UNIVERSITY OF SCIENCE

APCS

CS422 - Software Analysis and Design

Final Project Report

Nguyen Trong Nghia – 21125154
Bui Nguyen Hoang – 21125161
Pham Viet Hoang – 20125031

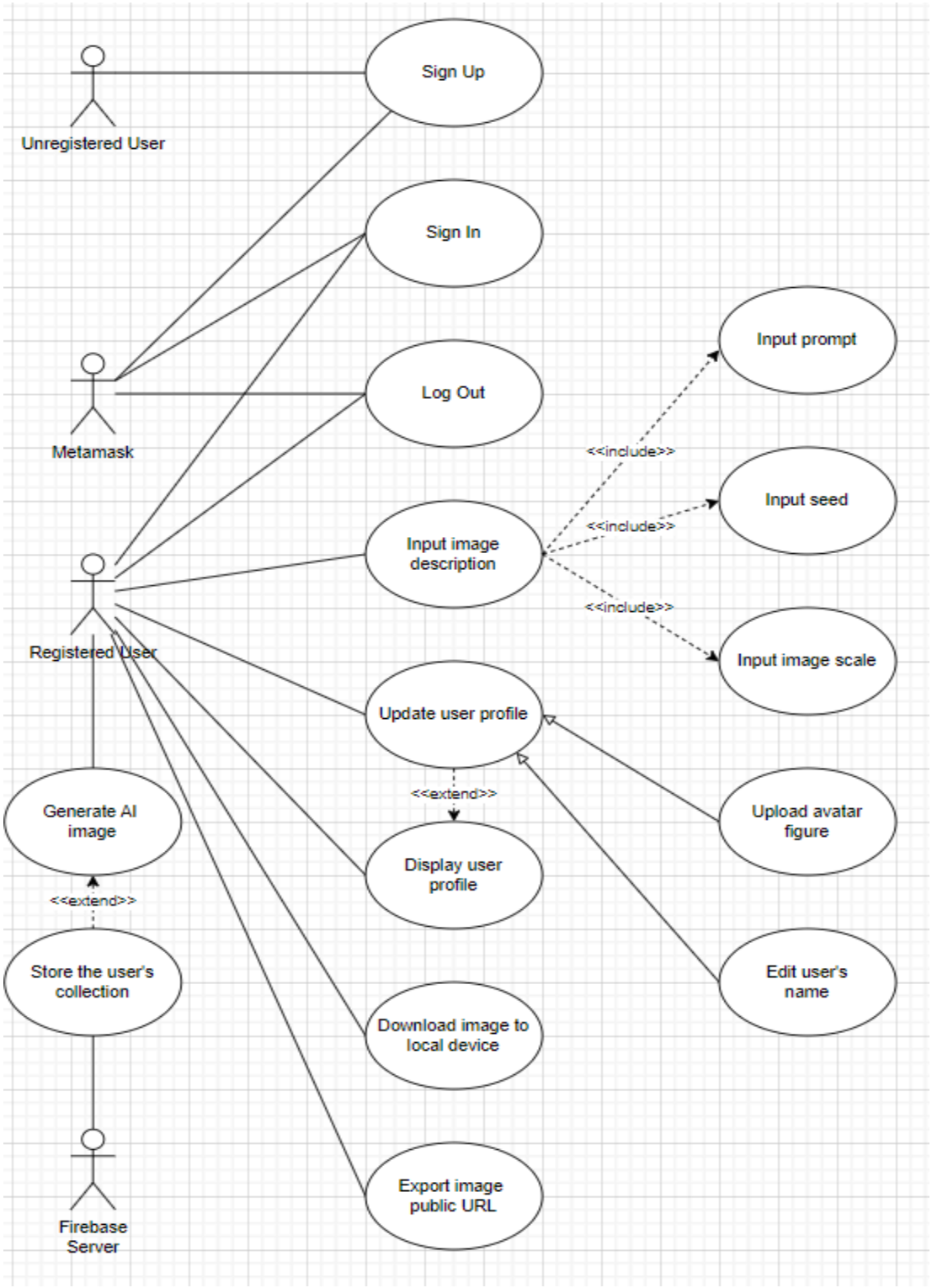
Table of Content

I. Problem Statement

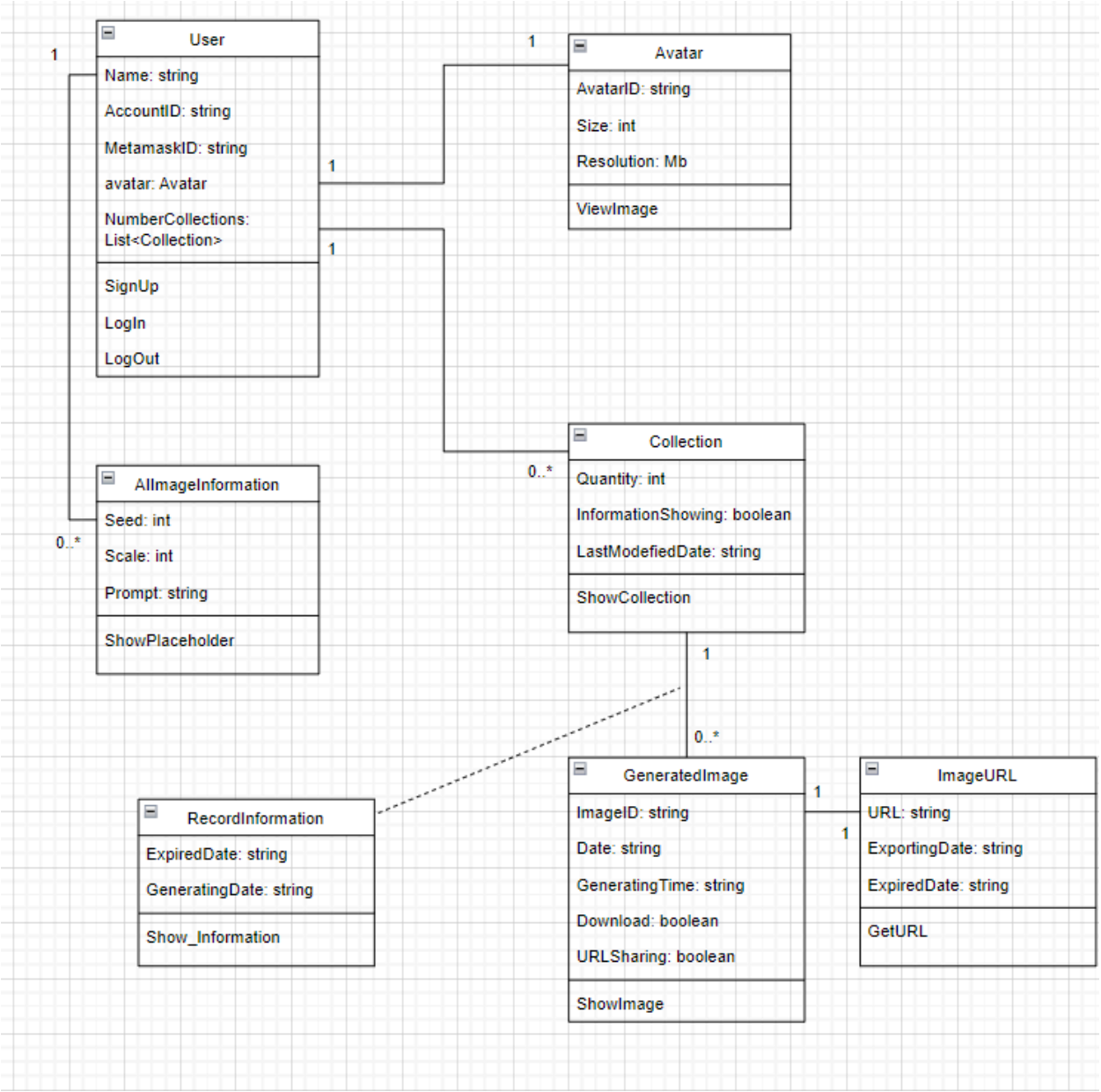
The problem of	providing a place where, in response to user input prompts, excellent and aesthetically pleasing photographs can be produced
affects	Users who desire to create personalized graphics according to their own input prompts or descriptions
the impact of which is	enables people to visually bring their imaginative concepts to life. It lets users employ created visuals to convey ideas, fantasies, or design specifications
a successful solution would be	offers a smooth and intuitive AI picture generation experience. The generated photos should be of the highest caliber, have a strong aesthetic appeal, and closely reflect the user's objective.

