****

**Schoolep**

*Web app for better productivity in school or work*

***autor***: fistrba

***portfolio***: codecanyon.net/user/fistrba/portfolio

**Thank you for choosing our app**

We want to express our heartfelt gratitude for choosing Schoolep as your go-to app. Your support means the world to us, and we are committed to providing you with an exceptional experience. Thank you for being a part of our community!

**Contents**

[***1. Schoolep***](#_30j0zll)

[1.1 How to setup Code](#_1fob9te)

[1.2 How to setup Firebase](#_3w347qxewn0y)

[1.2 How to setup Server](#_3znysh7)

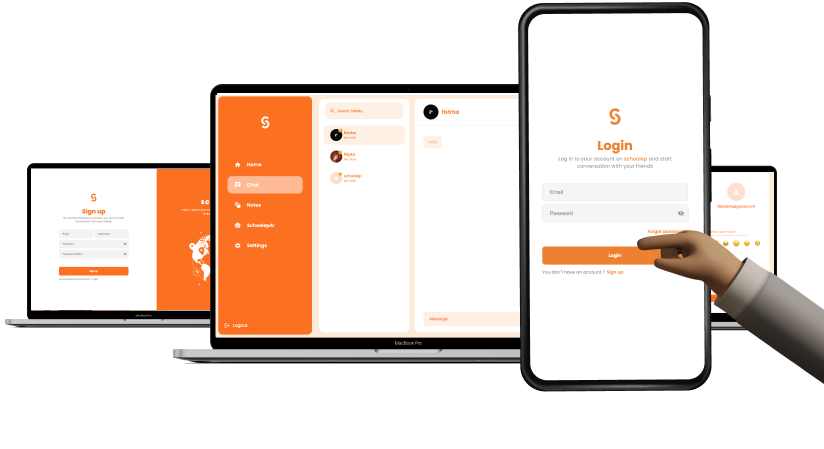
[1.3 How to setup reCAPTCHA](#_nm3gj8rxvy3a)

# 

# 

# ***1. Schoolep***

Schoolep is chat web application that serves as a platform for students or workers to communicate easily and effectively. It is available to all users who have access to the Internet through a web browser. It offers communication between users, creation of notes and collaboration with artificial intelligence, all in one place.



# ***1.1 How to setup Code***

To open a file in Visual Studio Code, follow these steps:

1. Launch Visual Studio Code.

2. Click on "***File***" in the top menu.

3. Select "***Open File***" from the drop-down menu.

4. Navigate to the directory where the file is located.

5. Choose the file you want to open and click "***Open***."

To open the terminal in Visual Studio Code, you can use the following keyboard shortcuts:

- For Windows and Linux: Press `***Ctrl +*** ` ` to open the integrated terminal.

- For macOS: Press `***Cmd +*** ` ` to open the integrated terminal.

Once you have the terminal open in Visual Studio Code, follow these steps to install all the packages using npm:

1. Ensure that you are in the correct directory where the project's `package.json` file is located.

2. In the terminal, type `***npm install***` and press Enter.

3. npm will then download and install all the packages listed in the `***package.json***` file.

After the package installation is complete, follow these steps to start the web application on localhost:

1. In the terminal, type `***npm start***` and press Enter.

2. This command will run the start script defined in the `***package.json***` file.

3. The web application will start, and you can access it by navigating to `***http://localhost***` or the specific port mentioned in your application.

Make sure you have Node.js and npm (Node Package Manager) installed on your system before running the above commands.

# ***1.2 How to setup Firebase***

1. Visit the official Firebase website at [firebase.google.com](https://firebase.google.com/) and navigate to the Firebase console.

2. In the Firebase console, click on the "Add project" button to create a new project.

3. Provide a name for your project (e.g., "example") and wait for the project to be set up.

4. Once the project is created, you will be redirected to the project overview page.

5. To integrate Firebase into your web application, click on the "Web" option to register your app and give it a suitable name.

