

# Blink Documentation

A comprehensive step by step guide to setup Blink



To always get the updated and interactive documentation,  
please click the link below:

<https://blink-docs.imaginious.com/>

# Table of Contents

<b>Blink Documentation.....</b>	<b>0</b>
Table of Contents.....	1
Introduction.....	3
Top Features.....	3
Changelog.....	3
Version 1.0.0.....	4
Getting Started.....	4
Initial Setup.....	4
Unzip the Folder.....	4
Domain Setup.....	5
1. Purchase a Domain Name.....	5
2. Configure Domain Name System (DNS).....	5
3. Configure SSL (HTTPS).....	5
Code Editor Installation and Setup.....	6
GitHub Account Setup.....	6
1. Git Setup.....	6
2. Setup GitHub.....	6
NodeJS Installation.....	7
Frontend Setup.....	7
Package Installation.....	8
Firebase Setup for Frontend.....	8
Analytics Setup.....	12
Edit Text for Frontend View.....	14
Change Name and Tagline.....	15
Logo and Icon Setup.....	15
Change Logo.....	15
Change Favicon.....	16
Theme and Design Setup (Optional).....	17
Customize Color and Radius.....	17
Database Setup.....	17
Self Hosting.....	18
Managed MongoDB Service.....	18
Payment Gateway Setup.....	18
Stripe Payment Gateway Setup.....	19
Collect Stripe Connection Keys.....	19
Create Product on Stripe.....	19
Create Stripe Webhook.....	21
Backend Setup.....	22

Maxmind Geo-Location Service Setup.....	22
Firebase Admin Setup.....	24
Cloudinary Setup.....	25
Package Installation for Backend.....	26
Administrator Setup.....	26
Admin Setup.....	27
Moderator Setup.....	30
Add New Moderator.....	30
Hosting and Deployment.....	30
Backend Deployment.....	30
Managed NodeJS Hosting Using GitHub.....	31
Self-Managed Hosting.....	31
Frontend Deployment.....	31
Managed NextJS Hosting Using GitHub.....	32
Self-Managed Hosting.....	32
Conclusion.....	32

# Introduction

## Blink - SAAS Bio Link Platform

Build your own SAAS bio link platform.

Demo: <https://blink.imaginious.com>

## Top Features

- Developed using modern tech stack including NodeJS, React, NextJS, MongoDB etc.
- High performance, scalable, stateless software architecture for your future growth.
- Powerful admin and user dashboard.
- Limitless link in bio design flexibility.
- Powerful inhouse analytics system.
- Administrator team management system.
- Payment gateway with exclusive access to premium features.
- Built in Google Analytics, Facebook Pixel, TikTok Pixel, LinkedIn Insight, Microsoft Clarity etc. integration.
- Built in support request system.
- and many more...

Let's delve into the documentation.

This documentation provides step-by-step instructions, written in a clear and easy-to-understand manner. Simply follow the process outlined, and you should experience a seamless workflow.

## Changelog

Stay up-to-date with the latest improvements and fixes to Blink, our link-in-bio platform. This changelog details all updates, from new features and enhancements to bug fixes and performance improvements.

# Version 1.0.0

February, 2025

- Initial release

## Getting Started

Welcome to Blink documentation. Get started quickly with Blink, our powerful link-in-bio platform built on a modern, stateless architecture using Node.js, Express.js, React, Next.js, and MongoDB.

This guide will walk you through the initial setup and key concepts, enabling you to integrate and utilize Blink effectively. It covers both frontend, backend, and database configuration for the platform. You can configure the whole platform following the doc.

*We offer setup, installation, and deployment services for Blink. If you can't do this by yourself and require assistance, please contact us at [contact@imaginious.com](mailto:contact@imaginious.com) for details.*

## Initial Setup

This section guides you through the essential setup steps for working with Blink. We'll cover configuring your custom domain, setting up your preferred code editor for development, connecting to your GitHub repository, and ensuring you have the correct Node.js environment installed. Completing these steps will prepare you to effectively configure and deploy the SAAS platform.

## Unzip the Folder

Unzip the main\_files folder. Inside the main\_files folder, you'll get the documentation folder and source\_code folder. The documentation folder has the pdf format documentation and the source\_code folder has the blink-frontend and blink-backend folders. These folders are the main source code of this platform. The blink-frontend folder contains the frontend code, developed using ReactJS and NextJS. The blink-backend folder has the backend code, developed using NodeJS and ExpressJS.

# Domain Setup

## 1. Purchase a Domain Name

Choose a domain registrar (e.g., Namecheap, GoDaddy, Cloudflare, etc.). Search for your desired domain name and purchase it.

*Cloudflare offers the most affordable domain registration pricing with no hidden fees or inflated renewal costs. Free WHOIS privacy is included, and seamless integration with Cloudflare's DNS, CDN, and SSL services ensures top-tier security and performance. While other registrars like Namecheap and GoDaddy may advertise lower first-year prices, their renewal fees often increase significantly—sometimes by two to four times—in subsequent years. Learn more at the Cloudflare Domain Registrar.*

## 2. Configure Domain Name System (DNS)

The main domain will connect with your frontend, and we need a subdomain for connecting with the backend service. Follow your frontend and backend service providers instructions for connecting your domain with them. You'll generally get the domain connecting credentials after you deploy the frontend or backend on their servers, so follow these steps below after the deployment.

- Go to your domain registrar's dashboard.
- Find the DNS settings or "Manage DNS" section.
- Set up CNAME records for subdomain (e.g., subdomain.yourdomain.com)

## 3. Configure SSL (HTTPS)

Most of the frontend and backend deployment service providers offer free SSL / TLS service and it's automatically generated.

*Using a Content Delivery Network (CDN) for frontend assets is highly recommended. CDNs significantly improve website performance and reduce page load times. Cloudflare, for example, offers a free plan that includes CDN, DNS, and SSL services, along with protection against various types of attacks.*

# Code Editor Installation and Setup

A code editor is a tool used to write, edit, and manage code efficiently. Popular code editors include VS Code, Sublime Text, Atom, JetBrains IDEs etc. We use VS Code, you can use any of your favourite code editors.

Download Visual Studio Code for your Mac, Windows or Linux system:

<https://code.visualstudio.com/Download>

## GitHub Account Setup

GitHub is a web-based platform used for version control, collaboration, and code management. It allows developers to store, track, and manage their code using Git, a distributed version control system.

It provides features like repositories (repos) for code storage, branches for experimenting with changes, pull requests for reviewing code, and issues for tracking bugs and tasks. GitHub also integrates with CI/CD pipelines, project management tools, and security features to streamline software development.

We will use GitHub to store our frontend and backend code. But to upload code on github, first we need to download Git.

### 1. Git Setup

- Download Git from <https://git-scm.com/downloads>
- Install Git.

### 2. Setup GitHub

- Open a free GitHub account on <https://github.com/>
- In the future we will create 2 private repositories to manage frontend and backend.

*If you create public repositories on GitHub, it will make your code available to everyone, which is a massive security risk for you and it also violates terms of license.*

## NodeJS Installation

Node.js is a JavaScript runtime used for building applications. We need NodeJS to run our frontend and backend servers. Below is a step-by-step guide for installing Node.js on Mac, Windows, and Linux.

- Download the latest LTS Version of NodeJS from <https://nodejs.org/en>
- Install NodeJS
- (Optional) Check NodeJS version using any Terminal/Command Line Tool using this command `node -v`
- (Optional) Check NPM version using any Terminal/Command Line Tool using this command `npm -v`

## Frontend Setup

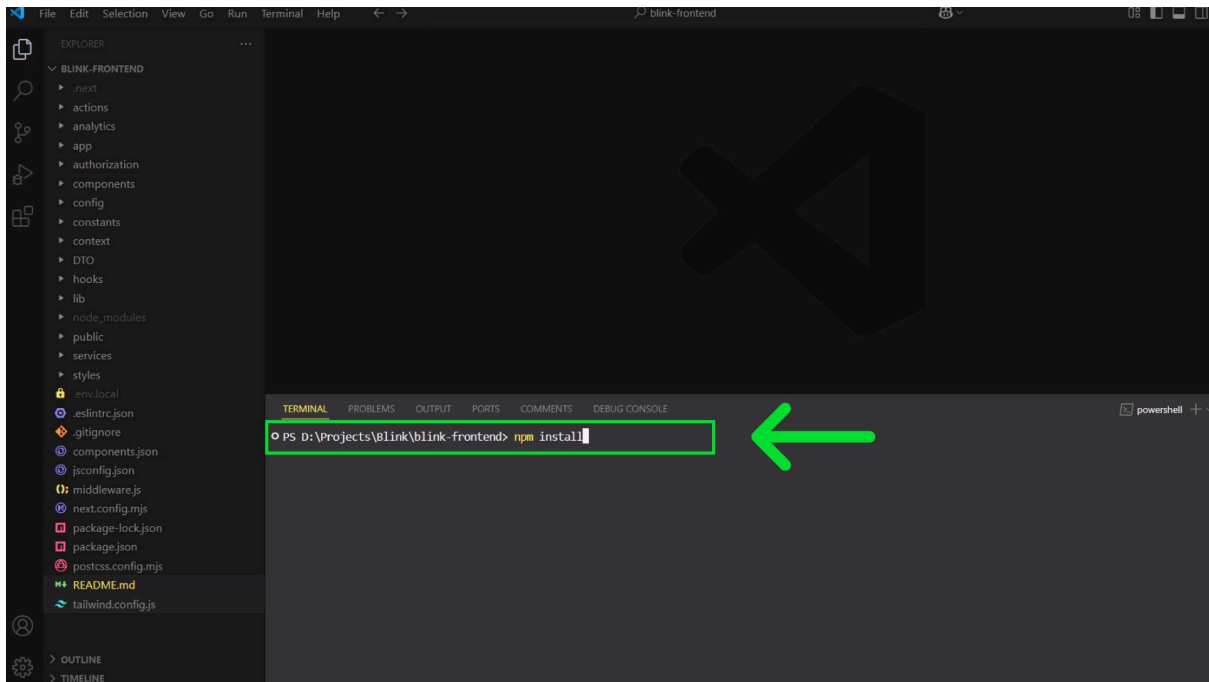
This section outlines the frontend setup process for our React and Next.js SaaS application. We'll cover essential steps including npm package installation, Firebase integration for authentication services, analytics setup for tracking user engagement, customizing the application's branding with logo and icon configuration, and applying the desired theme and design. Our stateless architecture ensures a predictable and scalable frontend experience.



# Package Installation

In this section we will install npm packages on our frontend project.

- Open frontend code in VS Code.
- Open terminal of VS code.
- Make sure the terminal is pointing to the root folder.
- Write this on the terminal `npm install`. Then press Enter.
- The above command will install npm packages. It might take a few minutes.

