SP26 – Red Budget App

CS 4850 – Section 03 – Fall 2024

November 7th

Josh Wilmath Developer	Garrett Heffner Developer
Avi Reale Documentation	

Project Team

Name	Role	Cell Phone / Alt Email
Josh Wilmath	Developer	630-850-0565 jrwilmath@gmail.com
Garrett Heffner	Developer	678.894.6317 garretttheffner27@gmail.com
Avi Reale	Documentation	678.378.1507 akreale2000@gmail.com
Sharon Perry	Project Owner or Advisor	770.329.3895 Sperry46@kennesaw.edu

Development Document Outline: Budget App Project

1. Bank Account Integration with Plaid

Goal: Automatically pull financial data from the user's bank to reduce manual input.

Implementation:

- Integrate with Plaid's API
- Process requires a JSON file to authenticate user and client.
- Data flow involves client, server, Plaid, and the bank.
- Persistent token is generated after user connects to Plaid, allowing ongoing data requests.

Setup Steps:

- Set up Plaid account and configure authentication.
- Generate a link token, then connect to obtain access token.
- Deserialize transaction data from JSON files to display in the app.

2. UI Development with Flutter

Goal: Create an intuitive and responsive UI for cross-platform use.

Implementation:

- Built with Flutter widgets, enabling a modular and easily adjustable interface.
- Supports quick updates for efficient UI testing and iteration.
- Design includes a navigation bar, dynamic transaction lists, and interactive buttons.

Setup Steps:

- Install Flutter and set up the development environment.
- Use Flutter's widget library to create modular components.

3. JSON Data Handling

Goal: Retrieve and display transaction information from JSON files.

Implementation:

- Deserialize JSON response from Plaid API into app-readable classes.
- Flutter enables transformation of JSON objects into displayable widgets.
- UI displays key information such as category, date, and amount.

Setup Steps:

- Set up Plaid API to receive JSON data.
- Implement JSON deserialization in Flutter.
- Create widgets to dynamically display JSON-derived data.

4. Data Visualization with FL Chart

Goal: Provide insights into spending habits through interactive charts.

Implementation:

- Uses FL Chart package for customizable charts.
- Charts reflect budget targets, color-coded for over- or under-budget statuses.
- Plans to add additional graphs, such as spending over time and filter options.

Setup Steps:

- Install FL Chart package in Flutter project.
- Set up data classes for chart integration.
- Map JSON data to chart values, with dynamic updates for user spending analysis.

5. Database Integration with MS SQL

Goal: Store and manage user account information and budgeting data.

Implementation:

- Connect to Microsoft SQL Server using MS SQL connector.
- Create account functionality with username and password storage.
- Assigns a unique user ID as a primary key.
- Plan to expand with additional tables for budgeting and long-term transaction data.

Setup Steps:

- Configure MS SQL Server and enable remote access.
- Connect to SQL database through the MS SQL connector in Flutter.
- Design SQL queries to handle account creation, transaction storage, and data retrieval.

6. Version Control with GitHub

Goal: Track project changes and collaboration.

Implementation:

- GitHub repository for managing branches, dependencies, and modifications.

Setup Steps:

- Set up a GitHub repository and define branch structure.
- Use Git for tracking and pushing changes, ensuring team collaboration.

7. Security Considerations

Goal: Protect sensitive data within the app, especially during transactions.

Implementation:

- Currently using Plaid’s sandbox for testing, with future security protocols planned.
- Plans to store access tokens securely on the server and limit client-side exposure.

Setup Steps:

- Transition from sandbox to production API.
- Implement secure access token storage in database.

8. Future Enhancements

Goals:

- Add features like two-factor authentication, additional chart views, and transaction filtering.

- Notifications for budget thresholds, duplicate transactions, and spending habits.
- Expand platform testing on iOS and Android devices.

Setup Steps:

- Integrate two-factor authentication with database.
- Refine iOS testing setup, possibly with Xcode on macOS.
- Implement notifications based on budget and transaction events.

Transcription

So then one of the main things we wanted for the budget app was to be able to connect to a bank account to be able to pull in all of that financial data.