II. Diagrams

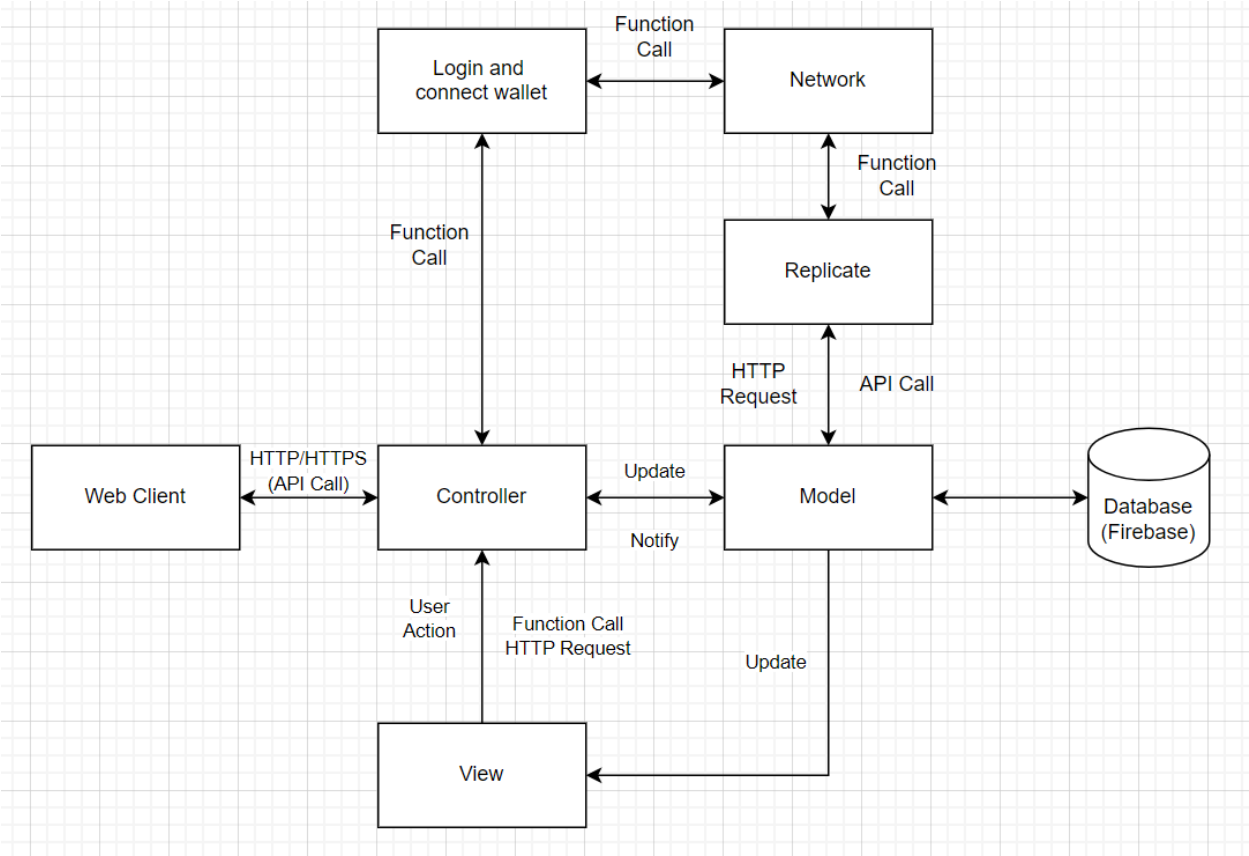
A. Use-case diagram



B. Class Diagram



C. Database


Login and connect wallet
Function Call
Network
Function Call
Function Call
Replicate
HTTP Request
API Call
Web Client
HTTP/HTTPS (API Call)
Controller
Update
Notify
Model
Database (Firebase)
User Action
Function Call HTTP Request
Update
View

III. Software requirements

A. Funtional Requirement

Sign up, Sign in	Users may use their Metamask wallet to log onto the program. This offers a safe, decentralized way to authenticate users to the site.
Log out	In order to conclude their session and prevent unwanted access to their account, users can safely log out of the program. The security and privacy of the user's account are guaranteed by this feature.
Connect Blockchain	Users may connect to a variety of blockchain networks with this program, including Polygon, Ethereum (ETH), and Binance Smart Chain (BSC). Users may now effortlessly complete blockchain transactions and engage with a variety of decentralized apps (DApps).
Input Prompt, Image scale	Permit the user to provide a prompt or a description for the image they want to create. Furthermore, whether they have any particular preferences for the preferred picture scale.
Update avatar figure	Within the program, users may upload and customize their avatar. This enables users to have a distinctive representation or image linked to their account and to customize their profile.
Update user profile	Users can change any pertinent information on their profile, including their username, bio, contact information, and other data. With the use of this function, individuals may maintain their profiles current and give other users or visitors correct information.
Store the user’s collection	The user's collection may be managed and stored using the application's capabilities. Images, artworks, and other digital assets may be stored and arranged by users inside their account for convenient access and retrieval.
View image	Users may see the image that the AI produces from within the application. With the help of this functionality, customers may easily see and assess the created photographs without requiring any additional hardware or software.

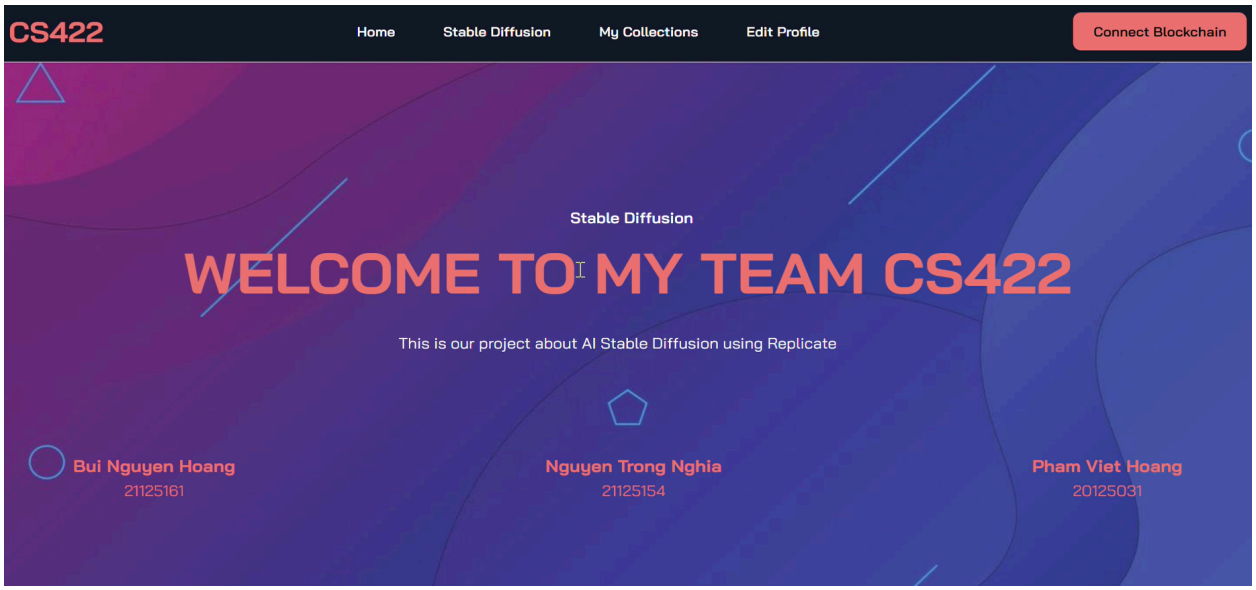
Show user profile	The application offers a specific page or area for showcasing user profiles. Users are able to view both their own and other users' profiles. Users may display their data, collections, and any other facts they wish to make public with this feature.
Download image to local device	The program offers users the ability to download artwork or photographs on their local device. This lets users save a local copy of the picture for use in offline mode, printing, or other purposes.
Export image public URL	For the photographs or artwork they post, users can create a public URL. You may share this URL with other people so they can view the image immediately without having to log into the program. This tool makes it simple to share and distribute photos.