6. After registering your app, you will see a configuration snippet. Make sure to copy your own Firebase config from this snippet.

7. Open your code editor and navigate to your "firebase.js" file. Paste the copied Firebase config into the appropriate location in the file.

8. Save the changes to your "firebase.js" file.

9. Return to the Firebase console by going back to firebase.com.

10. In the Firebase console, go to the "Authentication" section and click on "Get started". Then, choose the "Email/Password" option and enable it.

11. Next, navigate to the "Realtime Database" section and click on "Create Database". Choose the desired location for your database. Once created, go to the "Rules" tab and update the rules according to your requirements. You can refer to the provided image for guidance.

12. Similarly, go to the "Firebase Storage" section and click on it. Navigate to the "Rules" tab and adjust the rules as per your needs. You can refer to the provided image for guidance.

These instructions should help you set up a Firebase project, configure it for your web application, enable email/password authentication, and customize the rules for the Realtime Database and Firebase Storage.

# 

# 

# 

# 

# ***1.2 How to setup Server***

1. Open the main file and locate the server file.

2. Within the server file, you will find a JavaScript file named "server.js" along with "package.json" and "package-lock.json". Follow these steps:

a. Visit ***https://platform.openai.com/overview***

b. Click on your profile picture and select "Manage Account".

c. Look for the organization ID, and copy it.

d. In the "server.js" file, locate the configuration section and update the `***organization***` field with the copied organization ID. Use the following code:

const configuration = new Configuration({

organization: "org-nMUHWGfRiJwMmYcFN2oKrhZT",

apiKey: "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",

});

e. To obtain an API key, go to the "View API Keys" section and create a new key. Copy the generated API key.

f. In the same "server.js" file, update the `***apiKey***` field in the configuration with the copied API key.

3. Ensure that all server files are uploaded to your GitHub repository.

4. Visit ***https://app.cyclic.sh/#/,*** log in to your account, and click on "New App". Choose to link your own app using GitHub and locate the repository you added in step 3.

5. After adding your server to Cyclic, copy the provided link. Open the Visual Studio Code (VS Code) editor, navigate to `***src > pages > schoolepAI***`, and locate the fetch code block. Paste the copied link as the URL from which the web will fetch data.

These instructions should help you locate the server file, configure the OpenAI organization ID and API key, upload the server files to GitHub, link your server to the Cyclic platform, and update the fetch URL in your VS Code project.  
  
Change OPENAI\_SK variable in .variables.env on backend to openAI secret key

# ***1.3 How to setup reCAPTCHA***

To configure your own reCAPTCHA, follow these steps:

1. Visit ***https://www.google.com/recaptcha/admin/create*** to create a new reCAPTCHA.

2. Fill in the necessary details:

- Provide a name for your project.

- Select the reCAPTCHA type. Choose "reCAPTCHA v2" and enable the "I'm not a robot" checkbox.

- Add the domains. Start by adding "localhost" if you're testing locally. Later, when you're ready to deploy your website, modify the configuration and add your own domain.

3. Click the "Submit" button to create your reCAPTCHA configuration.

4. After the configuration is created, you will be provided with a site key. Copy this ***site key.***

5. Open the Visual Studio Code (VS Code) editor and navigate to `***src > pages > signup***` in your project.

6. Locate the reCAPTCHA component and find the `siteKey={}` prop.

7. Paste the copied site key inside the `siteKey={}` prop. Your code should look similar to the following:

***<reCAPTCHA siteKey="YOUR\_SITE\_KEY\_HERE" />***

8. Save the changes to the file.

These instructions should guide you in creating and configuring your own reCAPTCHA. Remember to replace "YOUR\_SITE\_KEY\_HERE" with the actual site key you obtained from the reCAPTCHA admin panel.

