Date	Anisha			Arshia			Jeena			Pooja		
	What I did since Last Daily Scrum	**What I Plan to do today**	**Any Blockers**	**What I did since Last Daily Scrum**	**What I Plan to do today**	**Any Blockers**	**What I did since Last Daily Scrum**	**What I Plan to do today**	**Any Blockers**	**What I did since Last Daily Scrum**	**What I plan to do today**	**Any Blockers**
04/12	Understanding the requirements of the given Banking problem statement	Exploring different design patterns to implement in the project	No	Requirements Analysis	Explore more on design patterns to figure out scenarios to implement	No	Requirement Discussion. Planning the sprint. Discussing tech stack to implement the project	Analyze and design the system	No	Analysed System Requirements	Study the teck stack like Spring Boot Framework, brush up on React	Yes, new to spring boot, so needed references
04/13	Explored different design patterns, Had more clarity on the requiremnets	Understand the Tech stack decided by the team (SprinBoot, React and Redux)	No	Research on Design Patterns and how UI should look like, identified CRC relationships	Get familiarized with the proposed tech stack (React, Redux, Springboot)	No	Created rough class diagram with different components	Discussion on better understanding of requirements. Discussing on who will handle which component	No	Watched Springboot tutorial videos	Create UI WireFrames using Pencil too	No
04/14	Handson with Springboot	Understand React and Redux	No	Watched tutorials on Springboot framework and React, Redux.	Start handson examples with springboot	No	Decided on what component I'll take	check on how to schedule recurring tasks	No	Done with UI Wireframes	Research how admin refunds actually happen in banks	No
04/15	Handson with React.js	Understand Redux wiring	No	Did simple CRUD in Springboot App	Setup Workspace for frontend and Backend in appropriate IDE's.	No	Found that Spring boot has scheduling with which can schedule cron jobs and stuff	Create Boilerplate for Java backend with H2 database	No	Checked about admin refunds process	Design the classes required to implement	No
04/16	Handson Redux wiring	Implement CRUD for Transaction and perform unit test	No	Workspace setup completed.	Getting familiarized with AWS Services. Start implementation of the customer service API	No	Boilerplate for backend created and pushed to github	add the changes to save transaction for a specific account.	No	Started schema for refund requests	Implement service and controllers for admin transaction	No
									No			
04/19	Implemented CRUD for Transaction and Unit Testing for the same	Write Junit testcase for CRUD Transaction	No	Added Code for - Starting Bank Account for Specific Customers using Factory Pattern	Add unit tests, will create elasticbeantalk, mysql	No	CRUD changes completed	Added Swagger UI to test the backend API from swagger	No	Done with saving request refund and raise refund services	Will add more functionalities like open and closed requests	No
04/20	Completed Junit Testcase for CRUD Transaction	Query Check for view transaction	No	created AWS MySQL instance and added the connection configuration as test profile in properties file.Also, completed the unit test cases for all operations in Account Service.	AWS Cognito Integration for user authentication	Error while performing deployment. Security groups rules needs to be checked.	Swagger UI tested for API, working fine	Check the error while performing deployment in ELB	No	Updated open and closed requests	Add transaction debit and credit for admin	No
04/21	Figured out the query to check view transaction for 18months	Will be implementing back end for View Transaction and perform unit testing for the same	No	Completed login for both user types - AWS cognito	Working on customer dashboard	No	Inbound security groups added properly to unblock the error while deploying in ELB	Add unit test and setup client boilerplate	No	Added transactions for admin	Will test all api for admin and integrate code	No
04/22	Implemented backend for view transaction and did unit testing	Will be changing the backend code of view transaction as per the decided Date type and perform unit testing for the same	No	Changed the customer account creation to admin side, customer profile view with details created by banker. Account screen which lists account. Raising refund request UI and integrated with backend	Will work on adding externalPayee UI	No	Added unit test and did testing	check on how to schedule recurring tasks	No	Tested apis and pushed code to main	Integrate banker dashboard and integra	No
04/23	Implemented the changes as per the different Date type and did its unit testing	Will start the UI implmentation of Transaction	No	Add New External Payee UI	Refund Request UI Design	No	Found that scheduling can be done in spring boot using schedule annotation	Today I'll be setting up the server in aws for production	No	integrated banker dashboard	Implement UI for admin and will add more admin features	No
									No			
04/26	Implmented UI design for Internal transaction	Will implement UI design for External Transaction	No	Refund Request UI Design	View Transactions UI design	No	Did the server setup in AWS	Working on setting up external payees and scheduling the recurring transactions between accoounts and external payee	No	Implemented a admin features and UI	will work on documentation	No
04/27	Implemented UI design for external transaction	Will be doing Redux wiring for Internal Transaction	No	View Transactions UI Design completed	Date selection to view transaction restrict to last 18 months	No	Did the code changes for recurring setup for transaction between accounts.	Will work external Payees recurring.	No	Created presentation	Update ReadMe	No
04/28	Implmented Redux wiring for Internal Transaction	Will be doing Redux wiring for External Transaction	No	Date selection UI design completed	Backend Integration of View Transactions	No	Completed the changes for recurring transaction for external payees	Will start with one time transaction for external and internal payees	No			
04/29	Implmented Redux wiring for External Transaction	Will be doing end-to-end testing for internal and external transactions	No	Backend Integration of View Transactions	View Added External Payees Screen	No	Completd 1 time txn setup, external payee 1 time and recurring to be completed	Testing of cron job for recurring transactions	No			
04/30	Testing completed for Internal and External transaction	Testing of View Transaction on production	No	View External Payees Completed.	Close bank account, View transaction, Process Refund from Admin Dashboard	No	Recurring transactions wroking fine	End-to-End backend testing	No			
									No			
05/03	Completed View Transaction testing	Full end-to end testing on production	No	Close bank account, View transaction, Process Refund from Admin Dashboard - UI and Integration with backend completed	Backend Integration of External Payee	No	All test cases passed	Start with the project documentation	No			
05/04				Backend Integration of External Payee	Backend Integration of Refund Request	No	Created Class diagram with exact details	Update Readme	No			
05/05				Backend Integration of Refund Request	Backend Integration of View Transactions	No						
05/06				Backend Integration of View Transactions	Deploy jar in Elastic Beanstalk	No						
05/07				Performed deployment and verified the end-to-end working	Performed Testing	No						

XP CORE VALUES

Value	**Week**	**How team kept these values throughout Project**
Communication	Week 1	In team discussions and daily standups, we prioritized and valued comments from each team member on deciding the tech stack.
	Week 2	The UI wireframes were proposed by the team, and the comments from each team member was considered and the necessary changes were proposed.
	Week 3	As the team progressed with the user stories, we had demo within the team once a week and the feedback from each team member was take into account and necessary changes from each comments were implemented.
	Week 4	