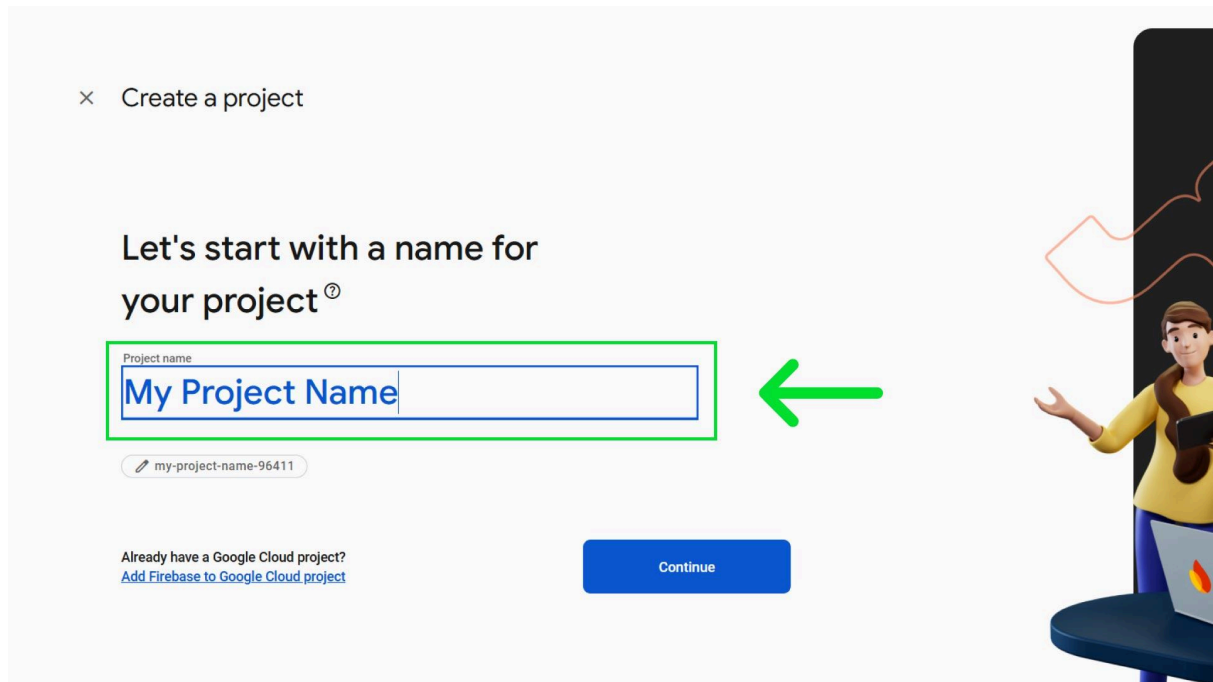


- Windows / Linux shortcut to open VS Code terminal `Ctrl+ ``.
- Mac shortcut to open VS Code terminal `Command + ``.

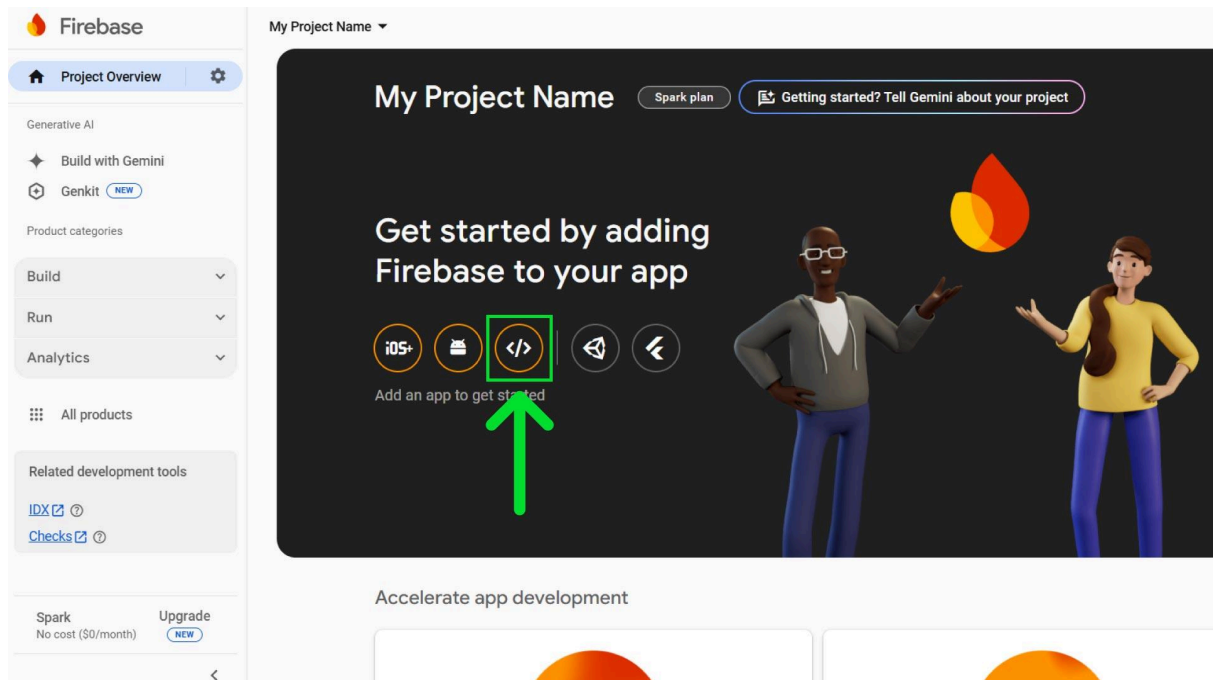
## Firebase Setup for Frontend

Let's setup Firebase for our frontend, later we will setup it for our backend also.

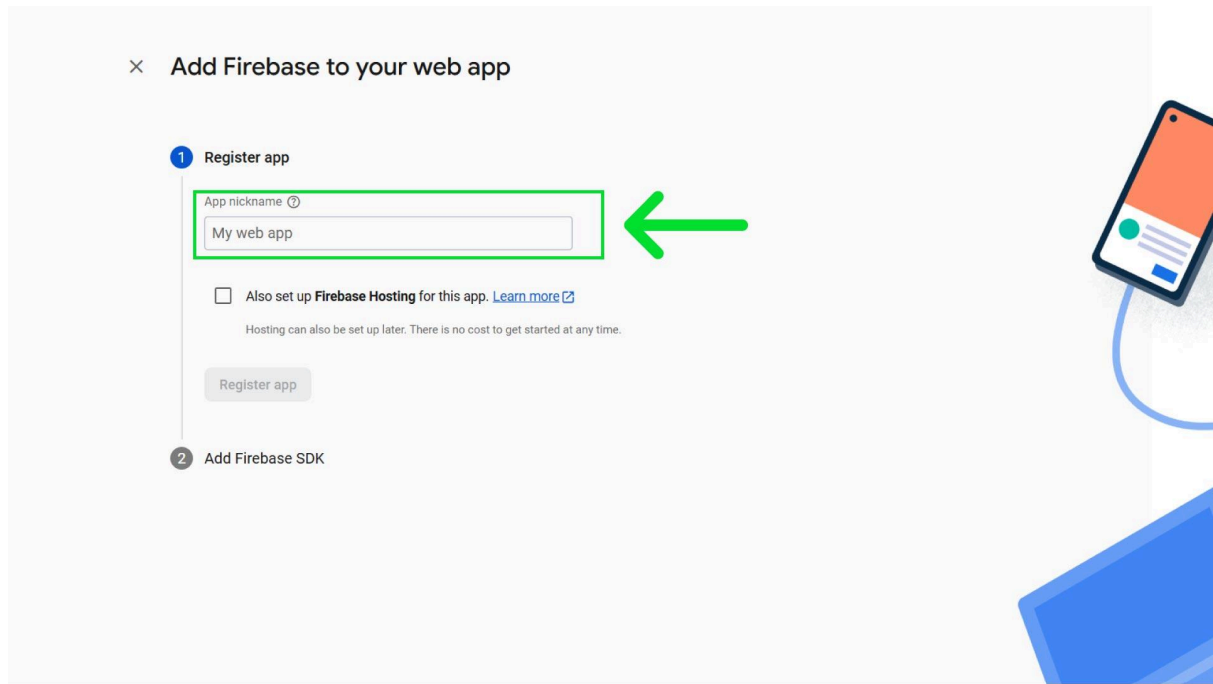
- Go to Firebase <https://firebase.google.com/>
- Click Go to Console
- Click create a project
- Then write your project name and press Continue button.



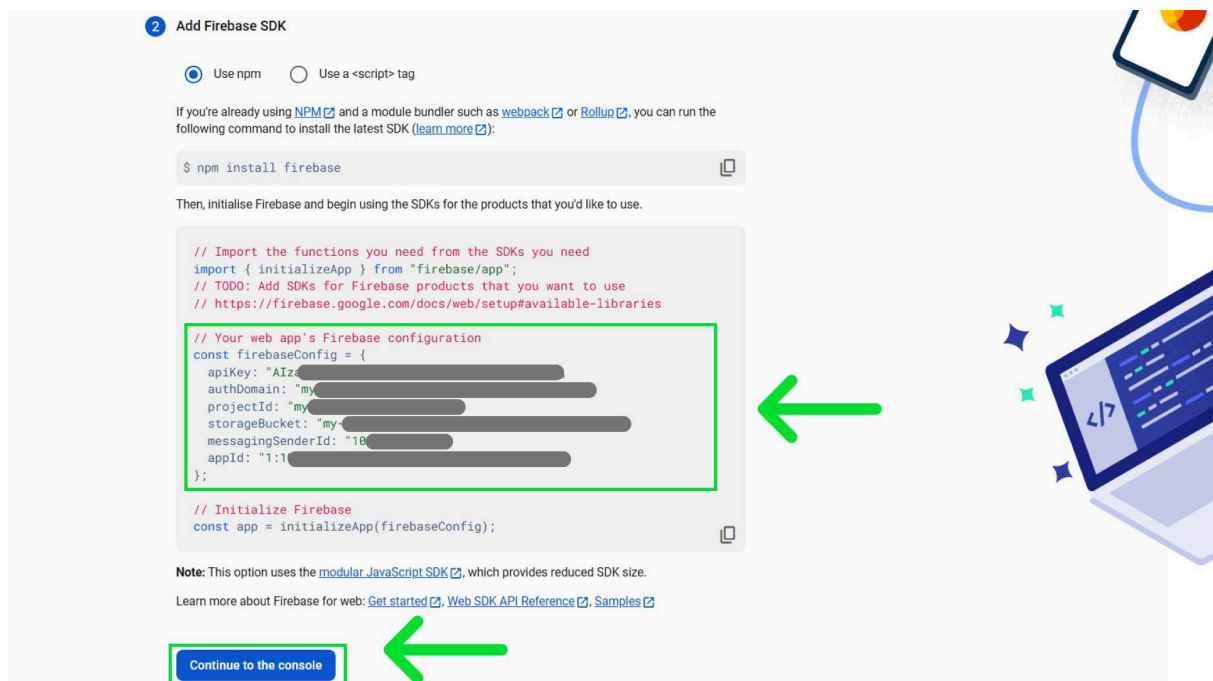
- AI assistance for your Firebase project is not necessary. Yet, if you want you can enable Gemini in Firebase option. Then press Continue.
- We will use Google Analytics separately, so you can disable it, and press Continue.
- Wait few moments and it will create your firebase project, then press Continue, it will take you to the firebase console dashboard.
- Press the Web button.



