Accounts - Customer Portal - Banking App - FS Java Capstone

Accounts

- Customer Portal: Transaction Management
- Customer Portal: Account Registration
- Customer Portal: User Loyalty Points (cashback, miles, points) History (Stretch)
- Customer Portal: Investing (Stretch)

Story	Description	Acceptance Criteria	Story Points
Customer Portal: Transaction Management			
View Transaction	As a user, I want to view all my banking transactions so that I can track and manage my spending, deposits, withdrawals, and transfers effectively. This function should provide a clear, comprehensive, and easily navigable display of all transactions across my various bank accounts. It should also include filters and sorting features to help me organize and review my transactions based on criteria like date, amount, transaction type, etc.	1. **Functionality** ◦ The system should display a list of all past transactions associated with the user's account. ◦ Transactions should include details such as date, amount, transaction type (debit/credit), and recipient/sender information. ◦ The user should have the option to filter transactions by date range and type. 2. **Security** ◦ Ensure that all transaction data is displayed over a secure connection. ◦ Implement role-based access control; only the account owner and authorized users can view transaction details. ◦ Sensitive data like account numbers should be partially masked. 3. **Validation** ◦ The system should validate the date range inputs for filtering transactions and prompt the user if the input is invalid. ◦ Transaction data should be validated to ensure completeness and accuracy before display. 4. **Error Handling** ◦ If transaction data fails to load, display a user-friendly error message. ◦ Provide a 'Retry' option if the initial attempt to fetch transaction data fails.	

		◦ In case of an invalid request (e.g., an unsupported filter), inform the user with a clear message. 5. **Output/Notification** ◦ Upon successful loading, the transaction data should be displayed in an easy-to-read format. ◦ If there are no transactions for the selected filters, display a message indicating 'No transactions found'. 6. **Logging** ◦ Log all user actions related to viewing transactions for auditing purposes. ◦ Any errors or exceptions should be logged with sufficient detail for troubleshooting.	
Create Transaction	As a user, I want to be able to create transactions (like money transfers, bill payments, and deposits) within my account so that I can manage my banking needs efficiently from anywhere and at any time. This feature is vital to meet the basic functionality of a modern banking application. It should allow transactions between my own accounts, as well as transactions to other bank users or institutions.	1. **Functionality** ◦ The User Portal must allow the creation of different types of transactions including transfers, bill payments, and deposits. ◦ Users should be able to select the account from which the transaction is made. ◦ The portal should provide a field for entering the transaction amount. ◦ Options for adding transaction notes or descriptions should be available. ◦ There should be a confirmation step before finalizing the transaction. 2. **Security** ◦ All transaction data must be transmitted over secure, encrypted connections. ◦ User authentication (e.g., password or biometric verification) is required before initiating a transaction. ◦ Implement timeout for session inactivity to prevent unauthorized access. 3. **Validation** ◦ Check for the validity of the account details entered. ◦ Ensure the transaction amount does not exceed the account balance. ◦ Validate transaction input fields for data type and format (e.g., numeric values for amount). 4. **Error Handling** ◦ Display user-friendly error messages for failed transactions due to network issues, insufficient funds, or invalid account details. ◦ Offer retry options in case of transaction failures. 5. **Output/Notification**	

		◦ On successful transaction completion, show a confirmation message with transaction details. ◦ Send a notification (email or SMS) to the user confirming the transaction. 6. **Logging** ◦ Log all transaction activities including successful and failed attempts, with timestamps. ◦ Ensure logs capture key details like transaction type, amount, and account involved, while maintaining user privacy.	
Sort Transactions	As a banking application user, I would like to be able to sort my transactions so that I can manage and view them in a sequence that best fits my needs.	1. **Functionality** ◦ The system should allow users to sort transactions by date, amount, and transaction type. ◦ Each sorting option should organize transactions in both ascending and descending order. ◦ The sorting feature should be easily accessible from the transaction management page. 2. **Security** ◦ Sorting requests should be handled in the frontend without exposing sensitive transaction details to backend services unnecessarily. ◦ The feature should comply with relevant data protection regulations for handling financial information. 3. **Validation** ◦ The system must validate the user's selection for sorting (e.g., valid date range, transaction types). ◦ Invalid sorting requests should prompt the user with an appropriate message to correct their input. 4. **Error Handling** ◦ In case of a failure to sort (e.g., service unavailability), the system should display a user-friendly error message. ◦ The system should handle unexpected sorting errors gracefully without crashing or freezing the user interface. 5. **Output/Notification** ◦ Upon successful sorting, the user interface should update to display transactions in the selected order without a full page reload. ◦ If no transactions meet the sorting criteria, a message should inform the user that there are no transactions to display. 6. **Logging**	

