Can Çermikli

User Stories Improvements

* A user story about how to create an account for a customer is not defined. This is necessary when a new customer profile is created, there will be no account created initially. Besides, an existing customer may want to create a new account too. So these processes have to be defined clearly.
* For the signup user story; it is not clear which fields are mandatory. Besides, for every field that needs a formatted value before sending to backend services, a validation process should be defined such as min/max character length, number/alphanumeric, etc.
* For monthly dashboard user story; it is not clear how the addition will be done in specific time interval when the customer’s accounts have different currencies.

Techical Questions

* ***How long did you spend on the coding test? What would you add to your solution if you had more time? If you didn’t spend much time on the coding test then use this as an opportunity to explain what you would add.***

I have spent about 6-7 hours to finish all the coding parts of the project that I am required to. If I had more time or want to make some additions to the project I may do the followings:

* 1. This is not defined in user stories but as I mentioned in the user stories improvements part, there needs to be an account creation process. So, I should add a screen to create an account to newly created user or an existing customer.

Besides, the following solution is depend on what the requirement is, but the account creation process can also be forwarded to backend server so that an account is automatically created and attached to a new customer.

* 1. *If customer is already logged in earlier, then its email address information can be remembered so that each time a customer open the app, s/he does not need to fill the email address field again and again.*
  2. *When a customer logged in, its session information will be stored in the app so that s/he can directly open monthly dashboard screen until session timeout occurs.*
  3. *While creating a customer profile, there can be error related to backend services such as data validation problems, network problems etc. So, when this kind of problem occurs, this information can be shown to user with a dialog or toast message.*
  4. *In the monthly dashboard/transaction list screen, a refresh feature can be added so that when user wants to see his/her latest transactions or statistics s/he can refresh them.*
  5. *There can be charts for each month’s total spent and earned values for all categories so that customer can see all the categories total values in a single graph. This graph can be a pie chart or bar chart.*
* ***What was the most useful feature that was added to the latest version of your chosen language? Please include a snippet of code that shows how you've used it.***

Actually it is not clear whether you mean Kotlin by this “your chosen language” or not, but I assume you state it as Kotlin.

In Kotlin 1.3.70, many new functions and classes are added for Kotlin collections. So, this is really nice. Previously while developing Android application with Kotlin, java.util.ArrayDeque (for stack and queue data structures) is used from the Java standard library. With this release, kotlin.collections.ArrayDeque implementation is included.

fun main() {

val deque = ArrayDeque(listOf(b, c, d))

deque.addFirst(a)

deque.addLast(e)

println(deque) // [a, b, c, d, e]

println(deque.first()) // a

println(deque.last()) // e

}

* ***How would you track down a performance issue in production? Have you ever had to do this?***

Almost every system have some performance problems in their environments, and it is really important to define these performance problems because it may lead huge problems for all the system. Generally, it may be difficult to find a performance issue at first, but if you know where to look and how to track down the issue then the process really gets simpler.

So, first of all, the source of performance problem should be defined; is it because of client application, network issues, backend services, database problems or other system problems. The process can change according to problem type, but to eliminate potential problematic parts, each of them should be investigated. **Client side problem** can be related to memory issues and there can be memory leaks, so memory usage should be monitored. **Backend services** may have memory leaks too, there can be wrongly implemented code parts such as infinite loops etc. So these problems can be identified both code review and memory usage monitoring. If it is related with **database queries**, those specific SQL should be monitored and analyzed. Yes, of course I have encountered these kind of problems and solved them.

* ***How would you debug issues related to API usage? Can you give us an example?***

First of all, the source of problem should be identified. And since we are currently

on mobile development side in this project, we have to look at our requests initially whether we send all related body, header fields correctly. If we know exactly which API endpoint has problem, the code part that prepares request should be debugged. Debugging can be done both placing debug points at real time or can print log to console so that what request is sent to endpoint and what response is got. If we make sure that our request is correctly prepared and sent, then we should contact backend API team so that there are some problems with their responses.

* ***How would you improve the Node server’s API that you just used? Please describe yourself using JSON.***

The following improvements can be made:

* 1. All endpoints needs to be called by authorized clients, but in this project all the endpoints can be called by everyone. So, when a customer opened the app and started a session with the backend servers, a session token can be generated all then the all remaining API calls should have that token in their headers.

Sample Header:

{

"outh\_token" : <token>

}

* 1. There can be a security mechanism added to the endpoints. Because a bad customer or anybody can interpret the API request/responses so that s/he can change them accordingly or replay it. For example, API call time, a nonce and a signature can be added to headers so that these values are validated by the backend services to identify correct clients.

Sample Header:

{

"call\_time" : <call\_time>,

"nonce" : <nonce>,

"signature": <signature>

}

* 1. Current transactions endpoint GET:http://localhost:3000/transactions queries all the transactions in the system which is not feasible in a real production environment. So, this endpoint should only return specific requested account’s transactions with paging.

Sample Request:

{

"account\_ids" : <[array of account ids]>,

"page" : <requested page>

}

Response:

{

"id": 2,

"account\_id": 1,

"amount": "7196.35",

"vendor": "Grimes, Quigley and Runolfsson",

"category": "Salary",

"date": "2019-03-17T00:04:00Z"

}

* 1. Current accounts endpoint GET:http://localhost:3000/accounts queries all the accounts in the system which is not feasible in a real production environment. So, this endpoint should only return specific requested customer’s accounts with paging.

Sample Request:

{

"customer\_id" : <customer id>,

"page" : <requested page>

}

Response:

{

"id": 2,

"customer\_id": 1,

"iban": "CR63 9723 6739 2728 6349 7",

"type": "Savings",

"date\_created": "1/14/2019",

"active": true

}

* 1. There can be a new endpoint which return total earned and spending values for specific month and customer. Current transaction endpoint returns all the transactions in the system, so this is a problem for both backend and client system.

GET:http://localhost:3000/total\_statistics

Sample Request:

{

"customer\_id" : <customer id>,

"month" : <requested mpnth>

}

Response:

{

"customer\_id": <customer id>,

"mont": <month>,

"total\_earned": <total earned>,

"total\_spent": <total spent>

}

* 1. Current customer endpoint POST: http://localhost:3000/customers lets to create a new user with same email address. Since in this system, email address is unique to a customer, a customer should not create a new account with an email address that is already used by another customer.
  2. Current transaction endpoint does not state whether that transaction is a source or destination type, so it may add this information to the response.

Response:

{

"id": 2,

"account\_id": 1,

"amount": "7196.35",

"vendor": "Grimes, Quigley and Runolfsson",

"category": "Salary",

"date": "2019-03-17T00:04:00Z",

**"type": "source"**

}

* 1. If a problem occurs while creating a new customer, the backend service can return related error message and code to customer so that s/he can contact to customer services with related message and code.

Response:

{

"error\_message": <returned error message>,

"error\_code": <returned error code>

}