- It will open a popup, then write a nickname and click Register App.



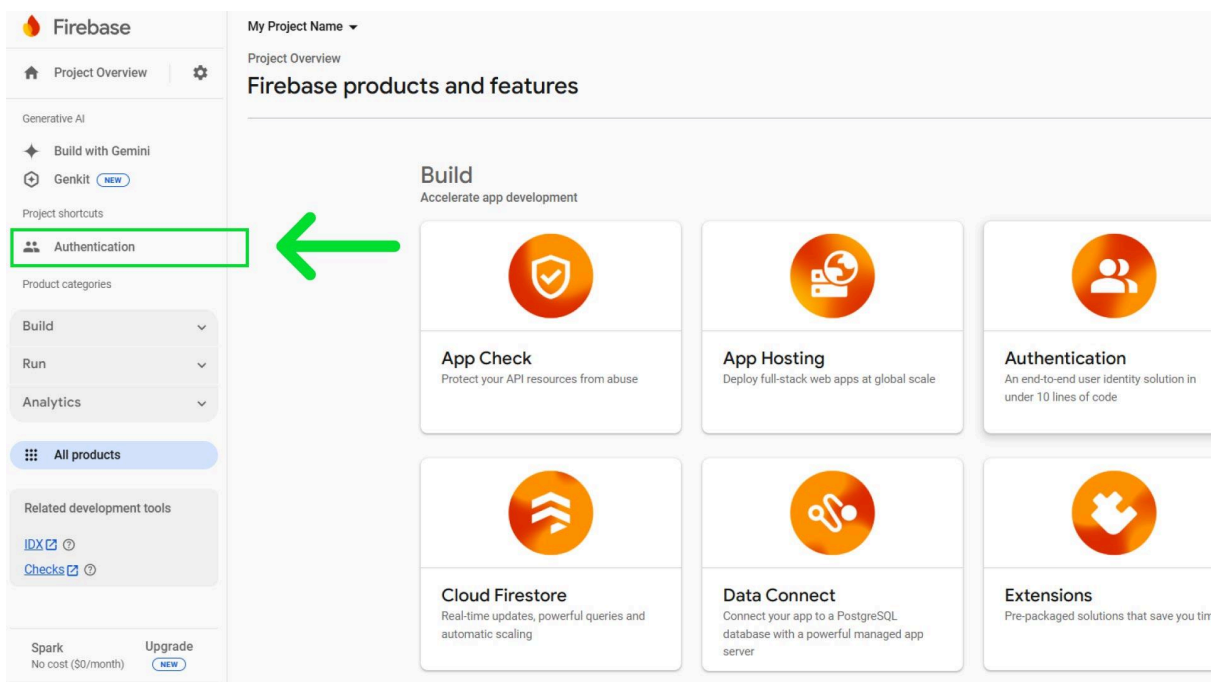
- Then it will provide your web app's Firebase configuration. Copy the `firebaseConfig` and save it in a secure place. Then press Continue to console.



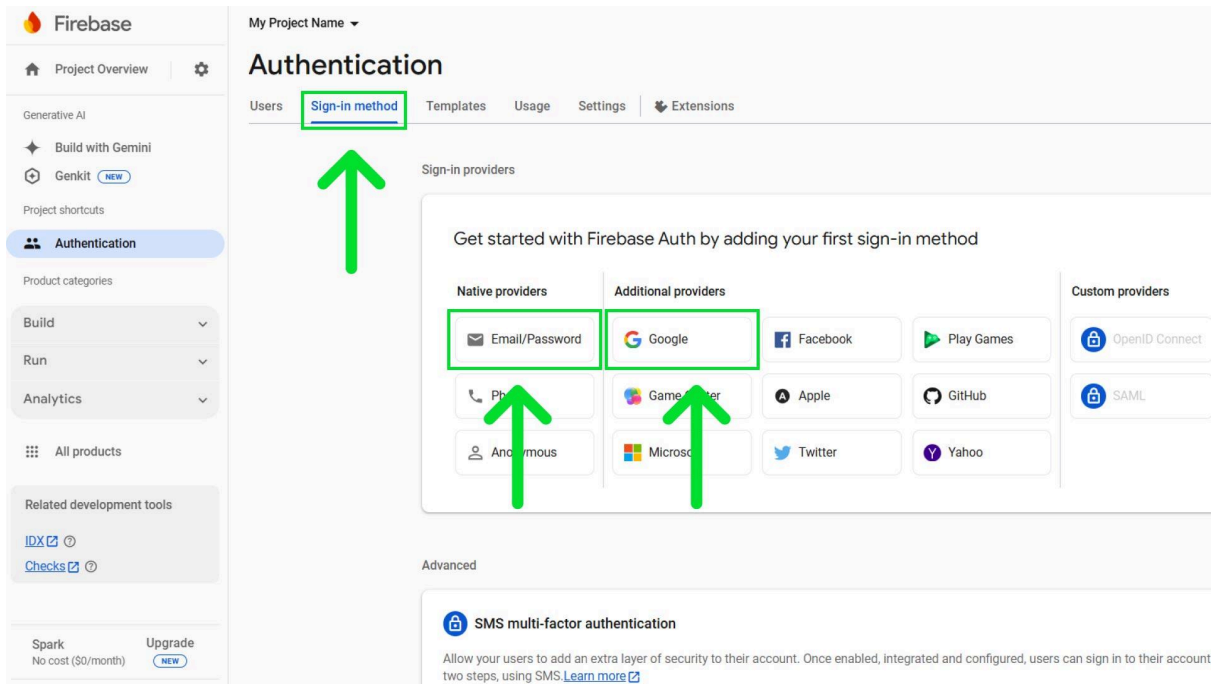
- Open your frontend code in VS Code and open `.env.local` file, and write the `firebaseConfig` values there and save it.

```
.env.local
1  NODE_ENV=development
2
3  # Firebase Config
4  NEXT_PUBLIC_FIREBASE_API_KEY=
5  NEXT_PUBLIC_FIREBASE_AUTH_DOMAIN=
6  NEXT_PUBLIC_FIREBASE_PROJECT_ID=
7  NEXT_PUBLIC_FIREBASE_STORAGE_BUCKET=
8  NEXT_PUBLIC_FIREBASE_MESSAGING_SENDER_ID=
9  NEXT_PUBLIC_FIREBASE_APP_ID=
10
11 # Backend
12 NEXT_PUBLIC_API_URL=http://localhost:8000/api
13
14 # Analytics
15 NEXT_PUBLIC_GOOGLE_ANALYTICS_GA_ID=
16 NEXT_PUBLIC_FACEBOOK_PIXEL_ID=
17 NEXT_PUBLIC_GOOGLE_ADSENSE_ID=
18 NEXT_PUBLIC_LINKEDIN_INSIGHT_ID=
19 NEXT_PUBLIC_MICROSOFT_CLARITY_ID=
20 NEXT_PUBLIC_TIKTOK_PIXEL_ID=
```

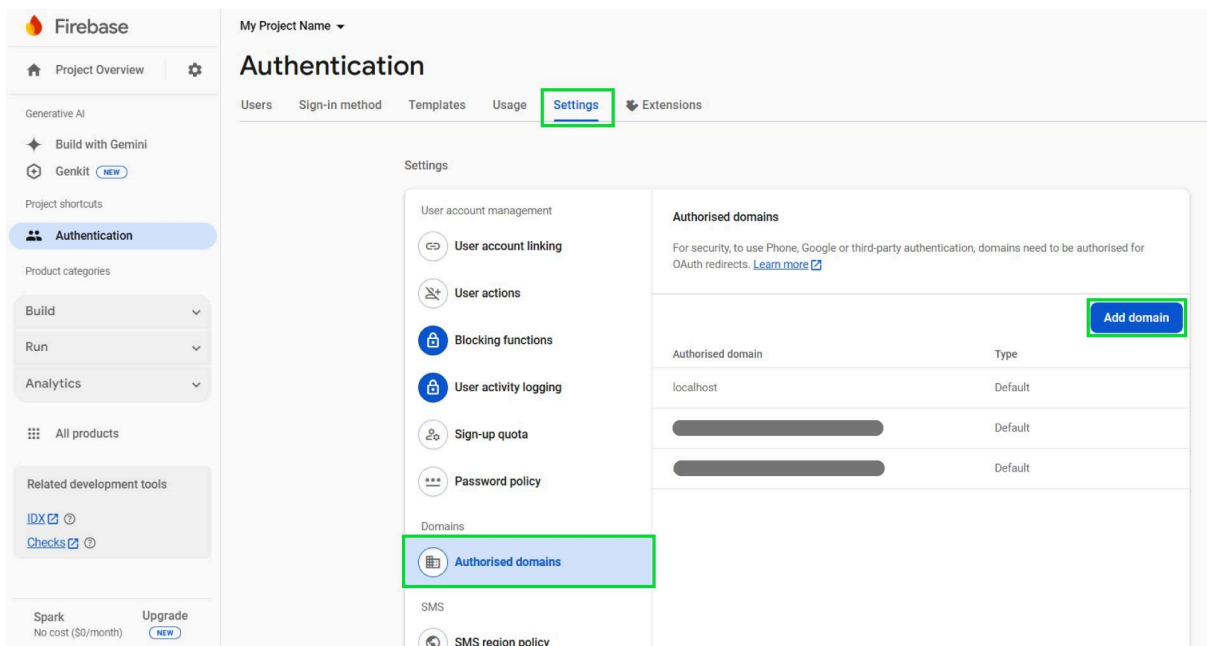
- Then go to All Products then select Authentication then click Get Started



- From sign in providers, click on Email/Password and enable Email/Password. Email link is not necessary. Then press Save.
- Now click add new provider button, then click on Google and enable it. Then if you want you can change Public-facing name. Then press Save.



- Then go to settings tab click on Authorised domains, don't forget to add your domain name here.



- That's it our frontend is ready with Firebase.

## Analytics Setup

This section is your comprehensive guide to integrating and managing analytics and tracking tools on Blink. Whether you're looking to monitor user behavior, optimize

marketing campaigns, or gain deeper insights into your audience we got you covered.

*Our platform seamlessly supports:*

- *Google Analytics*
- *Google AdSense*
- *Facebook Pixel*
- *Microsoft Clarity*
- *TikTok Pixel*
- *LinkedIn Insight*

Explore step-by-step instructions, best practices, and troubleshooting tips to make the most of these powerful tools and drive data-driven decisions for your business. Let's turn data into actionable insights!