So now it's one of the common like pain points using a budget app is that all of the simpler ones you manually have to make every transaction in there and it just gets to a point where the user is too fatigued to keep up with it.

So we had a goal to add applied API integration and so the way this works is it's not quite as simple as you would think with a normal API where you just call the website and it just gives you a response.

So Plaid actually you need to give it a JSON file in order to authenticate who you are, who the client is.

And so here's my diagram so it's the client, the server, Plaid and the bank and so how it works is our app server doesn't specifically contact directly through the bank.

So we call Plaid, you generate a link between the clients and Plaid that will connect them to the bank and then that's able to give us back a persistent token that we can use to call Plaid to give us more transactions as they come.

And so how we develop that is so looking at the applied API there's a lot of different fields and a lot of information we can get from from Plaid.

And so when we call the API eventually that will be returned in a JSON file and so we need to be able to read that to display to the users.

And so to make that connection to Plaid we need to generate a link token and then you have to connect and then after that we get an access token that we use after that to get data following the connection.

And also this is still a work in progress with our app.

We still need to work out some of the areas and get that connected fully.

All right, and so for the UI development, so we mentioned we used Flutter and so that was a little bit new to our group, but I found Flutter was pretty intuitive to use so it's all based on these widgets.

And so you can see here we got widgets like this graph, we've got all these transaction entities you can scroll through we have buttons.

And so Flutter really makes that easy to use by you know letting you draw the widgets you can wrap them in other widgets.

And so, and it lets us also hot reload, so we can easily, you know, make this mobile app UI and go through and then test everything as we're developing.

And so you can see we focused on, you know, mobile environment, we have this navigator UI so after we log in here, we get to this homepage, which has this little navigation bar at the bottom which really works well on a mobile environment.

And we can switch between our screens we have these dynamic list of transactions we can scroll through which are pulled from JSON files.

And they're dynamic.

And then we can navigate through the screens, as we please, and going back and forth.

All right, and then so getting into the JSON storage and retrieval elements of our project.

So, as I mentioned before, how we're going to get the transaction info from Plaid is from that request it's going to send us this big JSON file and so we need to be able to deserialize turn that JSON file into a class that we can then read into our app.

And so it really works well with this app is that with flutter, we can get like this, you know, object of a transaction, and then flutter really easily allows us to make that object into a widget.

And you encode all of this UI data as you can see here, we can put the name of the category we can put the dates and we can put the amount, and we can, you know, line up all those widgets into this one list widgets.

And then scroll through all of them and so that will adjust dynamically as we change our file.

And so it really makes it easy to develop.

So you can see here.

So before.

So this is just our class for their transactions.

And you can see here's the data that we're reading from, and that goes directly into the app into these widgets that you can then scroll through and interact with.

And so, moving on to the element of our charts.

So we mentioned before we're using using this package FL chart.

And so that's a really handy package that lets us basically customize the charts, almost however we want.

They have line charts bar charts histograms all kinds of stuff.

And so how that works is that we can go through all of this data that we generated from the JSON file and from the objects that we can then read that into our classes.

Read that into the chart class and display it.

And so you see we got like functionality with FL charts, like we can tap on the charts and get the value from that we can also, I think in my display here that you can see so.

So the red and the green is dynamic based on how your spending matches up with that specific budget target.

So you can see for a grocery spending, we add up a big expense, it goes over and it's in the red, revert that back down under the targets in the green so it kind of gives you.

This is our homepage to give you like a heads up view of how your spending makes matches up with the categories that you created that grocery dining utilities transportation.

So I thought this was a neat view that it goes through adds up all of the transactions and it's able to just give you the view that you want to see how are you doing based on what you're spending on the budgets there.

And so also this is a work in progress to we want to use multiple different kinds of graphs to show you like different insights on your spending.

And so, I, we also still have some functionality like to show graphs of your spending over time, like how your habits have changed what aspects need to improve on.

And then also filtering down the transactions more so you can focus in on a certain area.

And like I said that's still still very much a work in progress at this point.

