Semester 1

Backlog Refinement Meeting: Sprint 1 - 31 March 2021

Banking system - Mobile app

Software/Frameworks:

Use any sequel database to infer relational
PostgreSQL/ MySQL

Database Tables:

Users
Roles
Keep the two tables above separate
Create accounts table – relate accounts to users

Create in database:

Create an admin user
Create 3 or 4 standard client users

Security:

Store passwords in some form of hash encryption – can use any hash encryption
Examples: sha256 or bencrypt

Validation:

Do validation on everything
Restrictions on email addresses, phone numbers
Validate ID

User Story 1

Basic user log in feature – 2 user roles
General client and admin user
If the user is in the database then log them in if their credentials are valid

User Story 2

Registration for normal client

Data to collect: First name, last name, phone number, email address, age, ID number, location, password

Pull geolocation API in later, use plain text for now

Keys for Login: ID and password

User Story 3

Registration for admin

Data to collect: First name, last name, phone number, email address, age, ID number, location, password, secret key (extra text field)

Has secret key attached - only one admin, at this point we don't care about validating the secret key

Secret Key

- hash any word, if someone registers with the key, allow the person to be an admin
- random selection of characters - pass through hash algorithm, it will generate a string that's unique to the word
- Validation check on key – is it valid? If valid, then complete registration process

Future improvement on secret key

- it always changes, will never be the same
- a table in DB that maintains list of active keys or inactive keys
- when admin registers, compare key the proposed admin entered to set of active keys in DB, if it matches then admin can log in
- Store encryption of encryption key in DB for security

User Story 4

Client Side

Show account details page

Before that page we need a verification pending/granted

User Story 5

Admin Side

Once logged in, see list of users that submitted registration for verification

User Story 6

Admin must be able to click on a user and see their account details

User Story 7

Upon viewing details, admin must be able to reject/accept user

User Story 8

Case of verified user:

- continue button to go to next page
- on next page display 3 account options: checking, foreign, savings

Detailed Explanations

- Be able to verify users – list of users, admin can click a button that says verify on each user, admin sees details submitted during registration and can verify the user

  Example:

  Admin logs in, successful login, see list of submitted registrations in a vertical list, be able to click on a user and see their details, at end of the list we want an accept/reject button, then user side must be able to pick up the decision

- Client side will either see pending, accepted/rejected
- Unverified user logs in, we display details they used to log in, at the bottom of page display status
- Difference between admin and client page:

  Same details
  Admin sees button to accept/reject
  Client sees status

- A specific account will have one or more users associated to it

Database:

Define specific rules per each account

Backlog Refinement Meeting: Sprint 2 - 9 April 2021

User stories for Sprint 2 according to Taiga - #13, #12, #14

Push user story 6, 7 and 8 into sprint backlog

Backlog Refinement/Planning: Sprint 2 Continued – 16 April 2021

User Story 9 - Account View Screen

If a user has an account when they log in, display the top half of the screen with the name of the customer and the name of the account, along with the amount of money in the account.

User Story 10 - Add account

If a user does not have an account when they log in, display their details and have a button at the bottom that says set up an account. That will take us to the page that will have the three bank account options. The user should then be able to create a new account.

Push user story 9 and 10 to sprint backlog

Detailed explanation

Don’t want to show customer details when they log in.

Instead, show users what accounts they have and how much money they have in the accounts - END GOAL USER HOME PAGE

If the user doesn’t currently have an active account then they need to go through the process of selecting different accounts

When we log in, check if the user has set up an account or not, on the home page if they don't have an account set up then we want to show them their details and have a button at the bottom that says set up account. That will take us to the page that will have the three bank account options.

The user can have 1,2 or 3 accounts but must be different. Can’t have two of the same account. In DB have a table (Bank account options) with bank account types - name, description (make these intuitive). We can add to it later. To the user table, add accounts associated with the user.

Add column to user table.

Nedbank has a nice structure to display account options. Look at the UI.

New user stories:

- Ability to add a new account (Cheque, savings, investment).
- Should be able to select from different types of accounts. Basic UI
- If users have an account when they log in then show them their first account in that list
- Design it in a way that the user can have multiple bank accounts but for now we work with a one account basis
- If user does have a bank account, render top half of screen with name and the name of the account, amount of money in the account - called the account view screen
- Think of a way to extract user account information. In the user table we have user details and their associated accounts. Have another table that keeps track of the relationship between a user and an account and the specific details relating to that, have some unique primary key that relates those two entities. We want to keep track of the amount of money in that account. Think of how to expand the table to include transaction history as well.
- For now have at most one type of account
- User account pair table
- Don't store associated accounts to users in a list in the users table. Create a separate table called user account pair, if a user is associated with an account then there will be a pair in that table. We also want to capture an amount of money in the account.
- Every time a user creates an account, add them to this new table and give them an arbitrary amount of money
- Add to backlog
- Display account screen to user in a clean UI
- How to structure an account screen. It will be the template for all the other screens that we show.
- Add a log table that keeps track of everything that happens in the system. will be big table