- Get the tracking id for Google Analytics or any other platform motioned above.
- Open the front-end code, then head over to `.env.local` file. In the analytics section, write the id.



```
.env.local
1  NODE_ENV=development
2
3  # Firebase Config
4  NEXT_PUBLIC_FIREBASE_API_KEY=
5  NEXT_PUBLIC_FIREBASE_AUTH_DOMAIN=
6  NEXT_PUBLIC_FIREBASE_PROJECT_ID=
7  NEXT_PUBLIC_FIREBASE_STORAGE_BUCKET=
8  NEXT_PUBLIC_FIREBASE_MESSAGING_SENDER_ID=
9  NEXT_PUBLIC_FIREBASE_APP_ID=
10
11 # Backend
12 NEXT_PUBLIC_API_URL=http://localhost:8000/api
13
14 # Analytics
15 NEXT_PUBLIC_GOOGLE_ANALYTICS_GA_ID=
16 NEXT_PUBLIC_FACEBOOK_PIXEL_ID=
17 NEXT_PUBLIC_GOOGLE_ADSENSE_ID=
18 NEXT_PUBLIC_LINKEDIN_INSIGHT_ID=
19 NEXT_PUBLIC_MICROSOFT_CLARITY_ID=
20 NEXT_PUBLIC_TIKTOK_PIXEL_ID=
```

- Then go to analytics folder and open `analytics-integration.jsx` file. Uncomment any service that you want to integrate. For example if you want to implement Google Analytics, then uncomment `<GoogleAnalyticsIntegration />`.

```
analytics > analytics-integration.jsx > AnalyticsIntegration
1 import FacebookPixelIntegration from "@analytics/facebook-pixel";
2 import GoogleAdsenseIntegration from "@analytics/google-adsense";
3 import GoogleAnalyticsIntegration from "@analytics/google-analytics";
4 import LinkedInInsightIntegration from "@analytics/linkedin-insight";
5 import MicrosoftClarityIntegration from "@analytics/microsoft-clarity";
6 import TiktokPixelIntegration from "@analytics/tiktok-pixel";
7
8 export default function AnalyticsIntegration() {
9   return (
10     <>
11       {/* Google Analytics */}
12       {/* <GoogleAnalyticsIntegration /> */}
13       {/* Google Adsense */}
14       {/* <GoogleAdsenseIntegration /> */}
15       {/* Facebook Pixel */}
16       {/* <FacebookPixelIntegration /> */}
17       {/* LinkedIn Insight */}
18       {/* <LinkedInInsightIntegration /> */}
19       {/* Microsoft clarity */}
20       {/* <MicrosoftClarityIntegration /> */}
21       {/* Tiktok Pixel */}
22       {/* <TiktokPixelIntegration /> */}
23     </>
24   );
25 }
26
```

Example: We activated Google Analytics and Facebook Pixel  
/analytics/analytics-integration.jsx

```
{/* Google Analytics */}
<GoogleAnalyticsIntegration />
```

```
{/* Google Adsense */}
{/* <GoogleAdsenseIntegration /> */}
```

```
{/* Facebook Pixel */}
<FacebookPixelIntegration />
```

```
{/* LinkedIn Insight */}
{/* <LinkedInInsightIntegration /> */}
```

```
{/* Microsoft clarity */}
{/* <MicrosoftClarityIntegration /> */}
```

```
{/* Tiktok Pixel */}
{/* <TiktokPixelIntegration /> */}
```

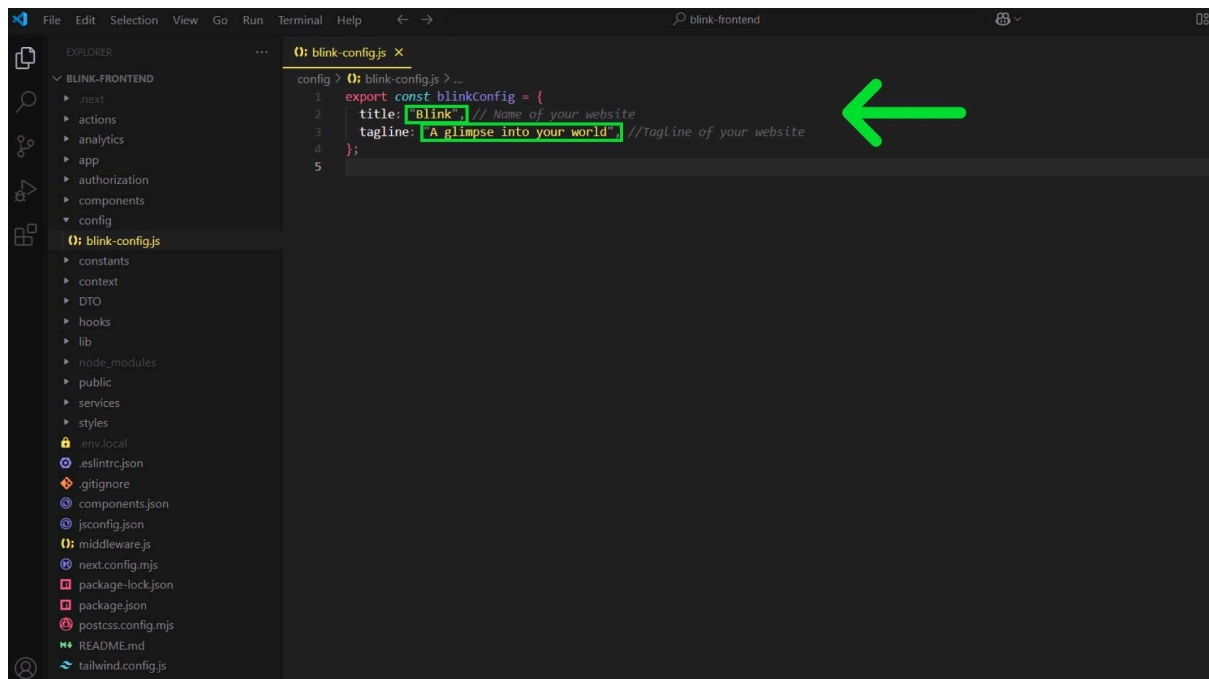
- That's it. Our analytics is ready to go.

## Edit Text for Frontend View

If you want to edit or change text of certain section, just head over to that section and change the text, every component and section has proper comments associated with them to help you edit the text.

## Change Name and Tagline

- Open your frontend code and go to config folder then open `blink-config.js` file. There change title and tagline of the website.



- FAQ section and some other places may contain the Blink name. Simply use VS Code's search feature and replace them.

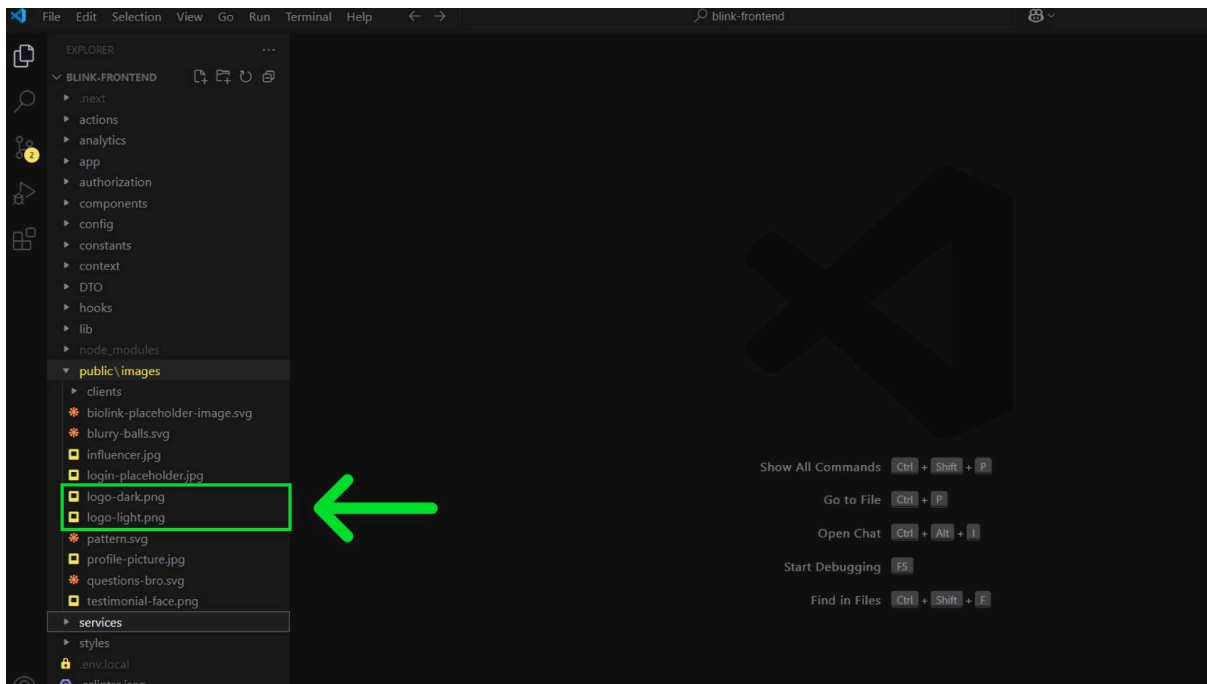
## Logo and Icon Setup

Logo and Favicon are two most important things in a web platform. We have used 2 different logo types in Blink, one for light background, one for dark background.

### Change Logo

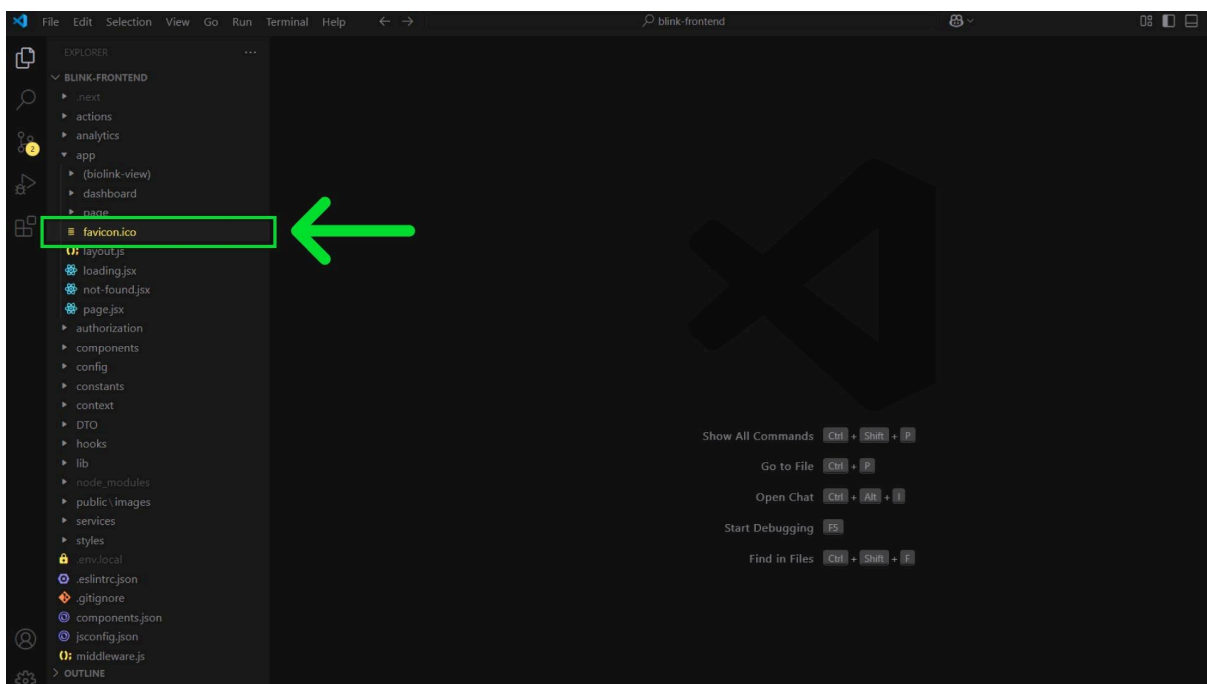
- Create 2 type of logo for your site. One should be light/bright color. Another one should be dark color. Export them in PNG format.
- Rename light colored logo as `logo-light.png` and dark color logo as `logo-dark.png`.
- Open your frontend code and go to `public\images` folder. There you will see `logo-light.png` and `logo-dark.png` file. Replace them with your logo files.