So the second big component is the database of our app, and you would think that connecting an SQL server through a TCP connection would be fairly straightforward.

But thankfully Microsoft and all of their infinite wisdom, their database server is so intuitive, it disables the browser that the server uses to discover itself.

So that had to be enabled remotely using their server manager.

And after that we used an API that Microsoft has.

Since Flutter doesn't have native database connection we use the MS SQL connector, which is given the IP that the database server is stored on and the port number that's used to access it.

After that, we tell it which database it wants to connect to in our case SP 26 red budget app you know super original name.

And on a page like the Create account page as you can see on our GIF, we type in a username and a password and when we click the Create account button, it will send a query formatted in SQL query format to create an account using that username and password.

Each of these accounts is given a user ID that's generated when the data is written into the table and that user ID acts as the primary key, which one of the big future plans is to have a second database that that primary key interacts with that stores some of the larger transactional information like long term budgeting amounts and progress towards those budgets and goals, so that we can store the information on two separate tables instead of just one table to minimize say users that aren't adding as much data into the database compared to users that are adding a lot of information in with different budgets.

And one of the larger concerns was security, thankfully SQL allows native TCP connections which adds in a layer of security that we don't really have to worry about managing the socket and that sort of connection information.

And you, for our version control we just use GitHub it's easy to see who's publishing what to which branch which pages they're modifying, so that we can see what dependencies are being modified and files like our pub spec yml file.

And if buttons are being added to say the Create account page the budget page the transaction page, and so on and so forth.

Finally, we have a lot of future goals we still need to interact with such as two factor authentication which will deal with the databases will have to store some sort of two factor methods such as an email or phone number to use.

And then, furthermore, expanding the connection that Plaid API has to enable adding more transactional information and even expanding Plaid to connect to the database API, and then, as I said before, expanding the database to have the transactional information and budgetary information in those long term transactions.

And then finally one of our big goals is notifications maybe we alert the user if their budgets going too low, or they're reaching their goal soon and they should start maybe saving a little more if they're able to.

And then, another transaction notification maybe if there's a transaction that's been noticed that's duplicated.

That sort of notification be super helpful.

And the goal of this app was to create a cross platform app, which has been a little difficult testing on the iOS side, because we can just connect an Android emulator, but iOS is a little more difficult because they want you to have the Xcode development app to connect to your Flutter app, which is a little annoying because none of us really have like Mac so that's a little bit of a pain.

Let's go Apple.

And then the tech stack was Flutter which uses its own built in programming language Dart so we didn't have to learn Swift so that's a win.

And we submitted our presentation before the due date.

Anybody have any questions?

I didn't hear much on security concerns.

What did you guys do?

Because it looked like from the architecture you were connecting to the Plaid API on the client side, which is really risky because that can be exploited.

Yeah, so yeah, let me chime in.

So, so far we're just connecting with the sandbox data, so I know it's kind of a work in progress to get everything connected.

I know with Plaid, they have specific security protocols for when you're signing up for you know their back end like dashboard, they would use to connect to all the API requests.

So right now it's just kind of in a tentative state, but so the idea is that longer term.

So basically I know the most secure part of the connection to the Plaid API is going to be this access token that you get.

And so I guess, so the idea is in long term, that would be stored on the database securely.

And then so that you could only, you know, that be associated with users and you can only access that, you know, through database calls and to be secure.

But yeah, so right now it's just sandbox data, nothing's real.

But yes, we do have, you know, ideas for long term security.

Flutter doesn't allow you to just plug in a USB port into your laptop and connect your iPhone.

And it wants Xcode, which we can install on our phones, but it needs Mac OS, at least from my understanding, to connect to the actual code that's on your laptop, rather than just connecting straight to your phone through the phone app.

But I gotta say for Android, it is pretty useful that it's really easy to install an emulator to run with Flutter and then you can see exactly what size it be.

And if you have an Android phone too, it's as simple as turning on the developer options and then just plugging it up to your computer and then you can run the app on the phone, and then also do the stateful hot reloads to refresh as you make changes to code and see it pretty much real time update.

Any more questions?

All right, thank you