**IGC.Plugin**

The IGC.plugin should work in a manner that it would connect all the member/users of the iHunt4 with the users in sub-databases and vice versa.

*Example,*

*A developer used IGC.plugin to create his gaming app called “Hit the Ball”,*

*A user of iHunt4 app downloads “Hit the Ball” app, and went to SignUp/LogIn screen, here the user shouldn’t have to SignUp as new player, he/she should be able to Login using the same credentials as iHunt4 app.*

Diagram

Description automatically generated

If one would observe above flow, when a new developer signs up for iHunt4 Developer account. A new sub-database is created inside iHunt4 DB. Onwards, **developer will be using his API-Key and API-Secret to make a call upon each and every Dynamic API Functions provided as following,**

1. **signUp\_to\_IGC**, this API will signUp new user directly into iHunt4 database, as well as in the sub-database of the app developer. In addition, **to call this API it requires to pass following values,**

{

“name”: “Eric”,

“email”: “eric@email.com”,

“password”: “123123123”,

“contact\_no”: “000001111”, 🡸 not mandatory

“gender”: “n/a”, 🡸 not mandatory

“date\_of\_birth”: “00-00-0000” 🡸 not mandatory

}

Providing correct values should return following outcome,

{

“status”: “200”,

“message”: “user registration successful”

}

If user already exists in the system,

{

“status”: “200”,

“message”: “user already registered”

}

1. **login\_to\_IGC**, calling this API requires passing of following credentials,

{

“email”: “eric@email.com”,

“password”: “123123123”

}

If the credentials are correct, the outcome should be as following,

{

“status”: 200,

“message”: “user found”,

“user\_id”: “22”,

“token”: “%$234wdsff1@”

}

Otherwise, if the user is not found or credentials are not provided correctly, the outcome should be as following,

{

“status”: 200,

“message”: “user not found”,

“user\_id”: “n/a”,

“token”: “n/a”

}

1. **get\_user\_profile,** to make a call upon this API function, passing of the following values is required,

{

“user\_id”: “22”,

“token”: “%$234wdsff1@”

}

If the user has not yet logged into his/her xumm account, it should return the outcome as following,

{

"status": 200,

"data": {

“id”: “22”,

“name”: “Eric”,

“email”: “eric@email.com”,

“contact\_no”: “000001111”,

“date\_of\_birth”: “00-00-0000”,

“gender”: “n/a”,

“delivery\_address”: “n/a”,

“xumm\_account”: “n/a”,

"assets": [],

“date\_created”: “2020-10-02 00:00:00”,

“date\_updated”: “n/a”,

“last\_login”: “2021-09-07 01:54:02”,

}

If the user has already logged into his/her xumm account, it should return the outcome as following,

{

"status": 200,

"data": {

“id”: “22”,

“name”: “Eric”,

“email”: “eric@email.com”,

“contact\_no”: “000001111”,

“date\_of\_birth”: “00-00-0000”,

“gender”: “n/a”,

“delivery\_address”: “n/a”,

“xumm\_account”: “rDpw131ffoine34eri4b123narqr43112a”,

"assets": [

{

"currency": "XRP",

"balance": "16.99",

"issuer": ""

},

{

"currency": "IGC",

"balance": "22",

"issuer": "raP6XZypcsSxw11WEaqFsdfWq24W89sDg"

}

},

“date\_created”: “2020-10-02 00:00:00”,

“date\_updated”: “n/a”,

“last\_login”: “2021-09-07 01:54:02”,

}

**Developers Guide For IGC.plugin Setup.**

1. A developer will go to [www.ihunt4.com/developersPortal](http://www.ihunt4.com/developersPortal)/register and signUp for new iHunt4-Developers-Account.

Graphical user interface

Description automatically generated

1. Developer will Login to [www.ihunt4.com/developersPortal/LogIn](http://www.ihunt4.com/developersPortal/LogIn)

Graphical user interface, text, application

Description automatically generated

1. Graphical user interface

   Description automatically generatedUpon Login new API-key and API-Secret will be generated for the developer on the dashboard.

***NOTE: Developers will have the option to go with traditional points-system or they can choose to use IGC point-system that would reward players as IGC per point.***

***1 point = 1 IGC***

1. If the developer is interested in granting users IGC tokens as rewards, he/she needs to “Login to IGC wallet”.

And upon successful login following UI will appear,

* IGC amount that developer holds in his/her wallet
* Buy IGC button - to top-up developers account with more IGC.
* Switch button - to switch to other IGC wallets.
* Settings button - that will take user to wallet settings screen.

Diagram

Description automatically generated with medium confidence

1. Upon click to the settings button, following screen will appear, holding following options,

* An option to put control of payment release.
* An option to put certain players on the black-list/blocked-list. Devs can add or remove certain players.
* A list that will show player waiting for their payments/rewards. Devs can add or remove certain players.
* An Instant Pay button that would open up “Instant pay screen”, allowing devs to make instant pay.

Graphical user interface, text, application

Description automatically generated

1. Following is Instant pay screen that can be accessed by the developers, when clicked on “Instant Pay” button in the main Wallet settings.

Diagram

Description automatically generated

Also, a developer will be able to access web page containing “API Docs”, that would provide developer with the functions to connect their app with their backend.

1. Developer will go to [www.github.com/treasureDev/igcplugin](http://www.github.com/treasureDev/igcplugin) and download the plugin for Unity games.
2. Plugin will get installed and would provide developer with ready to implement components and code.
3. Plugin will hold the following features,
   1. Demos.
   2. Xumm Login/Registration process.
   3. IGC/XRP exchange.
   4. Various dynamic APIs that would assist devs to create and manage their games.