## Change Favicon

- First convert your logo to .ico format. You can easily do it using <https://convertio.co/png-ico/>
- Rename the file as favicon.ico.
- Open your frontend code and go to app folder. Replace the existing favicon.ico file with your favicon.



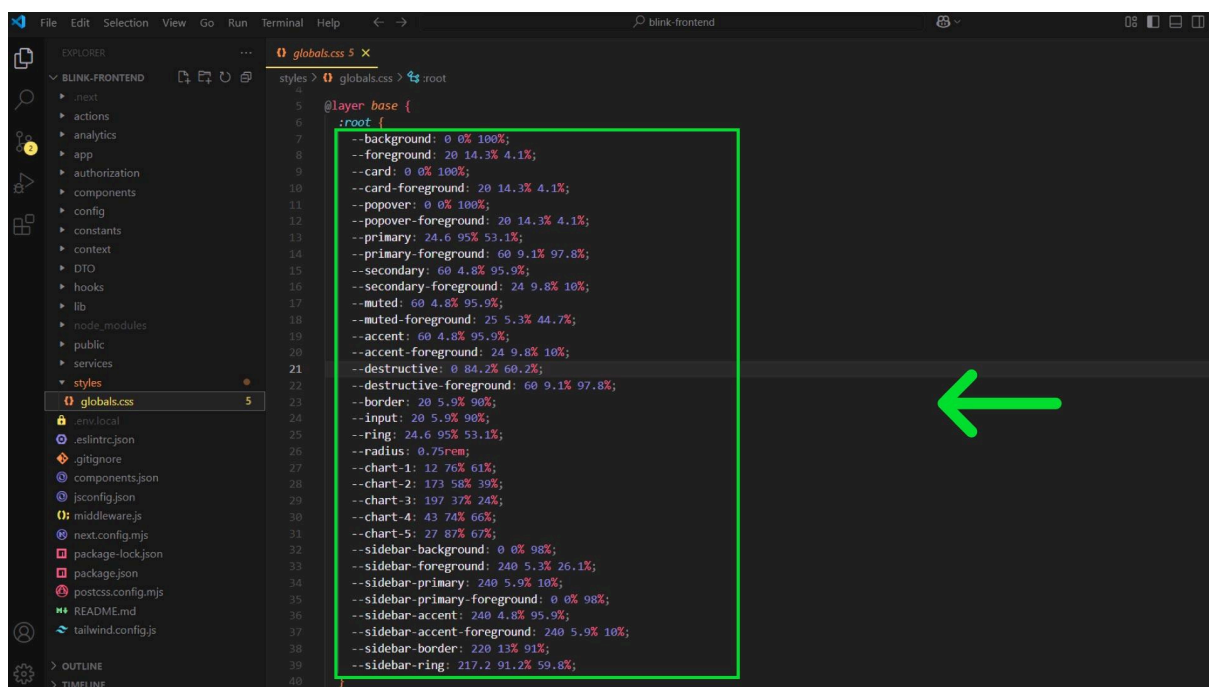
*Favicon is an icon associated with a particular website, typically displayed in the address bar of a browser accessing the site or next to the site name in a user's list of bookmarks.*

## Theme and Design Setup (Optional)

We used ShadcnUI to design and develop the Blink project. So blink is compatible with any ShadcnUI color theme.

## Customize Color and Radius

- Visit any ShadcnUI customization platform like <https://ui.shadcn.com/themes>
- Copy the code
- Open your frontend code and go to styles folder and open `globals.css` file.
- Replace the code there



## Database Setup

Blink uses MongoDB as database. It's a NoSQL database. Integrating with MongoDB requires only connection string. You can self host your database in the cloud or you can use any managed database service provider like MongoDB Atlas.

*Managed MongoDB service is the quick and easy way to get started with MongoDB. MongoDB Atlas provides 512MB of free database storage, which is enough for small workload. Microsoft Azure Cosmos DB for MongoDB provides 32GB of free database storage.*

- MongoDB Atlas: <https://www.mongodb.com/atlas>
- Microsoft Azure Cosmos DB for MongoDB:  
<https://learn.microsoft.com/en-us/azure/cosmos-db/mongodb/vcore/free-tier>

## Self Hosting

- Follow official MongoDB docs to self host the database:  
<https://www.mongodb.com/docs/manual/self-managed-deployments/>
- Get the connection string.
- Open your backend code in VS Code and open .env file, and write the MongoDB connection string there and save it.

## Managed MongoDB Service

- Get the MongoDB connection string from your database provider.
- Open your backend code in VS Code and open .env file, and write the MongoDB connection string there and save it.

MongoDB Connection String Example:

# MongoDB URI

MONGODB\_URI="mongodb+srv://\*\*\*\*\*/blink"

- How to get MongoDB Atlas Connection String:  
<https://www.mongodb.com/docs/guides/atlas/connection-string>
- How to get Azure Cosmos DB for MongoDB Connection String:  
<https://learn.microsoft.com/en-us/azure/cosmos-db/mongodb/connect-a-caccount#get-the-mongodb-connection-string-to-customize>

## Payment Gateway Setup

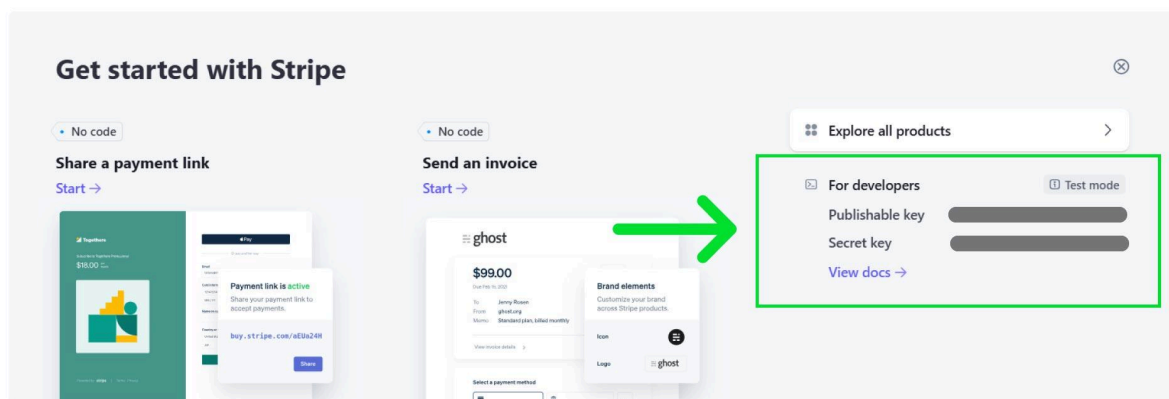
This section outlines the payment gateway setup process for Blink. Blink supports monthly and yearly subscription feature.

# Stripe Payment Gateway Setup

In this section we will discuss step by step process for integrating Stripe payment gateway in Blink.

## Collect Stripe Connection Keys

- Create a Stripe account at <https://dashboard.stripe.com/register>
- The it will redirect you to dashboard: <https://dashboard.stripe.com/test/dashboard>
- From the dashboard copy Publishable key and Secret key. Then save them in a secure place.



- Open the backend code in VS code and go to .env file, then place your secret key in STRIPE\_SECRET\_KEY variable and save it.

### # Stripe

```
STRIPE_SECRET_KEY="*****"
```

*By default Stripe comes with **Test mode**. You can use test mode to locally test stripe payments. To accept real money payments, follow [stripe business onboarding process](#).*

## Create Product on Stripe

- From dashboard go to Product catalog.
- Press on Create a product button.

*We have 2 paid products for subscription. There names are*

- Monthly price for Pro subscription is \$3.99 and yearly rate is \$39.00.*
- Monthly price for Premium subscription is \$7.99 and yearly rate is \$79.00.*
- You can price as you want.*

### Add a product

Close preview

×

Name (required)

Name of the product or service, visible to customers.

1

### Description

Appears at checkout, on the customer portal, and in quotes.

Image

Appears at checkout. JPEG, PNG, or WEBP under 2MB.

 Upload

More options 

Recurring

One-off

Amount (required)  USD 

Billing period

Cancel

Add product

## Preview

Estimate totals based on pricing model, unit quantity, and tax.

## Unit quantity

$$1 \times \$0.00 = \$0.00$$

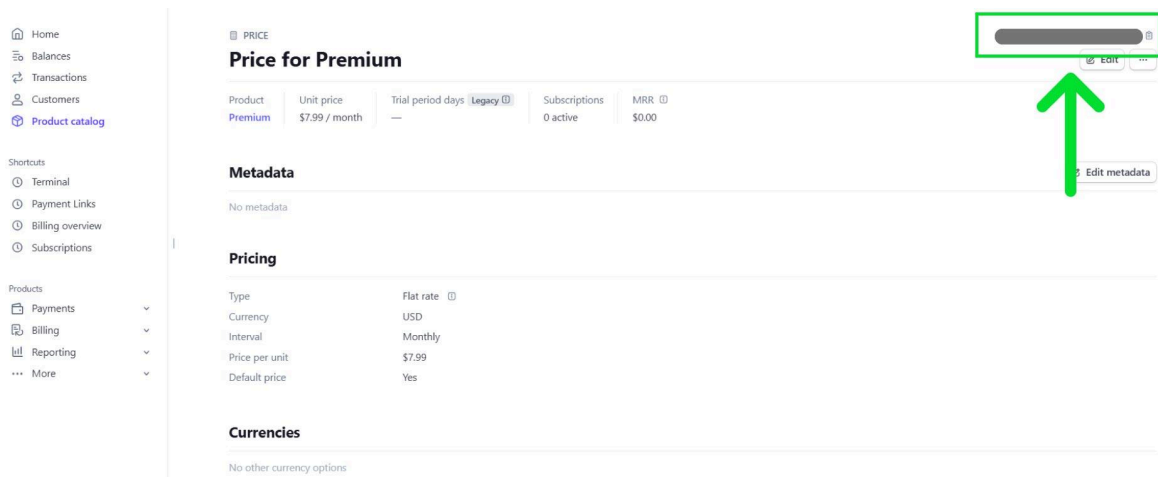
Subtotal	\$0.00
----------	--------

Tax [Start collecting tax](#)

Total per month	\$0.00
-----------------	--------

Billed at the start of the period

- For Pro product, provide name, description, amount, and select Recurring payment. Billing period should be Monthly, Unit quantity = 1. Save the product.
- Then collect and save the price\_id of monthly subscription.