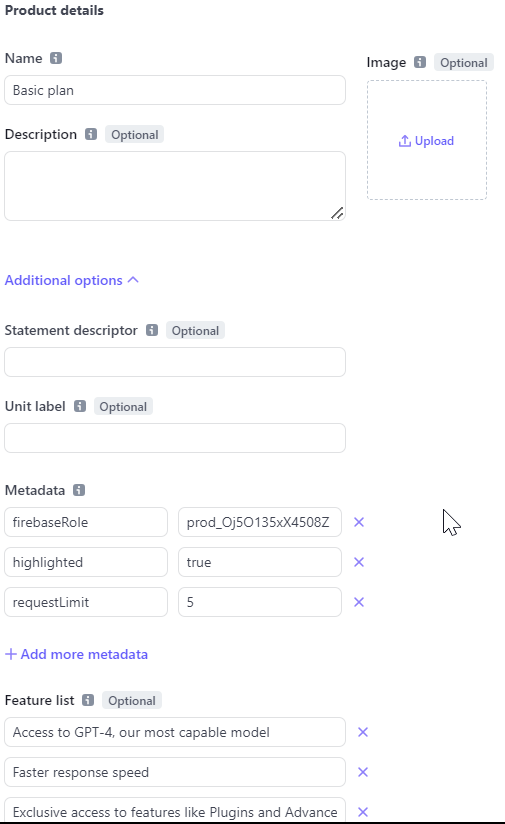
Clone frontend repository

Setup stripe account  
  
1. Create account and add store details.

2. Copy Secret Key to REACT\_APP\_STRIPE\_SK and Publishable Key to REACT\_APP\_STRIPE\_PK in .env file on frontend

3. Go To [Stripe Customer Portal](https://dashboard.stripe.com/settings/billing/portal) to Enable stripe customer portal and Toggle Cancel subscription immediately

4. Go To [Checkout and Payment Links settings](https://dashboard.stripe.com/settings/checkout#subscriptions) and Limit customers to 1 subscription  
5. Add your desired product plan with Metadata  
firebaseRole productID

highlighted true  
requestLimit 5   
  
Example  
  
  
Create Prices accordingly  
  
  
  
  
Firebase Account setup  
  
1. After following 1.2, enable firebase firestore.

2. Run and install [Firebase Stripe Extension](https://blog.jarrodwatts.com/set-up-subscription-payments-with-stripe-using-firebase-and-nextjs#the-firebase-stripe-extension) following the article from Firebase Stripe Extension untill Configure Stripe Webhooks.

3. Change the firestore rules to

3. Change firestore rules to

rules\_version = '2';

service cloud.firestore {

match /databases/{database}/documents {

match /users/{uid} {

allow write: if request.auth.uid != null ;

allow read: if true;

match /messages/{id}{

allow read,write: if request.auth.uid != null ;

match /sawTime/{timeId}{

allow read,write: if request.auth.uid != null ;

}

}

match /messages/{document=\*\*}{

allow read,write: if request.auth.uid != null ;

}

match /checkout\_sessions/{id} {

allow read, write: if request.auth.uid == uid;

}

match /subscriptions/{id} {

allow read: if request.auth.uid == uid;

}

}

match /messages/{document=\*\*}{

allow read,write: if request.auth.uid != null ;

}

match /notes/{uid}{

match /all/{id}{

allow read, write: if request.auth.uid == uid;

}

match /folders/{document=\*\*}{

allow read, write: if request.auth.uid == uid;

}

}

match /noteIds/{id}{

allow read,write: if request.auth.uid!=null;

}

match /messageIds/{id}{

allow read,write: if request.auth.uid!=null;

}

match /shutdown/{id}{

allow read,write: if true;

}

match /products/{id} {

allow read: if true;

match /prices/{id} {

allow read: if true;

}

match /tax\_rates/{id} {

allow read: if true;

}

}

}

}

[Cite your source here.]