B. Non-functional Requirement

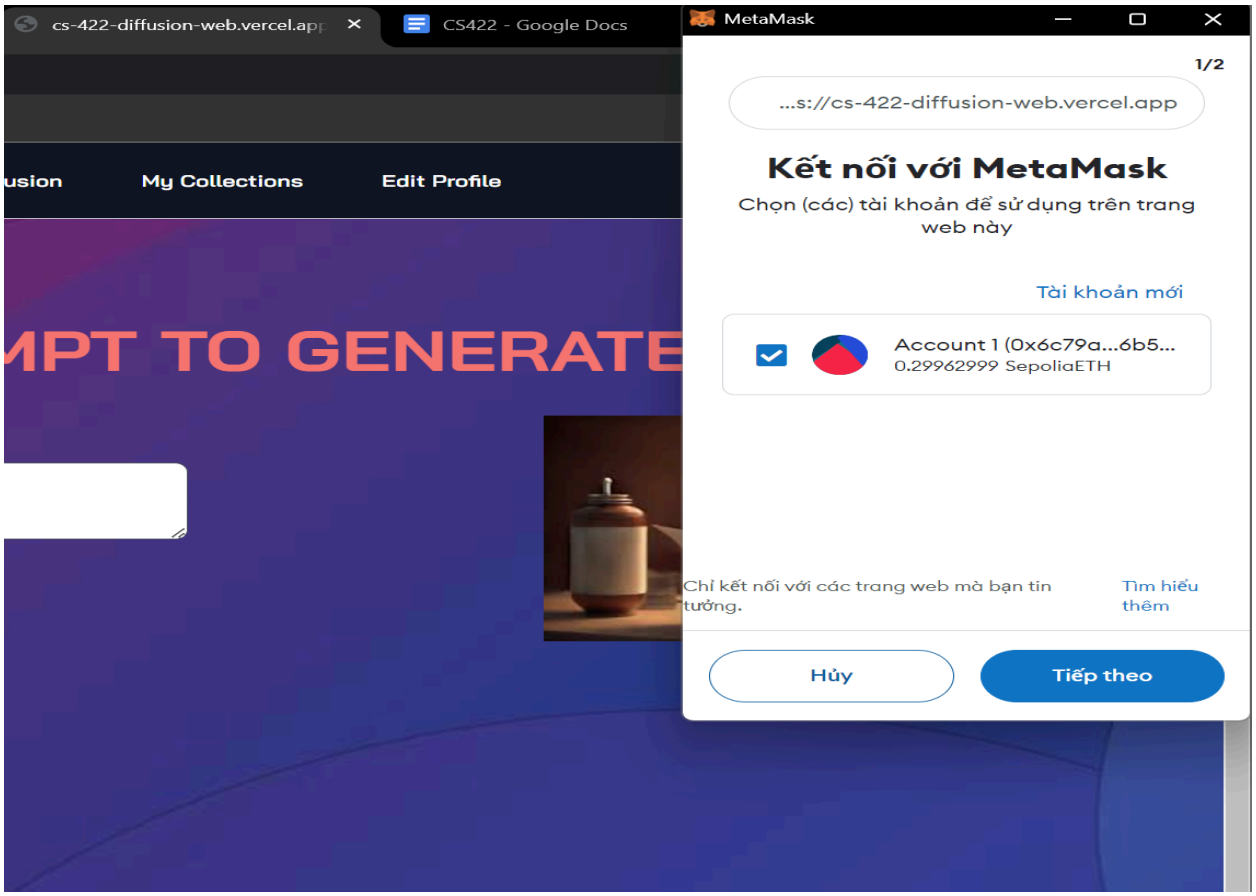
Performance	If there were no execution errors, the response time would not exceed 10 seconds.
Resilience	Users will just need to refresh the page or give it some time before trying to use it again in the event of minor errors. Larger failures would include sufficient elements on the website to allow people to learn about the impact of the failure on themselves.
Maintainability	Reducing the time and expense required to identify and fix system flaws in the future is possible when website maintenance is prioritized from the start.
Security	Ensure the security of user information.
Scalability	The system can function adequately even with a lot of data in the database, but it might not be able to handle too many users at once (about thousands of people at a time will be able to utilize the system effectively).
High reliability	Ensuring consistent and dependable operation under varying conditions.
Usability	Designing interfaces and interactions that are intuitive and efficient for users to navigate and utilize effectively.