		◦ All user actions related to sorting transactions should be logged for auditing purposes. ◦ Any errors or exceptions encountered during the sorting process should also be logged with sufficient detail for debugging.	
Filter Transactions	As a user, I want to be able to filter my transactions so that I can easily view and manage my account transactions based on certain criteria. This includes filtering by transaction amount, transaction type (debit/credit), transaction date, and transaction category (such as groceries, utilities, salary, etc.).	1. **Functionality** ◦ The system shall provide a filter option in the User Portal under the Transaction Management section. ◦ Users shall be able to filter transactions by date range, amount range, and transaction type (e.g., deposit, withdrawal, transfer). ◦ The filtered results shall be displayed immediately after the user applies the filter criteria. 2. **Security** ◦ All transaction data displayed must adhere to the user's permission level and privacy settings. ◦ The filter feature shall not expose any transaction details of other users. ◦ Secure coding practices must be followed to prevent injection attacks or data leaks. 3. **Validation** ◦ Date fields shall validate for correct format (e.g., MM/DD/YYYY) and logical dates (e.g., start date cannot be after end date). ◦ Amount fields shall only accept numerical values and shall validate for logical range (e.g., start amount cannot be greater than end amount). 4. **Error Handling** ◦ If no transactions match the filter criteria, display a message: "No transactions found matching your criteria." ◦ Invalid inputs in any filter fields shall prompt an error message detailing the correct format or expected value range. 5. **Output/Notification** ◦ Upon successful application of filters, the system shall display the number of transactions found. ◦ If the filter operation fails due to a system error, display a notification: "Unable to filter transactions at this moment. Please try again later." 6. **Logging** ◦ Each filter operation shall be logged with details including user ID, timestamp, and filter criteria used.	

		◦ Any errors encountered during the filtering process shall be logged for further investigation by the technical team.	
Search Transaction	As a user, I want to have the ability to search my transactions, so that I can easily find specific transaction details based on various criteria such as transaction date, amount, type, recipient, and more. This feature should support quick navigation and filtering capabilities for efficient management and review of my banking transactions.	1. **Functionality** ◦ The search function should allow users to filter transactions by date, amount, transaction type (e.g., deposit, withdrawal), and transaction ID. ◦ Users should be able to sort the search results by date, amount, and transaction type. ◦ The system should display a summary of the transaction, including date, amount, transaction type, and a brief description etc. 2. **Security** ◦ All transaction searches and results must be conducted over a secure, encrypted connection. ◦ User authentication is required before accessing the search functionality. ◦ Access to transaction data should be restricted based on user roles and permissions. 3. **Validation** ◦ Input fields for dates and amounts should validate for correct format (e.g., mm/dd/yyyy for dates, numerical values for amounts). ◦ The system should prompt the user with an error message if the search criteria are invalid or if no matching transactions are found. 4. **Error Handling** ◦ In case of a system error or failure during the search process, the user should receive a clear and informative error message. ◦ The system should gracefully handle network or server issues, ensuring the user can retry the search once the issue is resolved. 5. **Output/Notification** ◦ Upon successful search, the transaction results should be displayed in a clear, readable format. ◦ If no transactions match the search criteria, the system should display a message indicating "No transactions found." 6. **Logging** ◦ All user search activities should be logged for audit purposes, including date, time, user ID, and search criteria used. ◦ Error logs should be maintained, capturing details of any issues encountered during the transaction search process.	

Customer Portal: Account Registration			
View Account Types	As a User , I want to be able to view my account details so that I can check my registration details and verify their accuracy.	1. **Functionality** ◦ The system displays a list of all available bank account types. ◦ Each account type includes a brief description and key features. ◦ The user can view more details for each account type upon selection. 2. **Security** ◦ The display of account types does not require user login; no sensitive data is shown. ◦ All data transmissions are encrypted using industry-standard protocols. 3. **Validation** ◦ The system checks the availability of account type data from the back-end before displaying. ◦ If data is not available or outdated, the system refrains from displaying incorrect or incomplete information. 4. **Error Handling** ◦ In case of a failure to retrieve account type data, a friendly error message is displayed. ◦ The system offers options to retry or contact support in case of persistent errors. 5. **Output/Notification** ◦ Upon successful display, the user receives visual confirmation through the orderly presentation of the account types. ◦ Any updates or changes in the account types are dynamically updated without needing a page refresh. 6. **Logging** ◦ All user interactions with the account types view feature are logged for future analysis and improvement of user experience. ◦ Error occurrences are logged with details for troubleshooting and system improvement.	
Register for Account	As a User, I want to be able to register for an account so that I can utilize the bank's online services and manage my finances digitally.	1. **Functionality** ◦ User should be able to register for a bank account ◦ Users are able to select a specific bank account type ◦ User should be able to enter any addition information that is required to register for the account 2. **Security** ◦ All user data is transmitted over a secure, encrypted connection (HTTPS). ◦ Sensitive information should be encrypted	

