**Blockchain-Ads**
Blockchain Advertising platform
contact@blockchain-ads.com

# Developer's Project Document

**24th November 2023.**

## DEVELOPMENT TEAM

### Backend

The backend team's role is in ensuring the seamless functionality of the platform. It leverages Google Firebase Backend-as-a-Service offerings. The team's goal is to develop a robust API endpoint to facilitate user account management. Users can effortlessly create accounts, and during login, retrieve account information. Additionally, there would be endpoints for blockchain enabled payment transactions and user balance retrieval, providing a secure and efficient financial experience for the users. Furthermore, the backend team integrates the Propeller Ads API to enable users create new advertising campaigns directly from the platform. To enhance user insights on their created ads(campaigns), we retrieve metrics from the Propeller Ads via their API and expose it for the frontend to call and display to the user.

### Frontend

The frontend team is the face of the platform, creating a user-friendly experience irrespective of the development framework. The goal is clear: make interacting with the platform easy and enjoyable. From creating accounts to managing ads, intuitive interfaces for smooth navigation should be designed, with appropriate calls to the API exposed by the backend. The frontend should ensure the user's experience is seamless on any device, focusing on simplicity and visual appeal.

## Project Overview

- **API Creation & Connection:**

  Integrate user accounts, payments, and ad campaign features using backend APIs.

- **Metrics Display:**

  Fetch and display ad metrics on the frontend.

- **UI Optimization:**

  Ensure a user-friendly and responsive interface.

- **Testing:**

  Test features for a seamless experience.

- **Deployment:**

  Deploy the frontend, following provided instructions.

- **Feedback and Refinement:**

  Gather user feedback and refine as needed.

## System Architecture

The backend side of things is based on a B-a-a-S provider with other integrating parts.

- Server host: Firebase

- Authentication: **Firebase Auth**

- Database: **Firestore(Firebase)**

- Analytics Provider: **Firebase(Analytics)**

- Storage: **Firebase Storage**

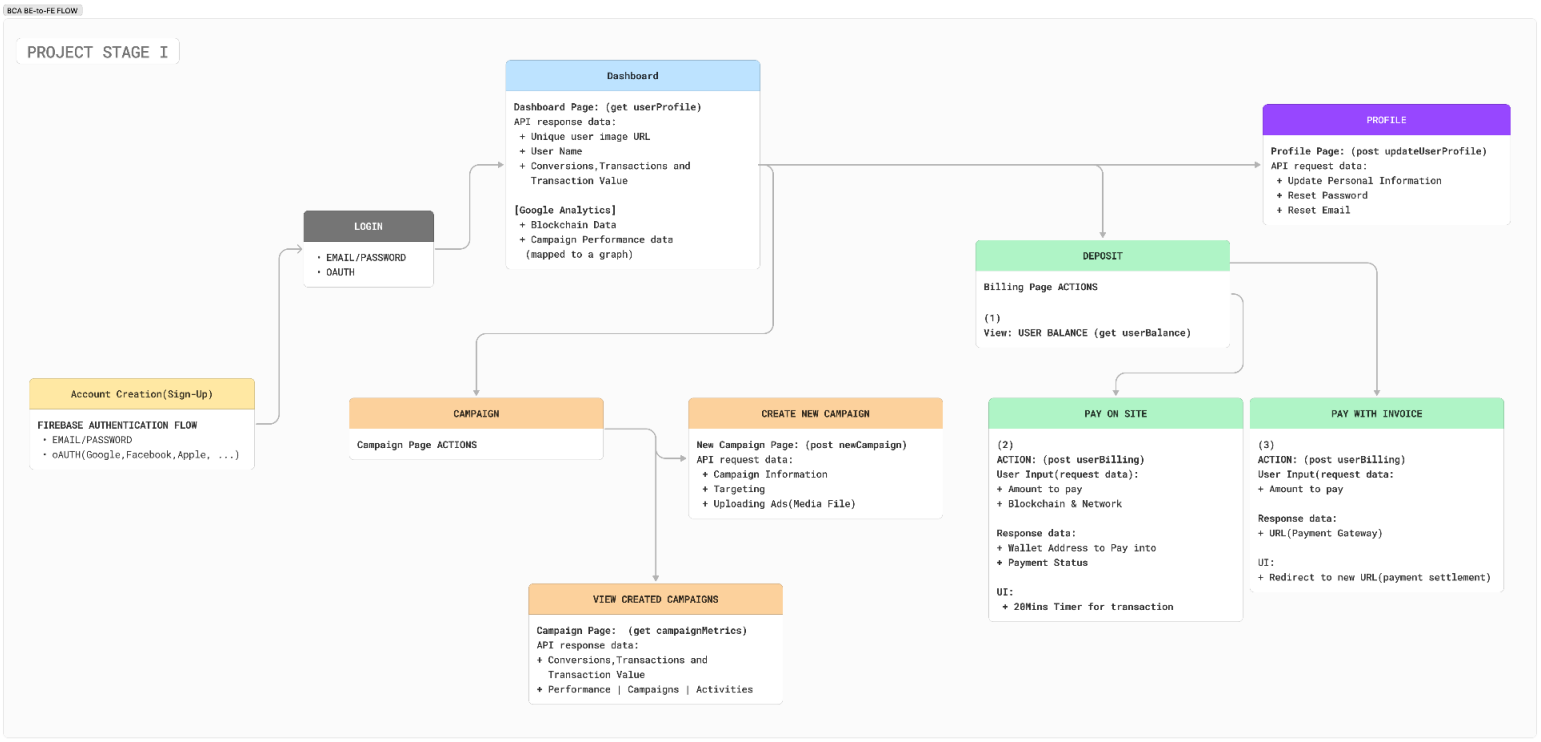
- Third Party Services API: [ **nowpayment** ]

- Language:  **NodeJS**

The backend architecture is built on a Backend-as-a-Service (BaaS) model, with Firebase serving as the central provider for core functionalities. The server is hosted on Firebase, leveraging Firebase Auth for user authentication. Data is stored in Firestore, and analytics tracking is facilitated through Firebase Analytics. For file storage needs, Firebase Storage is utilized. The backend is primarily developed in NodeJS, ensuring a scalable and efficient server-side infrastructure.