IV. Interface Design

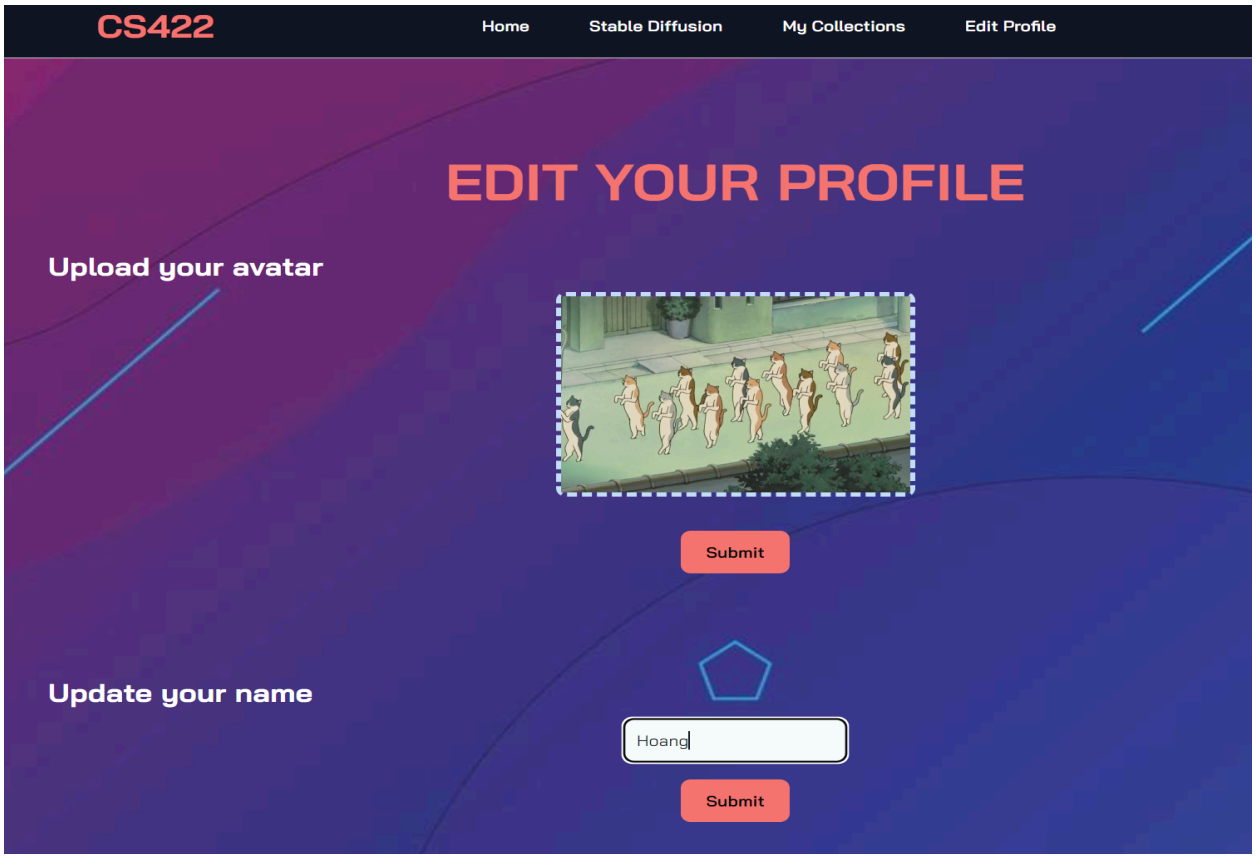
A. Home Page:

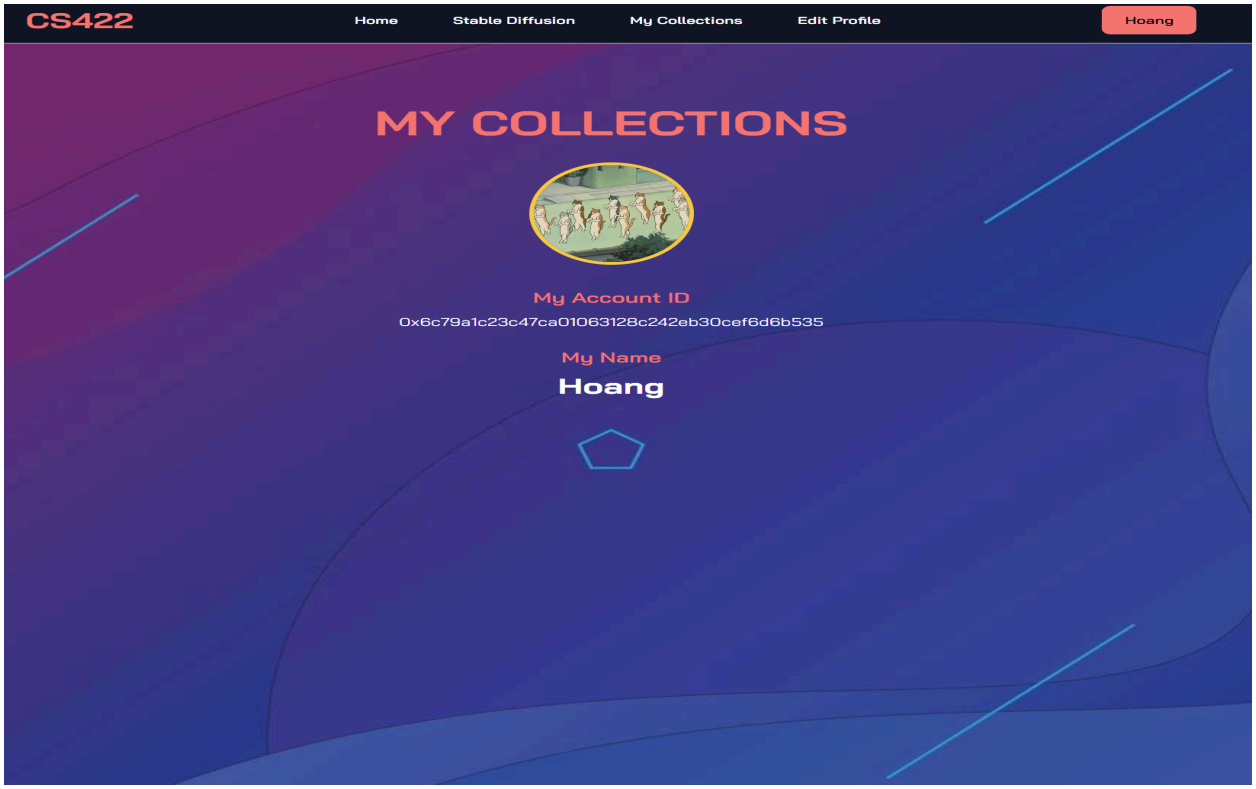


B. Connect Metamask Wallet:

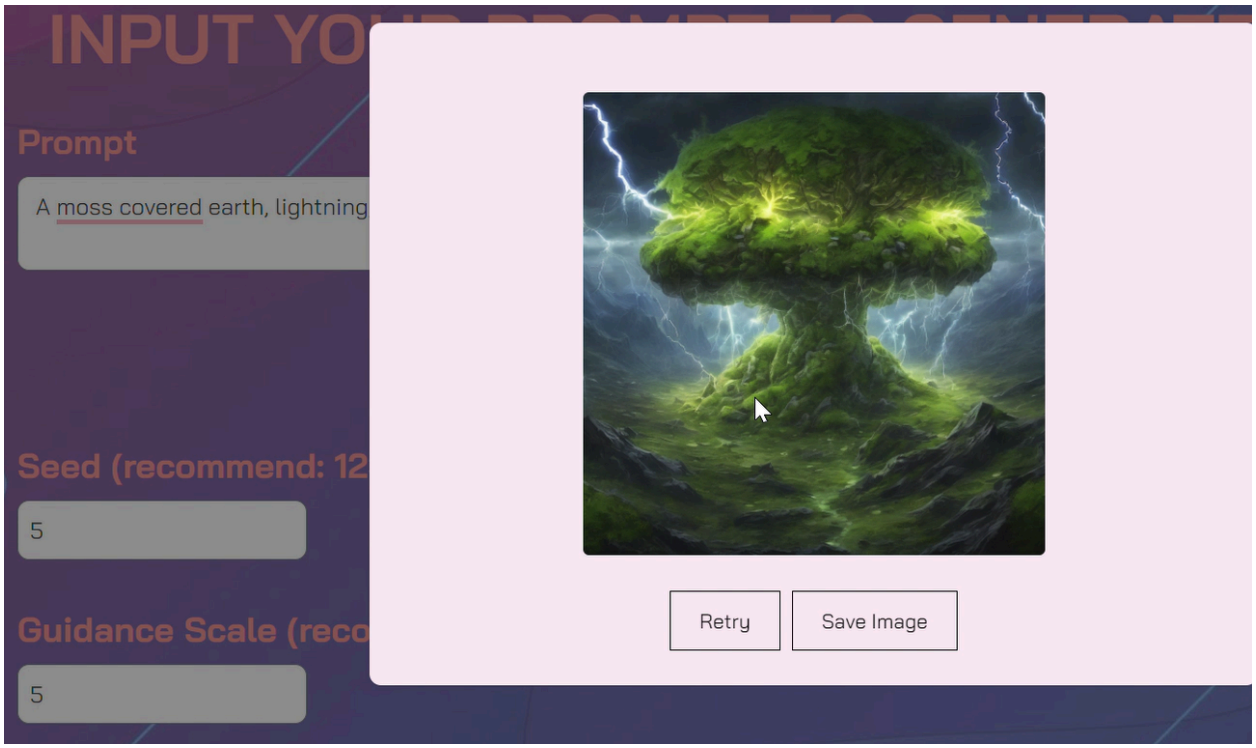


C. Edit profile Name and Avatar:

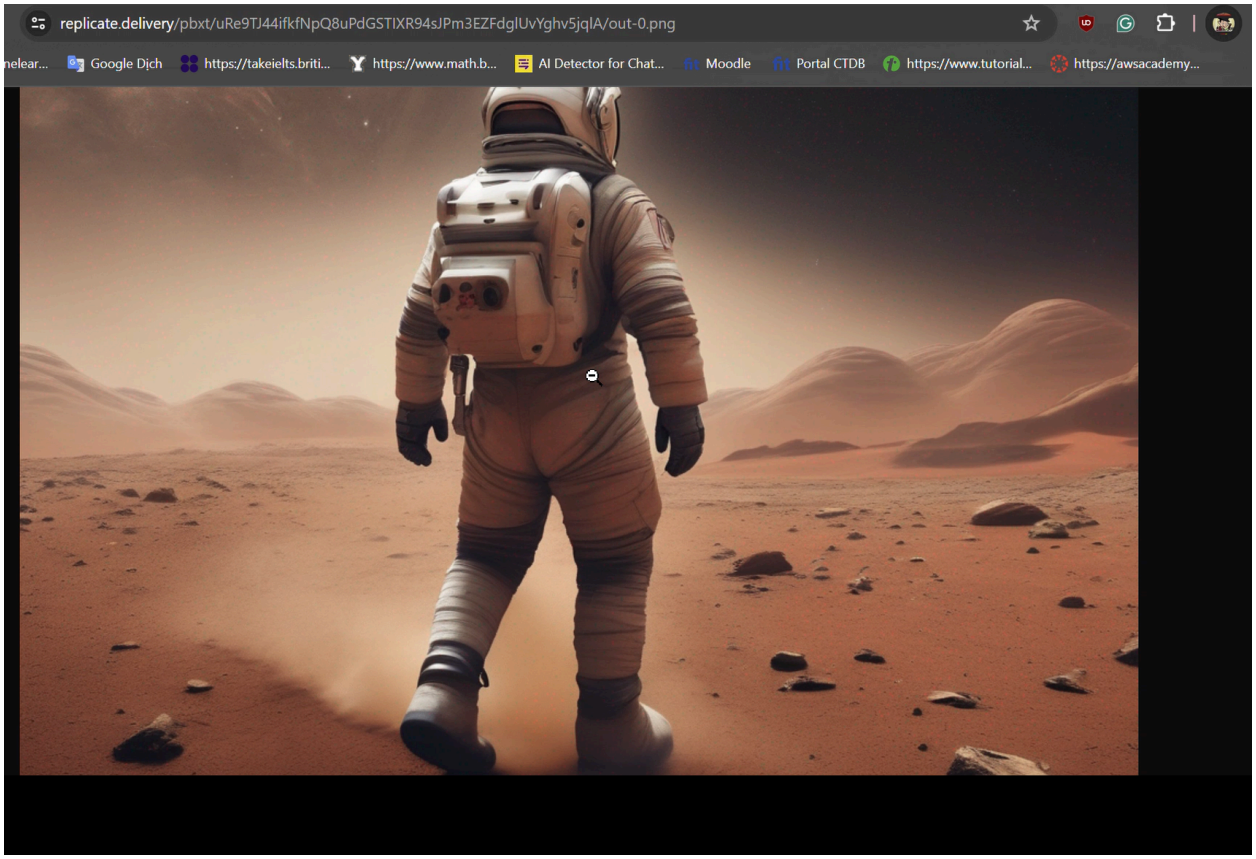




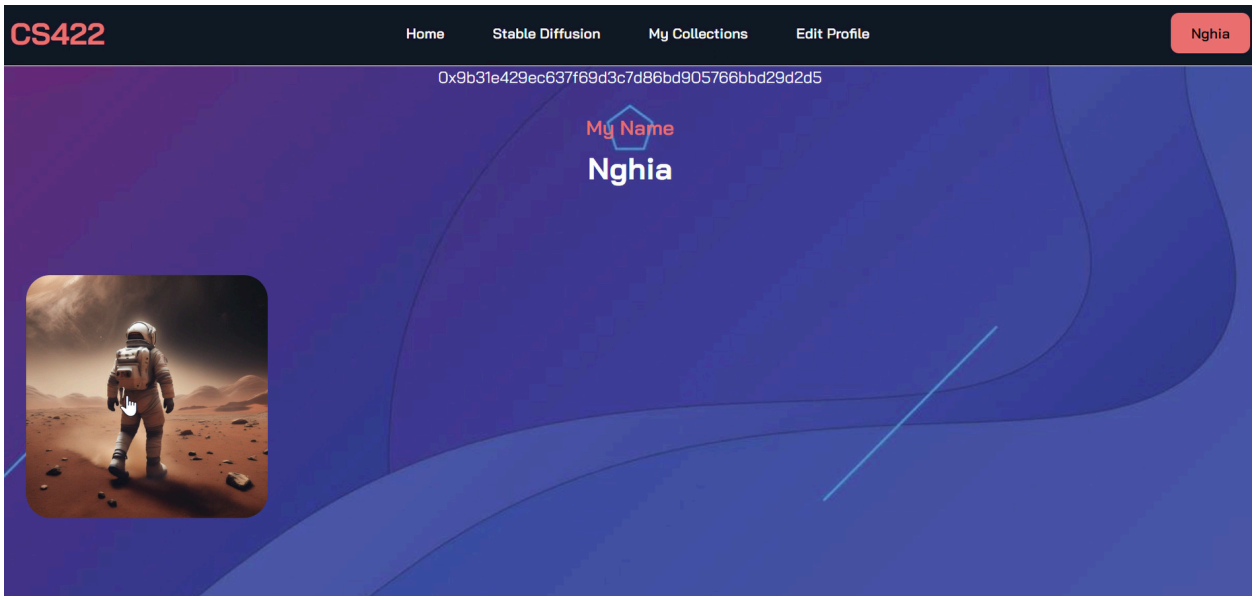
D. Generate AI images:



E. Share URL or download image:



F. View the Collection:



V. Implemented Functions

A. Connect Blockchain via Metamask wallet

```
const ConnectToMetamask = async () => {
  if (window.ethereum && window.ethereum.isMetaMask) {
    await window.ethereum
      .request({ method: "eth_requestAccounts" })
      .then((result: any[]) => {
        localStorage.setItem("account", result[0]);
        setDefaultAccount(result[0]);
        accountChangeHandler(result);
      });
  } else {
    console.log("Install Metamask");
    setPopupInstall(true)
  }
};


const accountChangeHandler = async (newAccount: any) => {
  setDefaultAccount(newAccount);
  localStorage.setItem("account", newAccount);
  console.log(localStorage.getItem("account"));
};
```

B. Generate AI image by Prompt, seed and Guidance Scale through Replicate API

```