		3. **Validation** ◦ The system validates all input fields for format and completeness. ◦ Email addresses are verified for proper formatting. ◦ Social security numbers are validated against standard formats. ◦ The system checks for duplicate usernames or email addresses. 4. **Error Handling** ◦ Users are informed of any input errors (e.g., invalid email format, incomplete fields). ◦ Users receive clear instructions on how to correct the errors. ◦ The system prevents submission until all errors are resolved. 5. **Output/Notification** ◦ Upon successful bank account registration, users receive a confirmation message on the screen. ◦ Upon unsuccessful bank account registration, users receive a confirmation message on the screen that e.g. “Account Registration unsuccessful”. 6. **Logging** ◦ All registration attempts (successful or unsuccessful) are logged with timestamps. ◦ System logs capture user input errors and the nature of the errors. ◦ Security-related events (e.g., multiple failed attempts) are flagged in the system logs for review.	
Account Registration Confirmation	As a User, I want to receive confirmation after successfully registering an account so that I can be sure that my registration was successful and I can start using the banking services.	1. **Functionality** ◦ The system shall send a confirmation message to the user’s registered email upon successful account registration if approved or unapproved. 2. **Security** ◦ All communication during the confirmation process shall be encrypted. ◦ The system shall implement measures to prevent brute force attacks on the confirmation process. 3. **Error Handling** ◦ The system shall provide clear error messages for invalid or expired confirmation links or codes. ◦ In case of system errors during the confirmation process, the system shall prompt the user to try again later.	

		4. **Output/Notification** ◦ Upon successful confirmation, the user shall receive a notification of successful account activation. ◦ If the confirmation fails, the user shall receive an appropriate notification explaining the reason. 5. **Logging** ◦ All user actions during the confirmation process shall be logged for audit purposes. ◦ System errors and failed confirmation attempts shall also be logged for further analysis.
Customer Portal: User Loyalty Points (cashback, miles, points) History (Stretch)		
Account Points History	As a user, I want to be able to view the history of my loyalty points (cashback, miles, points) in the banking application, so that I can track my earnings and redemptions over time. This feature should provide a detailed history including dates, points earned or redeemed, and the corresponding transactions.	1. **Functionality** ◦ The system shall display the history of loyalty points earned and redeemed. ◦ The history shall include details such as date of transaction, number of points earned or redeemed, and a brief description of the transaction. ◦ The user shall be able to filter the history by date range. 2. **Security** ◦ Access to the loyalty points history shall be restricted to authenticated users only. ◦ The system shall ensure that users can only access their own points history. ◦ Secure communication protocols shall be used for data transmission. 3. **Validation** ◦ The system shall validate date ranges entered by the user for filtering history. ◦ Any invalid date range inputs shall prompt an appropriate error message. 4. **Error Handling** ◦ In case of failure to retrieve points history due to server or connectivity issues, an error message shall be displayed. ◦ The error message shall be clear and non-technical, guiding the user to try again later. 5. **Output/Notification** ◦ Upon successful retrieval, the points history shall be displayed in a clear, readable format. ◦ If no points history is found for the selected date range, a message shall indicate that no records are available. 6. **Logging** ◦ All user interactions with the loyalty points history feature shall be logged.