- Then go to the same product again and add another price, This time for Yearly subscription. Again collect and save the price id of yearly subscription.
- For Premium product, repeat the same process again.
- Then open the backend code in VS code and go to .env file, then place your Price Ids in their respected fields.
- Save the .env file.

## # Stripe Price IDs

```
STRIPE_SUBSCRIPTION_PREMIUM_PLAN_YEARLY_PRICE_ID="*****
*****"
```

```
STRIPE_SUBSCRIPTION_PREMIUM_PLAN_MONTHLY_PRICE_ID="*****
*****"
```

```
STRIPE_SUBSCRIPTION_PRO_PLAN_YEARLY_PRICE_ID="*****
*****"
```

```
STRIPE_SUBSCRIPTION_PRO_PLAN_MONTHLY_PRICE_ID="*****
*****"
```

## Create Stripe Webhook

- From Dashboard's bottom-left corner click on Developers
- Then click on Webhooks tab and then click on Create an event destination.
- Then Add an endpoint

**Listen to Stripe events**

[Add an endpoint](#) [Test in a local environment](#)

Set up your webhook endpoint to receive live events from Stripe or [learn more about Webhooks](#).

**Endpoint URL**  
https://

**Description**  
An optional description of what this webhook endpoint is used for...

**Listen to**  
☒ Events on your account ☐ Events on Connected accounts

**Version**  
Your current version (2022-11-15)

**Select events to listen to**  
[+ Select events](#)

[Add endpoint](#) [Cancel](#)

**Sample endpoint** **Received events** **Node.js**

```

1 // server.js
2 //
3 // Use this sample code to handle webhook events in your integration.
4 //
5 // 1) Paste this code into a new file (server.js)
6 //
7 // 2) Install dependencies
8 //   npm install stripe
9 //   npm install express
10 //
11 // 3) Run the server on http://localhost:4242
12 //   node server.js
13
14 // The library needs to be configured with your account's secret key.
15 // Ensure the key is kept out of any version control system you might be using.
16 const stripe = require('stripe')('sk_test...');
17 const express = require('express');
18 const app = express();
19
20
21 // This is your Stripe CLI webhook secret for testing your endpoint locally.
22 const endpointSecret = 'whsec...';
23
24 app.post('/webhook', express.raw({type: 'application/json'}), (request, response) => {
25   const sig = request.headers['stripe-signature'];
26
27   let event;
28
29   try {
30     event = stripe.webhooks.constructEvent(request.body, sig, endpointSecret);
31   } catch (err) {
32     response.status(400).send("Webhook Error: " + err.message);
33   }
34
35   return;
36 }

```

- Your webhook Endpoint URL looks like <https://www.yourdomain.com/api/stripe/server-webhook> . Replace [www.yourdomain.com](https://www.yourdomain.com) with your domain name.
- Then write description if you want and select Listen to Events on your account.
- Then select all events
- Copy endpointSecret code.
- Click Add endpoint.
- Now open .env file from backend code, and add the endpointSecret value to STRIPE\_WEBHOOK\_SECRET and save the file.

## # Stripe

```

STRIPE_SECRET_KEY="*****"
STRIPE_WEBHOOK_SECRET="*****"

```

# Backend Setup

This section outlines the backend setup process. Our backend is developed using NodeJS and ExpressJS.

## Maxmind Geo-Location Service Setup

In this section we will discuss step by step process for integrating Maxmind Geolocation service in Blink.

- Create a free Maxmind account at <https://www.maxmind.com/en/geolite2/signup>
- Go to Manage License Keys section.

- Get Account id and License key from Maxmind.

## License Keys

If you have purchased web services, your license key may be used to [query them](#). If you have purchased GeoIP2 databases, your license key can be used to [automate the database downloads](#).

Generating a new license key will not disable existing keys. You can have a maximum of 100 active license keys at one time.

Before removing a license key from your account, please be sure to update your integration so that it is no longer in use. [Learn more about replacing your license keys on our knowledge base](#). You will not be able to re-activate a key once it has been removed.

We hash your license keys for security purposes, and as a result, we only display the first six characters of each key. This means that MaxMind does not have the ability to view your license keys in full. A license key is displayed, in full, only once to whomever generates it.

Account Summary

Account Information

Manage License Keys

Manage Account Services

Manage Users

Account Activity

Edit My Info

Sign-In Security

### Billing

Payment Method

Payment History

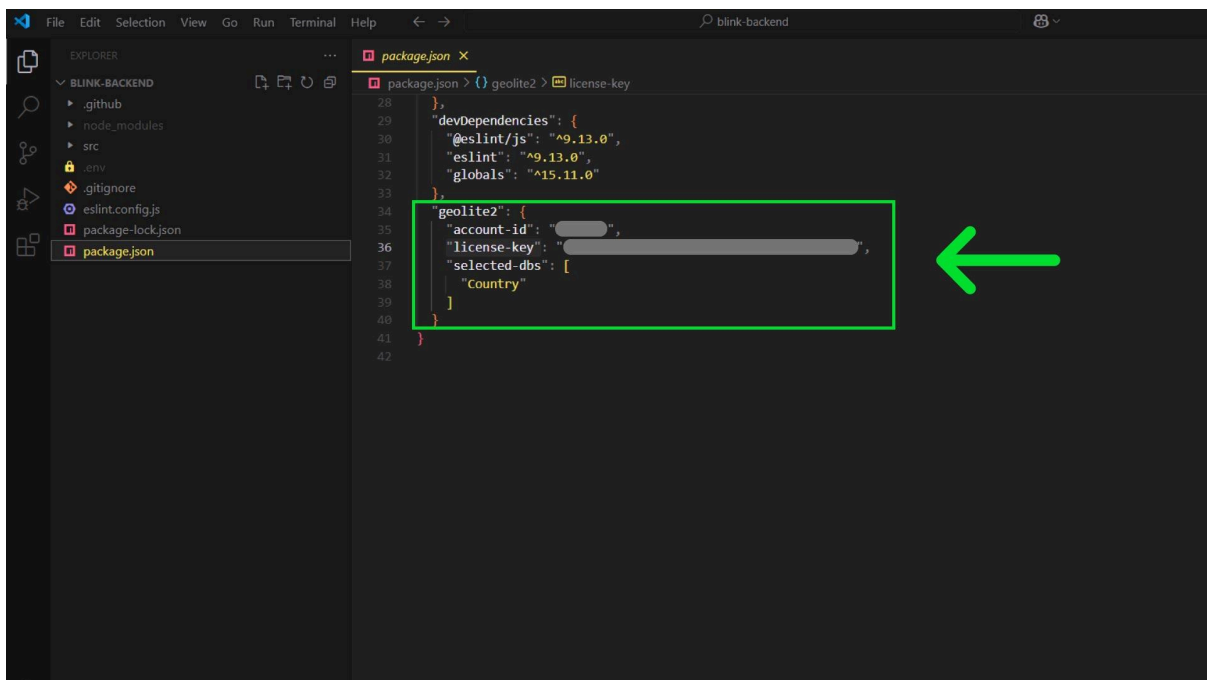
Purchase or Manage Databases

Account ID: [REDACTED]

Description	License key	Key created	Created by	Last used		
License Key #0	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]		

Generate new license key

- Open your backend code in VS Code and open `package.json` file. Write account id and license key in `geolite2` section's `"account-id"` and `"license-key"` respectively.



```
"geolite2": {
  "account-id": "*****",
  "license-key": "*****",
  "selected-dbs": [
    "Country"
  ]
}
```



- Then open the .env file and write the MAXMIND\_ACCOUNT\_ID and MAXMIND\_LICENSE\_KEY variables respectively.

# Maxmind

MAXMIND\_ACCOUNT\_ID="\*\*\*\*\*"

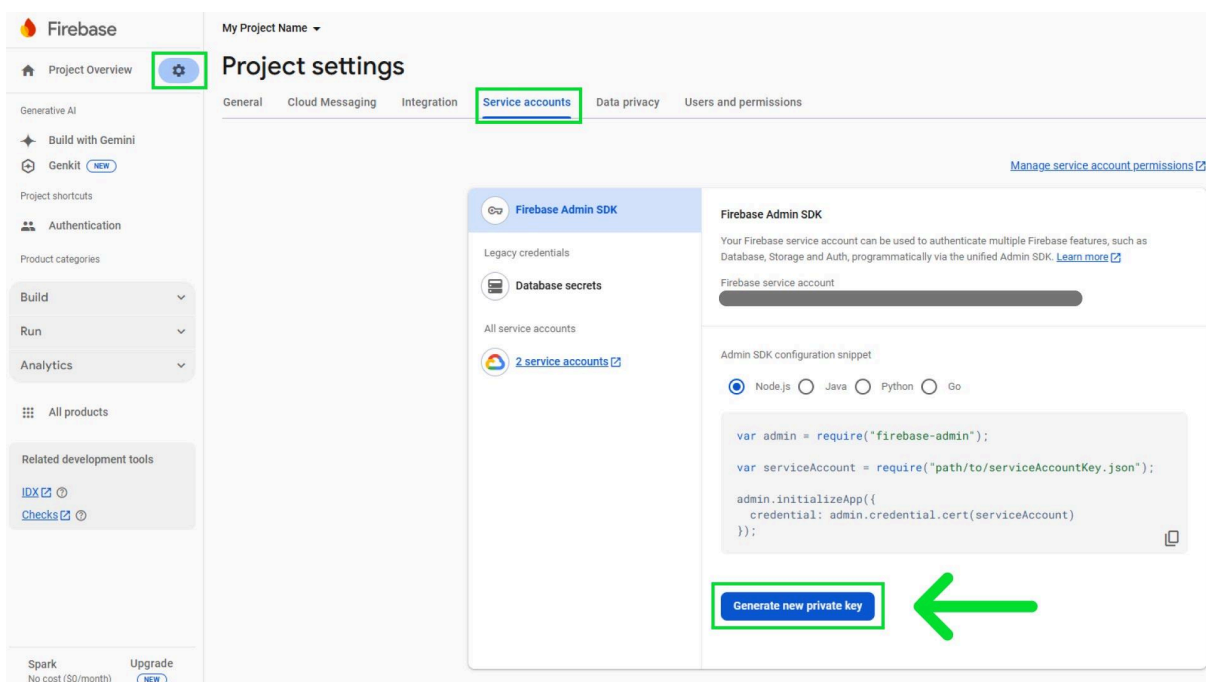
MAXMIND\_LICENSE\_KEY="\*\*\*\*\*"

*For better geolocation service consider buying Maxmind's premium plan.*

## Firestore Admin Setup

In this section we will connect our Backend with Firestore.