export default async function handler(req, res) {

    const {prompt, guidance_scale, seed} = req.body;

    const response = await fetch("https://api.replicate.com/v1/predictions", {
      method: "POST",
      headers: {
        Authorization: `Token ${process.env.REPLICATE_API_TOKEN}`,
        "Content-Type": "application/json",
      },
      body: JSON.stringify({
        // Pinned to a specific version of Stable Diffusion
        // See https://replicate.com/stability-ai/stable-diffussion/versions
        version: "6359a0cab3ca6e4d3320c33d79096161208e9024d174b2311e5a21b6c7e1131c",

        // This is the text prompt that will be submitted by a form on the frontend
        input: { prompt: req.body.prompt },
      }),
    });

    if (response.status !== 201) {
      let error = await response.json();
      res.statusCode = 500;
      res.end(JSON.stringify({ detail: error.detail }));
      return;
    }

    const prediction = await response.json();
    res.statusCode = 201;
    res.end(JSON.stringify(prediction));
  }
```

C. Client

```
const handleSubmit = async (e:React.FormEvent<HTMLFormElement>) => {
  e.preventDefault();
  const formData = new FormData(e.currentTarget);
  const {prompt, seed, guidanceScale} = Object.fromEntries(formData.entries()) as {prompt:string; seed:string ; guidanceScale:string};
  const response = await fetch("/api/predictions", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify({
      prompt,
      seed:parseInt(seed),
      guidanceScale: parseInt(guidanceScale)
    }),
  });
  let prediction = await response.json();
  if (response.status !== 201) {
    setError(prediction.detail);
    return;
  }
  setPrediction(prediction);

  while (
    prediction.status !== "succeeded" &&
    prediction.status !== "failed"
  ) {
    setLoading(true)
    await sleep(1000);
    const response = await fetch("/api/predictions/" + prediction.id);
    prediction = await response.json();
    if (response.status !== 200) {
      setError(prediction.detail);
      return;
    }
    console.log(prediction)
    setPrediction(prediction);
  }
  if(prediction.status == "succeeded")
  {
    setLoading(false)
    setAnh(true)
  }
};
```