		◦ Any errors encountered during the usage of this feature shall also be logged for further analysis.	
Customer Portal: Investing (Stretch)			
View Stocks	As a user, I want to view my stock portfolio within the banking application's user portal, so I can easily monitor my investment performance, check stock prices, and view key details about my investments in real-time.	1. **Functionality** ◦ The system displays a list of stocks currently owned by the user. ◦ Users can view real-time prices and performance metrics for each stock. ◦ The system provides detailed views for individual stocks, including historical performance charts. 2. **Security** ◦ Stock viewing is protected by user authentication; only the logged-in user can view their stock information. ◦ The system employs HTTPS for data transmission to ensure data privacy and integrity. ◦ Sensitive data, like stock quantities and values, are encrypted at rest. 3. **Validation** ◦ The system validates stock symbols and user ownership before displaying information. ◦ Invalid or unrecognized stock symbols result in a notification to the user without exposing system details. 4. **Error Handling** ◦ If stock data fails to load, the system shows a user-friendly error message. ◦ The system handles partial failures gracefully, showing available data with notifications about unavailable information. ◦ Errors due to network issues prompt the user to try again later. 5. **Output/Notification** ◦ Users receive a confirmation message when the stock data is successfully loaded. ◦ Any changes in the stock's status (like significant price changes) trigger notifications, if opted in by the user. 6. **Logging** ◦ The system logs all user interactions with the stock viewing feature for audit purposes. ◦ Errors and anomalies in stock data retrieval are logged for system monitoring and troubleshooting.	
Purchase Stocks	As a user, I want to be able to purchase stocks through the User Portal in the Banking	1. **Functionality**	

	Application, so that I can invest in various companies easily and manage my investments within the same platform.	- The feature allows users to search for stocks by company name or ticker symbol. - Users can view current stock prices and relevant information before purchasing. - There is a functionality for users to specify the number of shares they wish to purchase. - The system calculates the total cost of the transaction based on current stock prices and the number of shares. - Users can confirm or cancel the purchase before final submission. 2. **Security** - All stock purchase transactions require two-factor authentication. - Sensitive user data, including transaction details, are encrypted. - Session timeouts are implemented to prevent unauthorized access during inactivity. 3. **Validation** - Input fields for stock search, number of shares, and other relevant data have validation checks for correct format and data types. - The system validates the availability of stocks and sufficient funds in the user's account before allowing the transaction to proceed. 4. **Error Handling** - Clear error messages are displayed for invalid inputs or failed transactions. - The system handles cases of stock unavailability or insufficient funds by prompting appropriate user actions. - Error logs are maintained for failed transactions or system errors for troubleshooting. 5. **Output/Notification** - Upon successful purchase, a confirmation message with transaction details is displayed. - Users receive email notifications summarizing the transaction details. - The user's portfolio is updated in real-time to reflect the new stock purchase. 6. **Logging** - All stock purchase transactions are logged with e.g. user ID, timestamp, stock details, and transaction amount. - Logs are maintained for auditing and tracking purposes, and are accessible only to authorized personnel.	
Sell Stocks	As a user, I want to be able to sell stocks through the User Portal of my Banking Application, so that	1. **Functionality**	

	I can efficiently manage my investments within the User Portal - Investing section.	◦ The system provides an option for the user to select 'Sell Stocks' under the 'User Portal - Investing' section. ◦ Allows the user to search and select the stock they wish to sell from their investment portfolio. ◦ Provides an interface to enter the number of stocks to be sold and displays the current market price per stock. ◦ Shows a preview of the transaction, including the total sale value based on the current market price and the number of stocks selected. ◦ Includes a 'Confirm Sale' button to complete the transaction. 2. **Security** ◦ Ensures all stock sale transactions require user authentication (e.g., password or biometric verification) before processing. ◦ Implements SSL/TLS encryption for the transmission of user data and transaction details. ◦ Conducts automatic logout after a period of inactivity during the stock sale process. 3. **Validation** ◦ Checks that the number of stocks the user wishes to sell does not exceed the number available in their portfolio. ◦ Validates that the entered stock symbols correspond to valid stocks in the user's portfolio. ◦ Ensures that the stock sale value is recalculated in real-time if the market price changes during the transaction process. 4. **Error Handling** ◦ Displays an error message if there is a failure in fetching the current market price of the stock. ◦ Provides a clear error notification if the user attempts to sell more stocks than available in their portfolio. ◦ Alerts the user in case of transaction failures due to network issues or backend problems. 5. **Output/Notification** ◦ Confirms successful stock sale transactions with a detailed summary, including the number of stocks sold, sale value, and transaction date. ◦ Sends a notification to the user’s registered email or phone number upon the successful completion of the stock sale.

		◦ Updates the user's investment portfolio in real-time to reflect the sold stocks. 6. **Logging** ◦ Logs all user actions and data inputs during the stock sale process for audit purposes. ◦ Records successful and unsuccessful transaction attempts, including timestamps and user identifiers. ◦ Maintains logs of system errors and exceptions encountered during stock sale transactions.	