Sprint Backlog Refinement Meeting - 12 May 2021

Implement a tab navigator and move functionality to the tab navigator - 3 or 4 tabs

<u>Detailed explanation</u>

1: *Account Details Page*

(Profile Tab - remove create account or view account button)

2: *View Accounts*

Show all the accounts that the user has. Display them in the same way we have displayed a single account (the flip card effect).

Allow users to have a max of one of each account type. Maximum of 4 accounts

They can go on to view these accounts by clicking on them. Implement either a swipe or a region on the left of the card that takes the user to a separate account page.

When you choose a card, display card at the top with the same flip functionality and below that the transaction history associated to the account (payments paid and payments received - time stamp, payment reference number and person who made payment)

So if you pay someone, put the name of the recipient. In the case where someone pays you, it should show your name. Don't put your own name unless it's a transfer done between your own accounts.

Transactions must have a unique reference number but instead of displaying that to the user, we must display a reference name which should be compulsory for the user to enter (for readability). So the user enters a ref name but we also have to create a ref number.

Move the create account button to this screen.

3: *Timeline for the user/history feed*

Captures everything the user does in the app, including creation of accounts and transactions made by the user (Transfers, payments to someone else, payments received and creation of accounts). Can be in list order.

Capture the function the user takes and the time.

Example:

If a user creates a savings account it will read:

User X created a savings account   Time Stamp

4: *Transact feature*

The user can transfer money between accounts or pay someone arbitrarily which is then shown as a deduction from the account. Capture that as a transaction and display that to the user.

Two types (displayed as buttons)

- Transfers which happen between the users own accounts
- Payments which happen between different users

To capture: The client that the money came from, the money that was transferred and the reference number (which is specific and unique)

<u>Transact Tab:</u>

- Transfers

  Required inputs - specify amount to transfer, choose account you want to transfer from (drop down), select account you want to transfer to. Add an input text for the reference name.

  Confirm transaction button at the bottom of the page.

  Once confirmed, remove money from one account and add to the other account.

- Payments

  Required inputs - capture amount to be paid (input text), select which account to pay from (drop down), account number (must exist in our db; don't use arbitrary account number - be able to track payment)

When the user logs in, don’t check if they have an account or not. After log in, display the user timeline. If they have no accounts or history then they see a blank timeline.

Create table to maintain transactions between accounts and users

If possible, include the transfer section in this sprint. First complete user stories above.

Start testing this sprint. Test files will be checked.

Don't test secondary functions and libraries used or UI. Specify which files in repo to ignore and which lines of code within a file to ignore (to get high code coverage).

Have high code coverage. 90% + is great.

Rewrite user stories as being implemented through tab navigator

1. Timeline of user activity
2. View Accounts Page mapped to tab navigator. Display all accounts with flipped card features.
3. View Specific Bank Account. Max is 4 accounts, one per account. Transact button at bottom of page.
4. Transact tab - buttons for transfers and pay someone
5. View profile page integrated into tab navigator.

User Story 11 - View timeline

When a user logs into the app, they should be able to see their timeline which would contain a list of items tracking the user's activity in the app.

User Story 12

Move the view accounts page into a tab navigator. On this page, the user should be able to view all the accounts they have with the bank and they should be displayed using the flip card format.

User Story 13

The user should be able to view a specific account by clicking on one of the cards on the view accounts page.

User Story 14

There should be a transact option in the navigation tab that takes the user to a page with transact options that are available.

User Story 15

Move the view profile page into the tab navigator.

DB Table Draft

Accounts created	Customer Name	Account Type	Time Stamp					
Transfers			Time Stamp	Amount	Account from	Account to	Reference Name	Reference Number
Payments	Customer Name (being the payer)		Time Stamp	Amount	Account from	Account to	Reference Name	Reference Number

Sprint Backlog Refinement Meeting - 25 May 2021

Transfers (split into 3)

- Transfer between accounts. Enter your details of the account you want to transfer to and from, specify the amount. That becomes a transaction
- View transaction on both accounts and history feed

  Example: If you transfer from Business into Savings. Show the debit/credit process. Add this to the timeline as well. Amount taken out from business. Amount added to savings account.

Payments (split into 3)

- Be able to pay someone. A different account. Assume the client knows the person's details. Account number must exist in the database. Reflect payment on both accounts.

