|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Decorative | | | | |
|  |  |  | |  |
| Goldin Account Manager  Functional Specification | | |
| Goldin Baloyi | | baloyi.jabu@gmail.com |

Contents

[Introduction 3](#_Toc139572116)

[Architectures and Flows 4](#_Toc139572117)

[Authentication architecture 4](#_Toc139572118)

[Application Architecture 4](#_Toc139572119)

[Authentication flow 5](#_Toc139572120)

[Create account workflow. 6](#_Toc139572121)

[6](#_Toc139572122)

[Credit account workflow 7](#_Toc139572123)

[Debit account workflow 8](#_Toc139572124)

[8](#_Toc139572125)

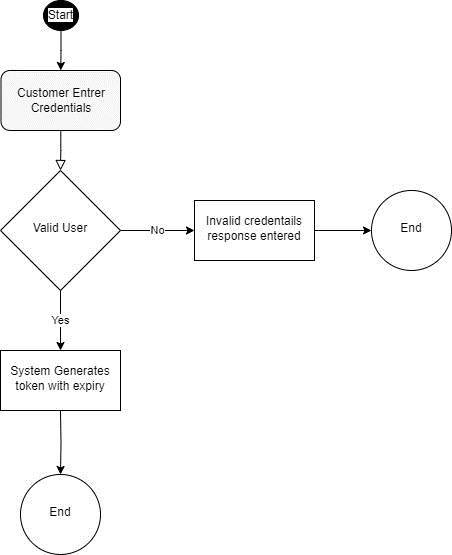
[Conclusion 9](#_Toc139572126)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Decorative | | | | |
|  |  |  | |  | |
|  | Introduction  This document contains all the information needed to use the Goldin Account Manager API Prototype. The API prototype uses JWT Authentication in order to access all account and transactional account details. This application uses a separate database to store the identity information and also uses an in memory database for account and transactional details. There’s also a Redis cache database which temporarily stores and syncs all accounts and transactions stores in the database for fast retrievals. | |  | |

|  |  |
| --- | --- |
|  |  |
| Architectures and FlowsAuthentication architecture  * User must be verified against a separate auth server and be issued an access token. * The token expires and in that case the user must re-login to get a new token. * To access the API prototype end point functionality, the token must be verified.    NB \*\* Only administrators of the API can create users.  \*\* For Demonstrations purposes the API can simulate creation of an API user. Application Architecture  * This is an example of a distributed caching environment. * The API prototype uses this behavior to temporary store large amounts of data which is constantly being used. * This lowers the amounts requests to the main MS SQL Database.   Easily Use Redis Cache In ASP.NET 6.0 Web API    This approach limits the number of times on which the MS SQL database is queried, the Redis storage can handle a lot of requests.  Redis — Beyond caching. Introductory blog on redis | by Amit Singh Rathore  | Dev Genius | |

## Authentication flow

1. Users enter credentials, which gets validated on the auth server.
2. If the credentials are valid, the auth server generates an auth token an expiry.
3. If the credentials are not valid, unauthorized exception is generated by the auth server.



**NB**

\*\*Should the token expire, please re-login to generate a new token.

\*\*If you do not have user credentials, please email [Baloyi.jabu@gmail.com](mailto:Baloyi.jabu@gmail.com) to get you started.

## Create account workflow.

1. The user must enter account/customer information, this information has the following conditions:

|  |  |  |  |
| --- | --- | --- | --- |
| **Key** | **Type** | **Required** | **Condition** |
| FirstName | String | Yes |  |
| LastName | String | Yes |  |
| Email | String | Yes | Valid email. |
| Telephone | String | No |  |
| IdentityNumber | String | Yes | Min 13, Max 13 |

1. The information is then checked against the production database using the identity number if the account details exist.
2. If everything is valid, a new account will be created.

## 

## NB

\*\*To access this functionality, you should be issued a valid access token.

\*\*Refer to authentication workflow on how to obtain an access token.

## Credit account workflow

**NB**

\*\* To access this functionality, you should be issued a valid access token.

\*\* The account must exist in the application before it gets credited.

\*\* The credit details get validated by a third-party system.

1. User enters credit information with an amount and an account ID.
   1. This can be a credit card or debit card.
   2. This can also be a bank EFT transfer.
2. The amount should be greater than zero.
3. The account must exist in the database.
4. If all is valid, a new transaction is loaded onto the account.
5. The balance of the account gets updated with the credit amount.

## Debit account workflow

## 

1. User enters debit information with an amount and an account ID.
2. The amount should be greater than zero.
3. The account must exist in the database.
4. The account must not have a zero balance.
5. The amount and the balance must not exceed zero.
6. If all is valid, a new transaction is loaded onto the account.
7. The balance of the account gets updated with the debited amount.

**NB**

\*\* To access this functionality, you should be issued a valid access token.

\*\* The account must exist in the application before it gets debited.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | |  | | |
| Conclusion There are a lot of ways to manage accounts in a real-world environment, this is just a prototype only to be used for demonstration purposes. No loss or injury obtained with the use of this application can be associated with the developer. Use at your own risk. | | | | |
|  |  | |  | |
| Reliable  This is a .net 7 application which is very reliable and also uses asynchronous programs with very limited bugs | Secure  With JWT tokens, the application is very secure | | High Performance  With the aid of Redis, this application can handle a very high transaction environment | |
|  | |  | |