- Log into your firebase console then go to your project.
- In the Firebase console, open Settings then go to Service Accounts tab.
- Click Generate New Private Key, then confirm by clicking Generate Key.



- It will provide you with a JSON file. Securely store the JSON file containing the key.
- Then open the backend code in VS code and go to .env file, then go to Firestore Admin SDK section and then place your private keys in the respected fields.

# Firestore Admin SDK

FIREBASE\_ADMIN\_TYPE="\*\*\*\*\*"

FIREBASE\_ADMIN\_PROJECT\_ID="\*\*\*\*\*"

```

FIREBASE_ADMIN_PRIVATE_KEY_ID="*****"
FIREBASE_ADMIN_PRIVATE_KEY="*****"
FIREBASE_ADMIN_CLIENT_EMAIL="*****"
FIREBASE_ADMIN_CLIENT_ID="*****"
FIREBASE_ADMIN_AUTH_URI="*****"
FIREBASE_ADMIN_TOKEN_URI="*****"
FIREBASE_ADMIN_AUTH_PROVIDER_X509_CERT_URL="*****"
*****"
FIREBASE_ADMIN_CLIENT_X509_CERT_URL="*****"
*****"
FIREBASE_ADMIN_UNIVERSE_DOMAIN="*****"

```

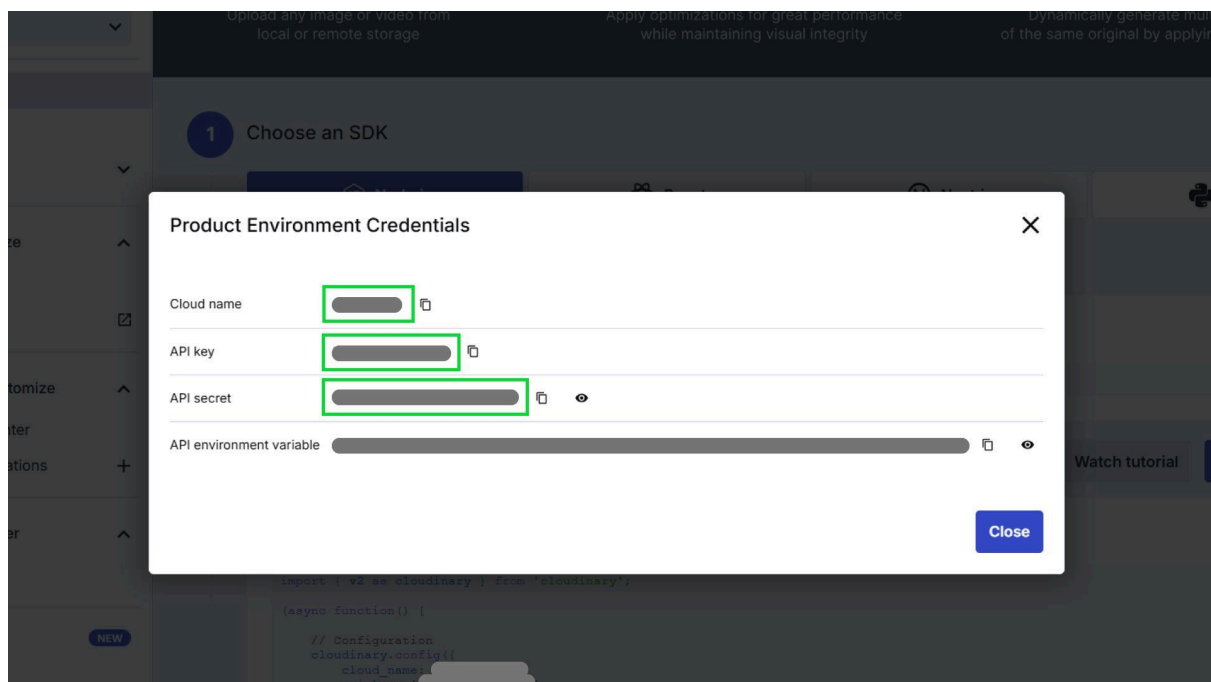
*Official Firebase documentation for Initializing the Firebase Admin SDK in non-Google environments:*

[https://firebase.google.com/docs/admin/setup#initialize\\_the\\_sdk\\_in\\_non-google\\_environments](https://firebase.google.com/docs/admin/setup#initialize_the_sdk_in_non-google_environments)

## Cloudinary Setup

We use Cloudinary to store and distribute images for our platform.

- Create a Cloudinary account at [https://cloudinary.com/users/register\\_free](https://cloudinary.com/users/register_free)
- From Cloudinary dashboard click View API Keys.
- Copy Cloud name, API key, and API secret keys.



- Then open the backend code in VS code and go to .env file, then go to Cloudinary section and place the keys in their respected fields.

# Cloudinary

```

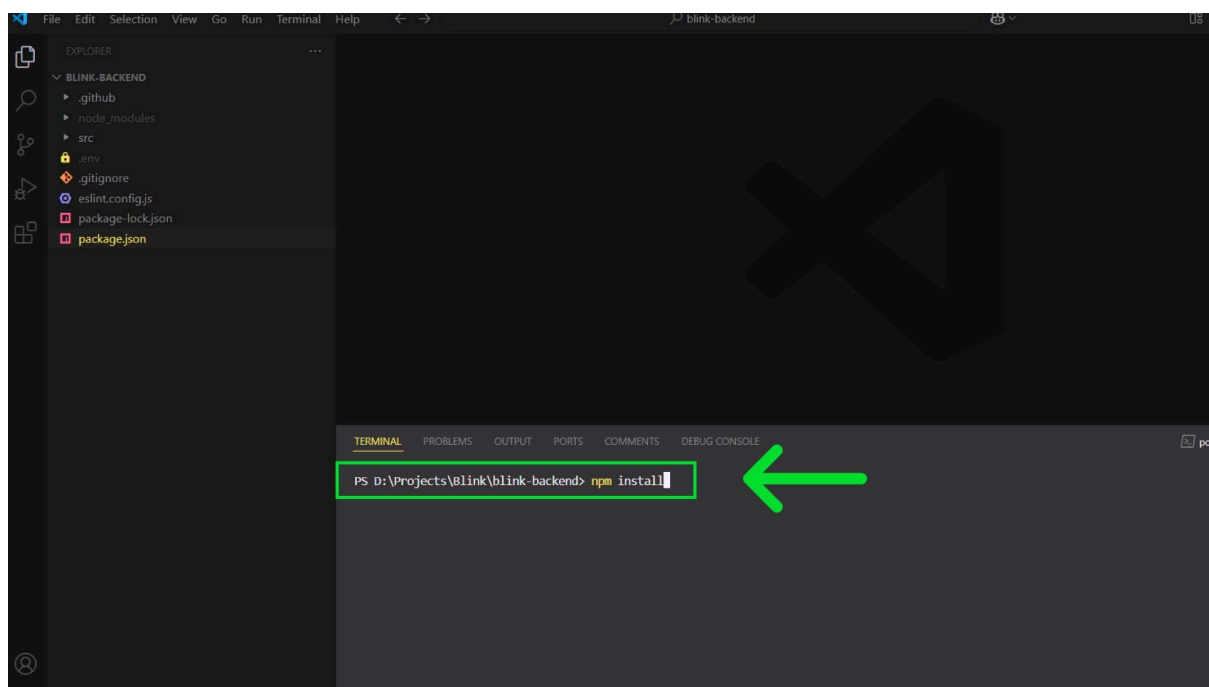
CLOUDINARY_CLOUD_NAME="*****"
CLOUDINARY_API_KEY="*****"
CLOUDINARY_API_SECRET="*****"

```

## Package Installation for Backend

In this section we will install npm packages on our backend.

- Open frontend code in VS Code.
- Open terminal of VS code.
- Make sure the terminal is pointing to the root folder.
- Run this command on the terminal `npm install`.



- The above command will install npm packages. It might take few minutes.
- Windows / Linux shortcut to open VS Code terminal **Ctrl + `**.
- Mac shortcut to open VS Code terminal **Command + `**.

## Administrator Setup

In this section we will show step by step on how to set admin and moderator.

# Admin Setup

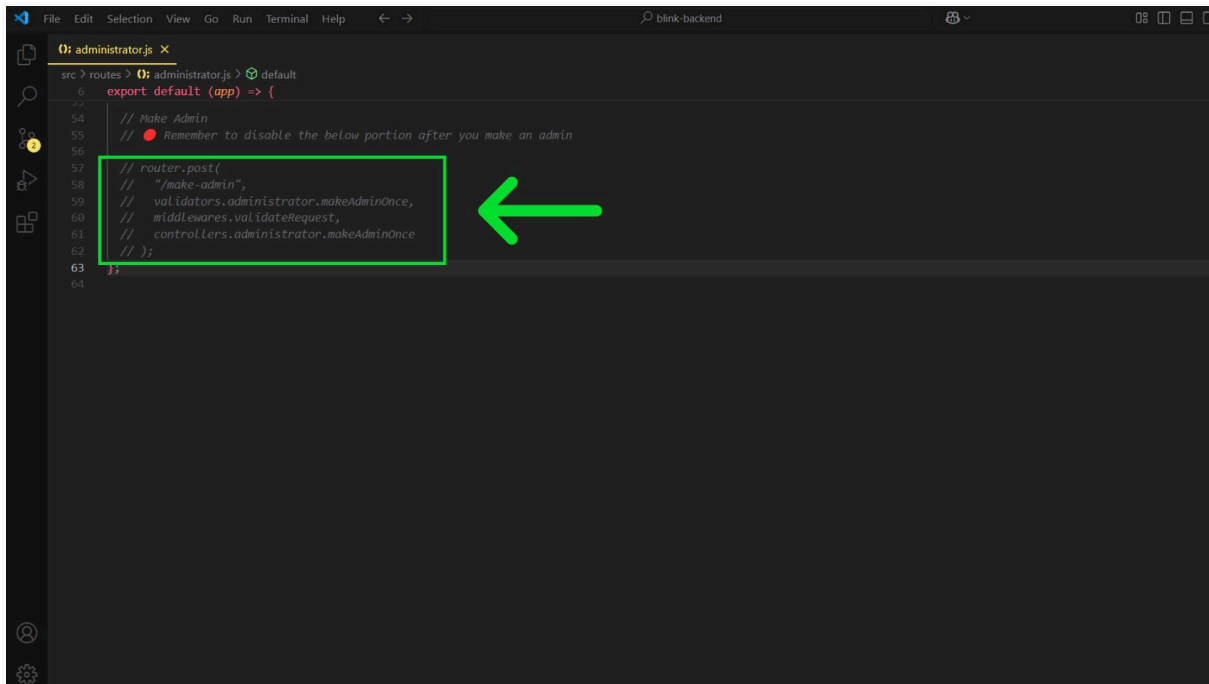
By default no user is assigned as an Admin. So we have to manually add an Admin for the very first time.