## Project Flow

The system flow of the application is visually represented in the accompanying diagram. This illustration outlines the journey of users through the application, showcasing key components and interactions. The diagram also clarifies how the frontend engages with the backend, highlighting integration points and major data flows(inwards and outwards).




**BCA PROJECT SYSTEM FLOW**

## API DOCUMENTATION

At this stage of the project, the backend development is just taking off. This means that rather than a fully defined API documentation, this is an iterative documentation that would change over the course of the backend development period. Once the backend development is completed, a properly structured documentation for each functioning part of the application would be well documented and provided. The documentation below, also highlights the crucial parts of the application and how they should work, giving the frontend team a window into how the application functions on technical grounds, and how collaboration and integration would happen over the course of development.

## FLOW DOCUMENTATION

### + ACCOUNT CREATION (SIGN UP)

The application sign up process relies on the Firebase authentication service. The application collects from the user during the account sign-up stage. It takes in the user's name, email address, a phone number(optional) and the user's password, to create an account. There are two ways to achieve this flow, either to take these information via a registration form and to then call an API endpoint exposed from the backend to take in that information and create an account for the user, or to use an OAUTH service sign in option from either Google, Facebook, Twitter, Apple, etcetera. Firebase documentation on how this works can be found **here**.

### + LOGIN

The login process works almost exactly the same way the sign up process does, but in this case, the entire process is handled on the frontend using the FIrebase web SDK(please take note of this during development). A form collects the user's email and password used during account creation and sends it to google to confirm if it exists, if it does, a successful response is sent back with the user's details, or a failed response is sent back. It works almost the same way with the OAUTH login. A youtube playlist on how to implement this on the frontend, can be found **here**.

### + DASHBOARD

Once a user successfully signs or logs in with the firebase SDK. The response data from Google contains identifying properties unique to each individual. That ID is then used to make a call to the backend, to call in the user's data, stored in Firestore. The response from that call is then used to populate the frontend. The video link above also contains implementation for how this works.

On the dashboard page, there are different sections to do different things offered by the product being built(Blockchain-Ads), that both consume and offer up data to the backend API for the user's experience.

**- Profile**

In this section of the application, the user has the ability to update their previously provided personal information. This is done by calling endpoints exposed on the backend, with the new information, with the user's personal identifying ID provided by the FIrebase AUTH SDK on page.

**- Campaign**

In the Campaign section, there are two actions that can happen, that relies on the backend to provide function, one is to start a new campaign, in the current test of the application **here**, there is a form flow that collects information from the user, and aggregates it in sections, with a summary at the end to show all the collected information, the information can then be posted to the backend. When this happens, the first thing that happens is to save the data to the backend, and then reach out to propellerAds via their API to submit the new campaign, the response from propellerAd is also saved in the DB(Firestore) and a response sent to the user on the status of the newly created campaign.

The other action regarding the campaign, is to view earlier created campaigns. This also makes a request to the backend to collect data previously stored in the DB for that user, and provide it to the user.

**- Deposit**

In the deposit section, the page should offer the user two options to make payments, the first being able to pay on the same page. In this case, the users put in how much they would be paying via a form, and also the blockchain and network they would pay with, this information is relayed to the backend, which in turn talks to **nowpayment** via their API to generate a wallet address for the user to pay into. The UI would also have to indicate to the user that they have a 20mins window to pay, or the transaction would be canceled.

The other option would be to pay with an invoice. This option also collects the amount to be paid, but that is all, it talks to the backend also, which communicates with nowpayment also, and a URL is created. The user should be redirected to the new URL which is a payment gateway controlled by nowpayment, with the option to pay with an address or a QR code, and on there, they can select which chain and network they would pay with.

**- Other Information**

Still on the dashboard, there is other information presented to the users, which is an aggregate of all their actions. There, they can see data pulled from the blockchain for the smart contract being advertised. Their advertisement performance, and the audience reached, all of this data being pulled from propellerAd.

## Integration and Testing Guidelines

As we enter the development stage, to allow us work seamlessly, we need to have a laid out outline for how we would collaborate and work. The easiest way to both work is to use Github, we both work on a repo, the repository would contain two distinct parts, the frontend section you would be working on, and the backend section I would be working on also.

As we work, I would also create a github gist file that would highlight in details the exact configurations for working with the APIs, the expected payload(Json), the expected responses and data, which you can implement into the frontend.

While at it, you can make reports on how the integration goes and if there are failures or bugs with comments on the github gist file and we would walk through fixing that as the project progresses.

### Testing

While building, real tests also have to happen outside our local machine and on the internet, to see and mitigate any problem that might be introduced when it is internet facing. I recommend that we use a free tiered version of Vercel for the test phase, for the frontend.

On the backend, I would be implementing directly on the Firebase  server for the project, with that, I'll be able to correctly provide the URL+endpoints that make up the API as we move forward.

## Conclusion

This comes to the conclusion of this document.

As stated earlier, this document is iterative and would improve as the need arises in all of the development phases.