4. find firebase configuraion credentals from Settings > General Settings and  
Change credentials on frontend/src/firebase.js  
Change credentials on schoolep/src/firebase.js

After a successfull run on creating a user edit it in firebase firestore database users collection add admin:true field  
this user will be the admin for backend admin pannel

Setup firebase-functions  
  
mkdir stripe-webhooks  
cd stripe-webhooks

npm install -g firebase-tools

firebase login  
firebase init

Choose firebase project  
Select firestore and cloud functions

cd functions/

npm install axios stripe cors

Replace contents of index.js

/\*\*

 \* Import function triggers from their respective submodules:

 \*

 \* const {onCall} = require("firebase-functions/v2/https");

 \* const {onDocumentWritten} = require("firebase-functions/v2/firestore");

 \*

 \* See a full list of supported triggers at https://firebase.google.com/docs/functions

 \*/

const functions = require("firebase-functions");

const admin = require("firebase-admin");

const logger = require("firebase-functions/logger");

const axios = require("axios");

const cors = require("cors")({ origin: true });

// Create and deploy your first functions

// https://firebase.google.com/docs/functions/get-started

const serviceAccount = require("./serviceAccountKey.json");

const stripe = require("stripe")(functions.config().keys.webhooks);

const endpointSecret = functions.config().keys.signing;

admin.initializeApp({

  credential: admin.credential.cert(serviceAccount),

  databaseURL: functions.config().keys.firebase\_databaseurl, // Replace with your Firebase project URL

});

const db = admin.firestore();

exports.events = functions

  .runWith({ maxInstances: 50 })

  .https.onRequest((request, response) => {

    cors(request, response, async () => {

      let sig = request.headers["stripe-signature"];

      let event;

      try {

        event = stripe.webhooks.constructEvent(

          request.rawBody,

          sig,

          endpointSecret

        );

        switch (event.type) {

          case "payment\_intent.created":

            const invoicePaymentCreated = event.data.object;

            // Then define and call a function to handle the event invoice.payment\_succeeded

            break;

          case "invoice.payment\_succeeded":

            const invoicePaymentSucceeded = event.data.object;

          case "customer.subscription.created":

            const customerSubscriptionCreated = event.data.object;

            const docsSnapshot = await db

              .collection("users")

              .where("stripeId", "==", customerSubscriptionCreated.customer)

              .get();

            const userSnap = docsSnapshot.docs[0];

            await db.doc(`users/${userSnap.id}`).update({

              userChatRequestCount: parseInt(

                functions.config().keys.user\_free\_request\_limit

              ),

            });

            break;

          default:

            console.log(`Unhandled event type ${event.type}`);

        }

      } catch (err) {

        console.log(`❌ Error message: ${err.message}`);

        response.status(400).send(`Webhook Error: ${err.message}`);

        return;

      }

      console.log("✅ Success:", event.id);

      response.json({ received: true });

    });

  });

async function requestOpenAI(message) {

  const response = await axios.post(

    "https://api.openai.com/v1/completions",

    {

      prompt: `

          You: Your name is ShoolepAI,

          You: You are artificial intelligence, you answer exactly like chatGPT and openai,

          Me: I am a person who asks about various things

          Me: What is a fish?

          You: A fish is an animal species that lives in an aquatic environment and its body is covered with scales. Many types of fish are found in water such as rivers, lakes, seas and oceans. Fish are an important source of food for humans and are also important for the ecosystem of aquatic habitats.