- Open your backend code then go to this path `src -> routes -> administrator.js`. Then scroll down to the end of the page, you will see `/make-admin` path. This path is commented in to disable access. Remove the comment and activate the path.

```
// Make Admin  
// 🚫 Remember to disable the below portion after you make an admin  
  
// router.post(  
  // "/make-admin",  
  // validators.administrator.makeAdminOnce,  
  // middlewares.validateRequest,  
  // controllers.administrator.makeAdminOnce  
  // );
```

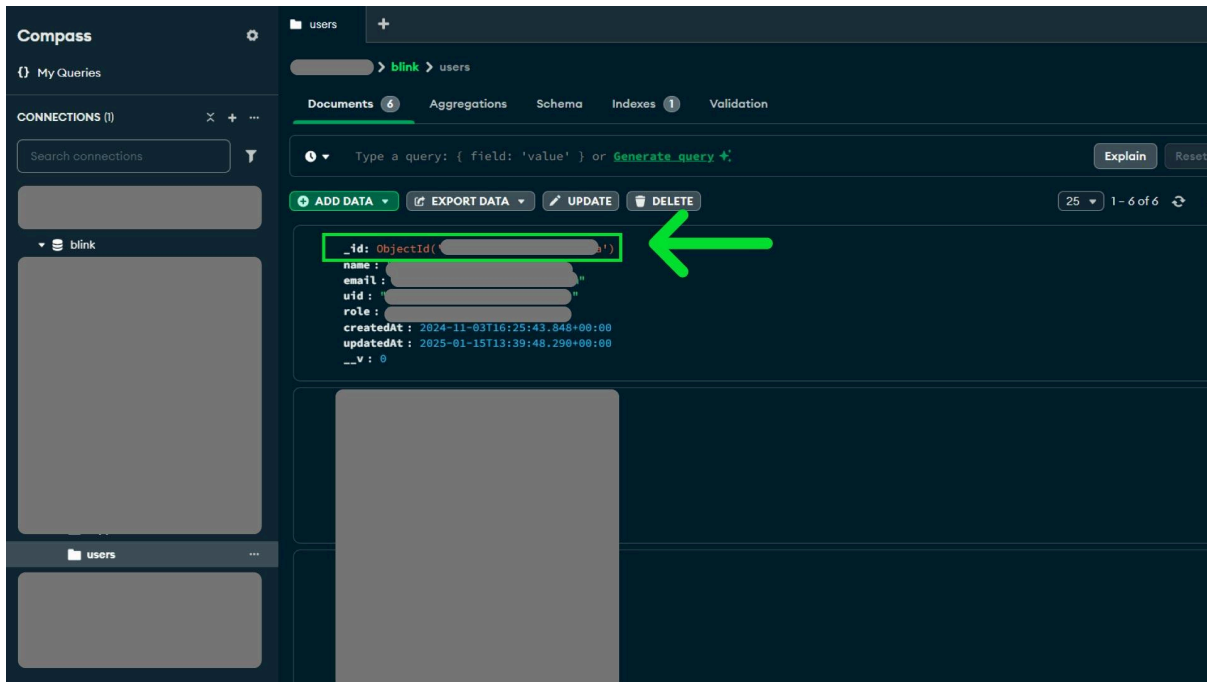
Make it like below:

```
// Make Admin  
// 🚫 Remember to disable the below portion after you make an admin  
  
router.post(  
  "/make-admin",  
  validators.administrator.makeAdminOnce,  
  middlewares.validateRequest,  
  controllers.administrator.makeAdminOnce  
);
```

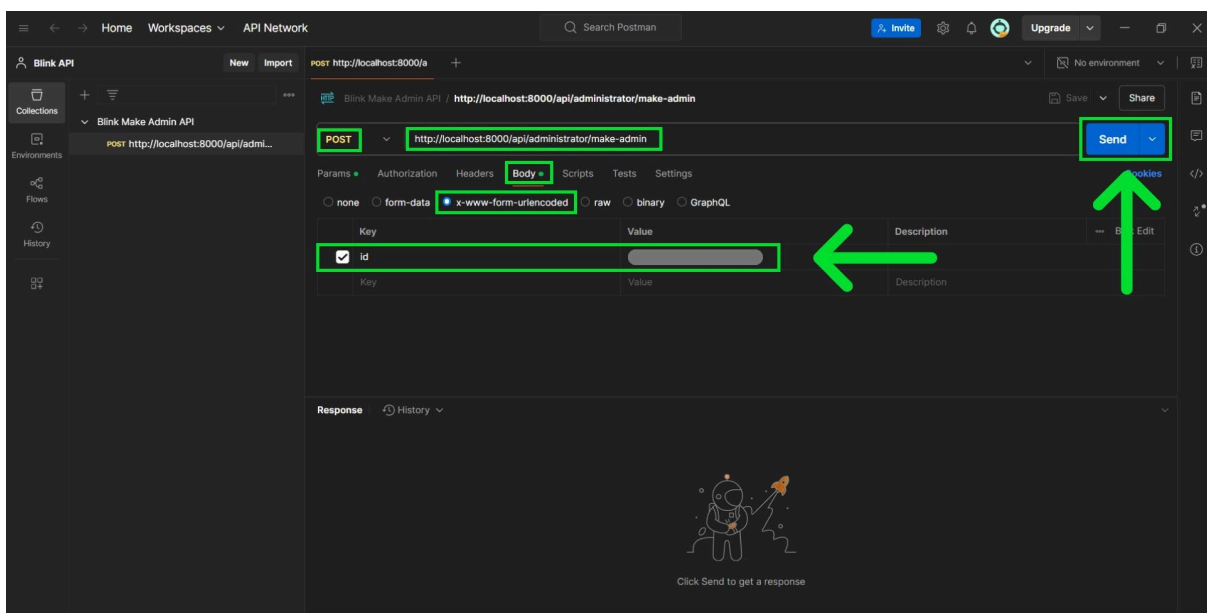


```
File Edit Selection View Go Run Terminal Help ← → blink-backend
0: administrator.js x
src > routes > 0: administrator.js > default
export default (app) => {
  // Make Admin
  // Remember to disable the below portion after you make an admin
  // router.post(
  //   "/make-admin",
  //   validators.administrator.makeAdminOnce,
  //   middlewares.validateRequest,
  //   controllers.administrator.makeAdminOnce
  // );
}
```

- Run your backend using `npm start` command. Your backend should run on `http://localhost:8000` port.
- Run your frontend using `npm run dev` command. It should run your frontend on `http://localhost:3000` port.
- Now visit `http://localhost:3000` and create an account.
- By default you will become a normal user, without any administrative capabilities.
- Now our task is to make you an Admin.
- Download and Install MongoDB Compass from <https://www.mongodb.com/products/tools/compass>
- Open MongoDB compass, and create a new connection using your MongoDB connection string.
- Then you will get to see your MongoDB data there.
- Go to users data. There you will see your account information.
- Copy the value of `_id`, for example it looks something like `ObjectId( ' 6775020a7e5ad4d6da9d1989' )`. We just need the value, that is `6775020a7e5ad4d6da9d1989`.



- Now we will make an API request to our server to make this user an Admin.
- You can use any API request making tool like [Postman](https://postman.com/), or you can use online tool like <https://apirequest.io/>
- From your API request making tool, select Request type to POST request.
- Set the API URL to `http://localhost:8000/api/administrator/make-admin`
- In request Body section, click x-www-form-urlencoded the write Key as id and Value as 6775020a7e5ad4d6da9d1989. Remember to place your own id here.
- Then send the request. It will make the requested user an admin.



- Important: Now go to your backend code and comment out the `/make-admin` path, so that no one get the access to this path again.

*Remember the `make-admin` only runs for once. If you have at least 1 admin, it'll not run.*

- Now logout from the frontend and login again, you will see the Admin privileges in your account. You can further manage admins from the frontend dashboard.
- 

## Moderator Setup

Let's make a moderator for our platform. To become a moderator, the user must have an account on the platform.

*Only Admin can add a new Moderator.*

## Add New Moderator

- Log in from an account which has Admin privileges.
- Go to the Administrators tab.
- Click on Add New Administrator button
- Search moderator with his email.
- Assign him a moderator role.

*If you want to revoke or change a Moderator's role, from Administrator List go to that user's details, then assign a new role.*

## Hosting and Deployment

This section outlines the backend and frontend hosting and deployment process.

## Backend Deployment

In this section we will deploy our backend code to a hosting provider. Our backend is developed using NodeJS, so we need a NodeJS compatible hosting provider. There are 2 ways to deploy a NodeJS project, one is using the NodeJS build file, and another one is using the CI/CD pipeline.

*There are 2 types of hosting providers.*

- *Managed NodeJS hosting, also known as App Service or Functions. Example Heroku, Hostman, Render, Azure App Service, DigitalOcean App Platform, AWS Elastic Beanstalk, Google Cloud App Engine etc. This services can directly take code from your GitHub and deploy them.*
- *Self managed hosting, where you have to manage and maintain the cloud server and deployments. It's mostly suitable for large applications.*

## Managed NodeJS Hosting Using GitHub

- Create a private repository in Github: Follow <https://docs.github.com/en/repositories/creating-and-managing-repositories/quickstart-for-repositories>
- Upload your backend code to your GitHub private repository. Follow official github guideline: <https://docs.github.com/en/repositories/working-with-files/managing-files/adding-a-file-to-a-repository#adding-a-file-to-a-repository-using-the-command-line>
- After successfully uploading you backend code to GitHub, follow your Managed Hosting provider's guideline to deploy your backend code. This step is easy and straightforward.
- Remember to add Environment variables , without environment variables, server will fail to connect with database and other services. In some providers you might need to press Redeploy button after providing Environment variables.

## Self-Managed Hosting

Follow your cloud service providers guideline to upload and deploy code.

## Frontend Deployment

In this section we will deploy our frontend code to a hosting provider. Our frontend is developed using ReactJS and NextJS, so we need a NextJS compatible hosting provider.

*There are 2 ways to deploy the frontend.*



- *Managed frontend hosting providers like Vercel, Netlify, Render etc. It's very easy to use, and most providers offer free options.*
- *Self managed hosting, where you can deploy the build file directly.*

## Managed NextJS Hosting Using GitHub

- Create a private repository in Github: Follow <https://docs.github.com/en/repositories/creating-and-managing-repositories/quickstart-for-repositories>
- Upload your frontend code to your GitHub private repository. Follow official github guideline: <https://docs.github.com/en/repositories/working-with-files/managing-files/adding-a-file-to-a-repository#adding-a-file-to-a-repository-using-the-command-line>
- After successfully uploading you frontend code to GitHub, go to your hosting provider and connect your github account.
- Then allow access to your private repository to your hosting provider.
- Remember to add Environment variables , without environment variables, frontend will fail to connect with backend.

## Self-Managed Hosting

- Run build command in your frontend code `next build`
- It will provide you with a build file.
- Then follow your hosting providers guideline to deploy the build file.

## Conclusion

Thank you for exploring Blink documentation. Hopefully, you have finished all of the steps and got your platform up and running. If you have any questions, need further assistance, or have feature requests, our support team is here to help. Stay updated with our latest improvements and enhancements by following our release notes. We look forward to seeing how you make the most of the platform!