Latency Testing

- Each member creates three users. Play around with the app. Create transactions and populate tables with data. Check for latency between functions, such as with payments and transfers.
- All 8 of us are doing a transaction at once. Server should be able to handle 8 users so probably won't cause latency
- Form a test that allows you to test 50 to 100 users at once. Test process of transaction happening, such as payments or transfers. Can set up a python script to ping the database. Do this to test if queries are slow.
- Set up a script to ping DB. Trigger for loop with multiple calls to transfer as well as to payments. Record the times for these transactions. Check if the process slows down if too many users are using it at once or is it still happening instantaneously.
- We can all have the script on our laptops and run it at the same time. First iteration would be accessing the DB at once. A further step would be making the for loop run in parallel - will probably be a hard implementation but keep in the backlog / back of our minds as a potential way to test the DB.
- For user story:

  As a user, I don’t want to experience delays when processing transactions.

User Stories:

1. Make Transfer
2. View transfer records in transaction history (both accounts)
3. View transfer records in timeline(logs) history
4. Make Payment
5. View payment records in transaction history (both accounts)
6. View payment records in timeline(logs) history
7. Latency testing

<u>Semester 2</u>

Sprint Backlog Refinement Meeting - 18 August 2021

<u>Questions:</u>

1. Are we continuing with the mobile app or doing the web app version?

   Finish mobile app and move to web app.

2. Will sprint planning meetings always be on Wednesday going forward?

   Yes, unless decided otherwise by the team

Encrypt everything that happens in the app. Amount transferred, acc going to, acc coming from. References and names don't need to be encrypted.

Use bcrypt or argon. Pass in a text string we want to encrypt with a salt which encrypts the data. Then use a decrypter to be able to read and display data on the app.

Mobile App:

- User story 1 – encryption function
- User story 2 – decryption function

Then move to the web version.
If the backend is written well then just UI left.
Framework for web app – flutter, and biz.nf (this has been changed to absolute hosting).
Check that hosting is still live.

Web App:

- Take 4 user stories from sprint 4 and implement it into the web app.

Add to backlog:

Be able to generate pdf statements for a particular account. Be able to select the month you want to download a statement for. Limit to 3 months, etc. Make it look nice with the theme. Must show all the transactions of the account.

General Note: Useful to generate pdfs on websites that do payments etc.

User Stories:

1. Encryption function
2. Decryption function
3. Web - Login
4. Web - Customer Registration
5. Web - Admin Registration
6. Web - Display customer details

Sprint Backlog Refinement Meeting - 9 September 2021

Rylan – Sprint 2

Website – UI user stories

Web and app have separate user stories.

PDF Statements

Menu item that's a drop down list. Doesn't have to say statements. Can be general. Drop down list must have a statements option. Must be account specific. Be able to do it in every account. When they click on statements, take them to a new page/screen. Two criteria for months to display; either last 6 months if the user has interacted with the system, otherwise show the most recent months they've been with us and then cap it off at 6 months.Gives you a list of months, click on a particular month. We want that statement to generate a transaction history for the particular account the user is in or the month that they selected.

Have a download pdf aspect

Same as generating an invoice and populating with transaction history.

Two approaches:

First approach is more general so that the user doesn't need to go into an account to get a statement.

Tab navigation:

Select what account you want to download the statement for. Can go the route of adding another item to tab nav that's like a general item or settings item. User clicks on it and we provide them with a drop down list which only has statements for now but we'll add to that later. They click on statements and it must show them their list of accounts and after they select an account, they must then select the month they want. Each item in those months will be a transaction.

Approach where being in a particular account/card, we can include a button like a hovering button. If the user clicks on it then the same process follows. In this case we won't select what account we want to select the statement for.

Alternative:

Be able to download a pdf based on the account that they selected or account that they are in and the month that they selected. Take those transactions and display them. Then automatically generate the data and simply put it into the pdf.

Display balance of account at start and end of month like opening and closing balance.

Have a nice header at the start of the pdf and banks logo and colour scheme. Neat format with relevant information. Have client name, address, etc.

User stories for PDF Statements to be added to backlog:

1. Web app – Statements button on specific account
2. Web app - Statements page
3. Web app - Generate pdf using acc and month
4. Web app - Download pdf

Sprint Backlog Refinement Meeting - 29 September 2021

User stories to be added to backlog for app:

1. Mobile app – statements button on specific account
2. Mobile app - Statements page
3. Mobile app - Generate pdf using acc and month
4. Mobile app - Download pdf

Sprint Backlog Refinement Meeting - 13 October 2021

Complete existing user stories and make required adjustments.

Add to backlog:

1. Web app - Adjust verification page