          Me: ${message}, ., ?, !,

          You:

         `,

      max\_tokens: 50,

      temperature: 0,

      frequency\_penalty: 0.0,

      top\_p: 1,

      model: "text-davinci-003",

    },

    {

      headers: {

        Authorization: `Bearer ${functions.config().keys.openai\_sk}`,

      },

    }

  );

  return response.data?.choices[0].text;

}

function updateUserRequestCount({ userId, userChatRequestCount }) {

  if (!userChatRequestCount) {

    userChatRequestCount = 1;

  } else {

    userChatRequestCount++;

  }

  return db.doc(`users/${userId}`).update({ userChatRequestCount });

}

exports.chat = functions

  .runWith({ maxInstances: 50 })

  .https.onRequest((req, res) => {

    cors(req, res, async () => {

      try {

        const user = await admin

          .auth()

          .verifyIdToken(req.headers.authorization);

        const { message } = req.body;

        const userDocSnapshot = await db.doc(`users/${user.user\_id}`).get();

        let userChatRequestCount = userDocSnapshot.data().userChatRequestCount;

        if (user.stripeRole) {

          const subsDocs = await admin

            .firestore()

            .collection(`users/${user.user\_id}/subscriptions`)

            .where("status", "in", ["trialing", "active"])

            .get();

          const subDoc = subsDocs.docs[0];

          const interval = subDoc.data().items[0].price.recurring.interval;

          let requestLimit = parseInt(

            subDoc.data().items[0].price.product.metadata.requestsLimit

          );

          if (interval === "year") {

            requestLimit \*= 12;

          }

          if (userChatRequestCount >= requestLimit) {

            res.status(200).json({

              message:

                "You have reached your limit! please subscribe to our plans",

            });

          } else {

            const responseMessage = await requestOpenAI(message);

            await updateUserRequestCount({

              userId: user.user\_id,

              userChatRequestCount,

            });

            res.status(200).json({ message: responseMessage });

          }

        } else {

          if (

            userChatRequestCount >=

            parseInt(functions.config().keys.user\_free\_request\_limit)

          ) {

            res.status(200).json({

              message:

                "You have reached your limit! please subscribe to our plans",

            });

          } else {

            const responseMessage = await requestOpenAI(message);

            await updateUserRequestCount({

              userId: user.user\_id,

              userChatRequestCount,

            });

            res.status(200).json({ message: responseMessage });

          }

        }

      } catch (error) {

        console.log(error);

        return res.status(200).json({ message: error.message });

      }

    });

  });

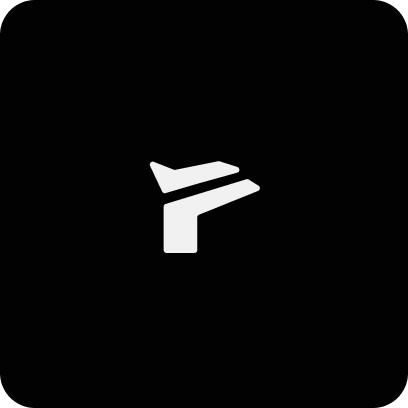
In Firebase console, open Settings > Service Accounts. Click Generate New Private Key, then confirm by clicking Generate Key. Rename the file to serviceAccountKey Add serviceAccountKey.json in functions folder.

Run  
  
firebase deploy --only functions

Copy link of “chat“ cloud function to frontend .env variable REACT\_APP\_AICHAT\_END\_POINT

Copy the link of “events“ cloud function and create a webhook endpoint on stripe enable customer.subscription.created event  
copy signing key and restricted key(from API keys)  
and run  
firebase functions:config:set \  
keys.webhooks="your\_restricted\_key" \  
keys.signing="your\_signing\_key"

Then deploy functions by running   
firebase deploy --only functions



*FISTRBA*

Contact us. We value your feedback and are here to assist you with any questions, concerns, or issues you may encounter while using Blockeact. If you need assistance or would like to reach out to us, please don't hesitate to contact our team.

[***FISTRBA@GMAIL.COM***](mailto:FISTRBA@GMAIL.COM)

[***SCHOOLEP.TEAM@GMAIL.COM***](mailto:SCHOOLEP.TEAM@GMAIL.COM)

[***instagram.com/fistrba***](http://www.instagram.com/fistrba)

[***t.me/fistrba***](https://t.me/fistrba)