professional career where the stakes are much higher. We also got an opportunity to work with a large team. This experience will help in our future endeavors. The project taught us a lot about the importance of security in software and how minor looking code bugs can lead to major security loopholes in the application.

Acknowledgement

We would like to express my heartfelt gratitude to Dr. Stephen Yau for giving insight and guidance throughout the course. We would also like to thank the Professor for the support that was extended by him during our interaction sessions and clarifying any questions that we had while doing the project.

We would like to thank Mr. Yaozhong Song and Ms. Bhargavi Rajagopalan for taking time to guide the team and clarifying questions that came up during the working duration of the project.

Finally, we would like to thank our team for the relentless support that they provided and work with patience.

References

[1] http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html