D. Save data image by UID to Firebase Realtime Database

```
const SaveImage =()=>{
  if(localStorage.getItem("account") == null)
  {
      setPopupInstall(true)
      setAnh(false)
  }
  else
  {
      if(prediction == null) return
      var id = getRandomInt(1,10000)
      set(ref(db, 'users/' + localStorage.getItem("account") + '/images'+ '/' + id), {
        image : prediction.output[0]
      })
      .then(()=>{
          setDoneSave(true)
          setAnh(false)
      })
      .catch((error)=>{
          console.log(error);
      });
  }
}
```

E. Load image by UID wallet address from Firebase Realtime Database to client Web

```
get(child(ref_database(db), 'users/' + localStorage.getItem("account") +'/images' )).then((snapshot) => {
    if (snapshot.exists()) {
        setPn(Object.values(snapshot.val()))
        console.log(pn)
    } else {
      console.log("No data available");
      setPn([])
    }
  }).catch((error) => {
    console.error(error);
  });
```

F. Update name, avatar using Firebase Storage and Firebase Realtime

```
const submitAva = ()=>{
  if (file == null) {
    return;
  }
  else if(localStorage.getItem("account") == null)
  {
    setPopupInstall(true)
    return;
  }

  setLoading(true);
  var uid = localStorage.getItem("account");
  const fileRef = ref(storage,`/image/${uid}`);
  uploadBytes(fileRef, file)
        .then((snapshot) => {
          getDownloadURL(snapshot.ref)
            .then((url) => {
                  update(ref_database(db, `users/`+ uid) ,{
                    urlImage : url,
                  }).then(()=>{
                      setSuccess(true);
                      setBlob("")
                      setLoading(false);
                    })
                    .catch((error)=>{
                        console.log(error);
                    });
            })
            .catch((err) => {
              console.log("get file url failed:", err);
            });
        })
        .catch((err) => {
          console.log("upload file failed:", err);
        });
}
```

G. Export URL to share

```
const shareImage = ()=>{
  navigator.clipboard.writeText(urlImage)
  setDetail(false)
  setCopy(true)
}
```

H. Save (download) image to local device

```
const download = (filename:any, content:any) => {
  var element = document.createElement("a");
  element.setAttribute("href", content);
  element.setAttribute("download", filename);
  element.style.display = "none";
  document.body.appendChild(element);

  element.click();

  document.body.removeChild(element);
};
```

```
const handleDownload = async (e:any) => {
  if(urlImage == linkUrl )
  {
    try {
      const result = await fetch("1.png", {
        method: "GET",
        headers: {}
      });
      const blob = await result.blob();
      var url = URL.createObjectURL(blob);
      console.log(url)
      download("image", url);
      URL.revokeObjectURL(url);
    } catch (error) {
      console.error(error);
    }
  }
  else
  {
    try {
      const result = await fetch("2.jpeg", {
        method: "GET",
        headers: {}
      });
      const blob = await result.blob();
      var url = URL.createObjectURL(blob);
      console.log(url)
      download("image", url);
      URL.revokeObjectURL(url);
    } catch (error) {
      console.error(error);
    }
  }
```

VI. Advanced Software Architecture

A. Blockchain

- Using blockchain Metamask wallet to connect blockchain network.
- To utilize the wallet address as a UID to save an image to the Firebase Realtime database, we obtain the wallet address and sign the wallet.
- A popular browser extension wallet for communicating with Ethereum-based apps is called MetaMask. It manages the intricacy of key management, transaction signing, and connection with the Ethereum network while offering an easy-to-use interface. The MetaMask API allows developers to include MetaMask into their online applications.

B. AI Generate Image

- An open-source Python package called Transformers offers a standardized user interface for language models. These models are pre-trained on extensive text corpora and may be tailored to particular applications with comparatively little training data.
- In addition, Transformers includes models such as Longformer, BERT, and RoBERTa, which are typically applied to more conventional NLP tasks like named entity identification, categorization, and so forth. Both types of models can use the procedure we're going to go over here; in fact, every Transformers model should be able to use it.
- A prediction is made each time a model is run. A prediction is an object that contains all of the information about a single outcome of running the model, including the inputs you gave, the outputs the model produced, and additional metadata like the model's version, the creator's name, the prediction's status, and timestamps.
- A forecast is produced each time a model is performed. Certain models exhibit high computational speed, yielding results in a matter of milliseconds. Certain models, particularly generative models—that is, models that generate pictures in response to text prompts—may require more time to execute. You must query the API for these persistent models in order to find out how each forecast is doing.

VII. Releases

Google Drive Link:
https://drive.google.com/drive/folders/1j4in7rp1jcYGaEYe7TTV5c4pzOrV6Eee?fbclid=IwZXh0bgNhZW0CMTAAAR25oJyepCHG9pg395W3xtOES4H5vcml6p0Fqmq6K38ggRx1hnEXILNnpC4_aem_AaCNonYXsJsPXgG0KFfqWixAuLZn-mVCi3EBFsLBHe-BPLoiJHfE2CwqK5_hamxGCGybXEN5ByfihSS6wwzMCPXi

Deployed URL: https://cs-422-diffusion-web.vercel.app/AI

VIII. References

- https://firebase.google.com/docs?hl=vi
- https://replicate.com/docs
- https://nextjs.org/docs
- https://tailwindcss.com/docs/installation