StableAI

CMPE 492
Final Report
05.06.2023

Arda Atakol 13624005832
Serdar Hoşver 10061964142
Furkan Özelge 14758028780
Yağız Hikmet Karakuş 44164883604

URL of the webpage: https://furkanozelge.github.io/seniorstedu/

Table of Contents

1. Introduction

Background information on the project

The StableAI project is focused on developing an innovative application called Stable Diffusion, which harnesses the power of deep learning and artificial intelligence. This application offers three main functions: text-to-image conversion, image-to-image transformation, and style transfer.

With the text-to-image conversion feature, users can input text descriptions and generate corresponding visual representations. For instance, by simply typing a phrase like "a photograph of an astronaut riding a horse," the application can produce a realistic image depicting exactly that. The text-based prompts serve as a foundation for Stable Diffusion to create detailed and visually appealing images.

The image-to-image transformation function allows users to modify existing images by applying a wide range of transformations. By specifying desired alterations such as background changes, object addition or removal, or color adjustments, users can bring their creative visions to life. Deep learning algorithms within the application enable seamless and accurate image modifications.

Furthermore, Stable Diffusion provides the style transfer capability, enabling users to apply artistic styles from one image to another. Users can select a source image with a specific style and a target image that they wish to transform. Leveraging deep learning techniques, the application extracts the artistic style from the source image and applies it to the target image, resulting in a visually transformed output that merges the chosen style with the original image.

In addition to these core functions, the StableAI project recognizes the significance of social sharing. Users have the ability to share their created images within the application, allowing them to showcase their designs, inspire others, and foster a creative community.

Overall, the StableAI project aims to deliver a user-friendly and versatile application that leverages the capabilities of deep learning and AI. By providing functionalities such as text-to-image conversion, image-to-image transformation, style transfer, and social sharing, Stable Diffusion empowers users to explore their creativity, generate captivating visuals, and connect with like-minded individuals within the application's community.

Objectives and scope of the project

The objectives of the StableAI project are as follows:

1. Develop an application that utilizes stable diffusion, deep learning, and artificial intelligence to convert text descriptions into visually appealing images.
2. Provide users with a user-friendly interface for easily inputting text prompts and generating corresponding images.
3. Enable users to modify existing images through various image-to-image transformations, allowing for creative experimentation and customization.
4. Implement style transfer functionality to apply artistic styles from one image to another, giving users the ability to create unique visual effects.
5. Facilitate social sharing within the application, allowing users to showcase their created images and foster a creative community.
6. Ensure the application's stability, performance, and reliability through comprehensive testing and quality assurance processes.
7. Prioritize user privacy and data security by implementing robust security measures to protect user information.

The scope of the project includes:

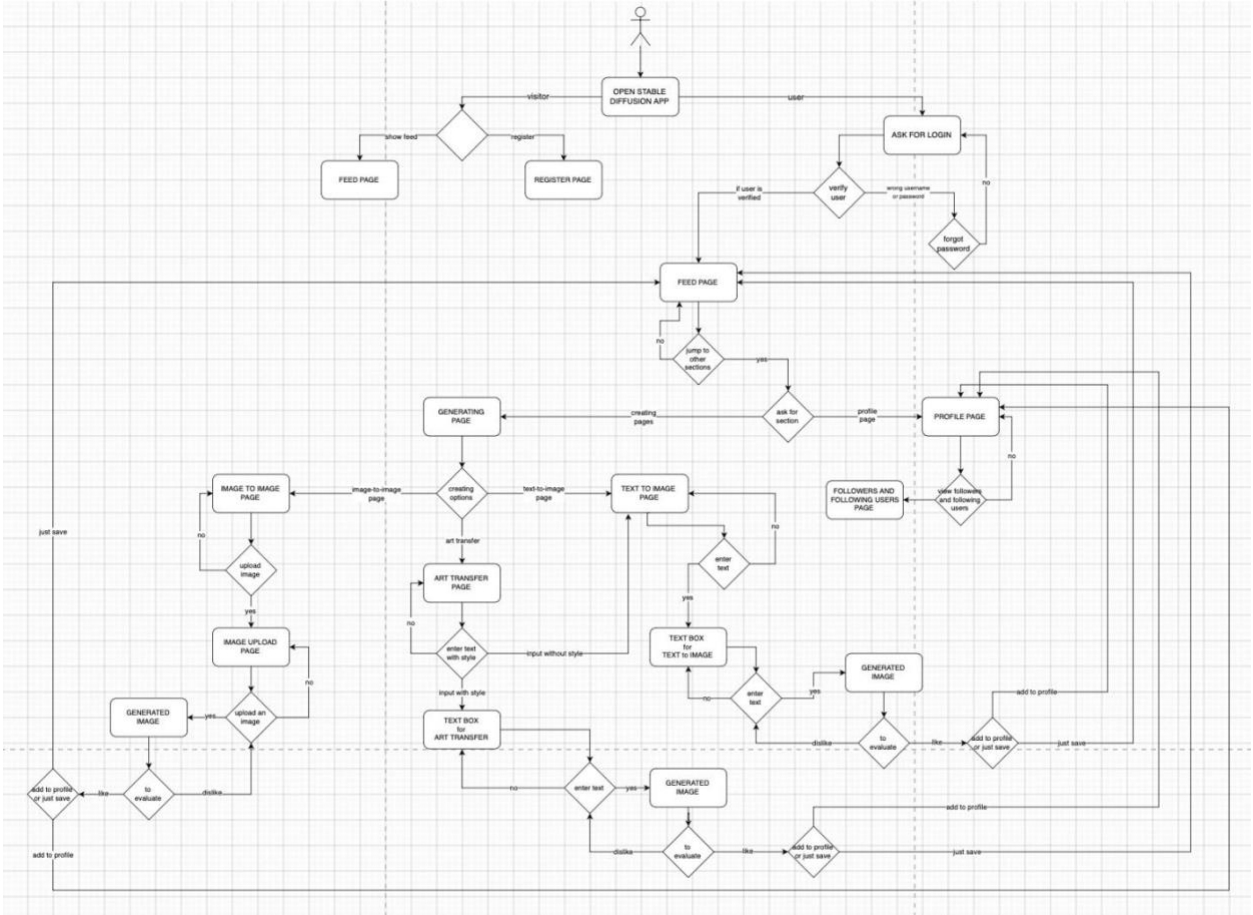
1. Developing the Stable Diffusion application with the three main functions: text-to-image conversion, image-to-image transformation, and style transfer.
2. Designing a user-friendly interface that enables users to input text prompts, select images for transformations, and share their creations.
3. Implementing stable diffusion algorithms and deep learning models to generate high-quality images based on text prompts.
4. Integrating image-to-image transformation capabilities, allowing users to perform various modifications and adjustments on existing images.
5. Enabling style transfer functionality to apply artistic styles from one image to another, enhancing creative possibilities.
6. Incorporating social sharing features to allow users to share their created images within the application's community.
7. Conducting thorough testing and quality assurance procedures to ensure the stability, performance, and reliability of the application.
8. Implementing strong security measures to protect user data and ensure privacy.
9. Providing regular updates and bug fixes to improve the application's functionality and address user feedback.

The project aims to deliver a comprehensive and user-centric application that empowers users to create visually appealing images, explore their creativity, and connect with others in a creative community.

Overview of the report structure

The StableAI final report is structured in a comprehensive manner to provide a clear understanding of the project's development, testing, and overall outcomes. The report is divided into several sections, each focusing on a specific aspect of the project.

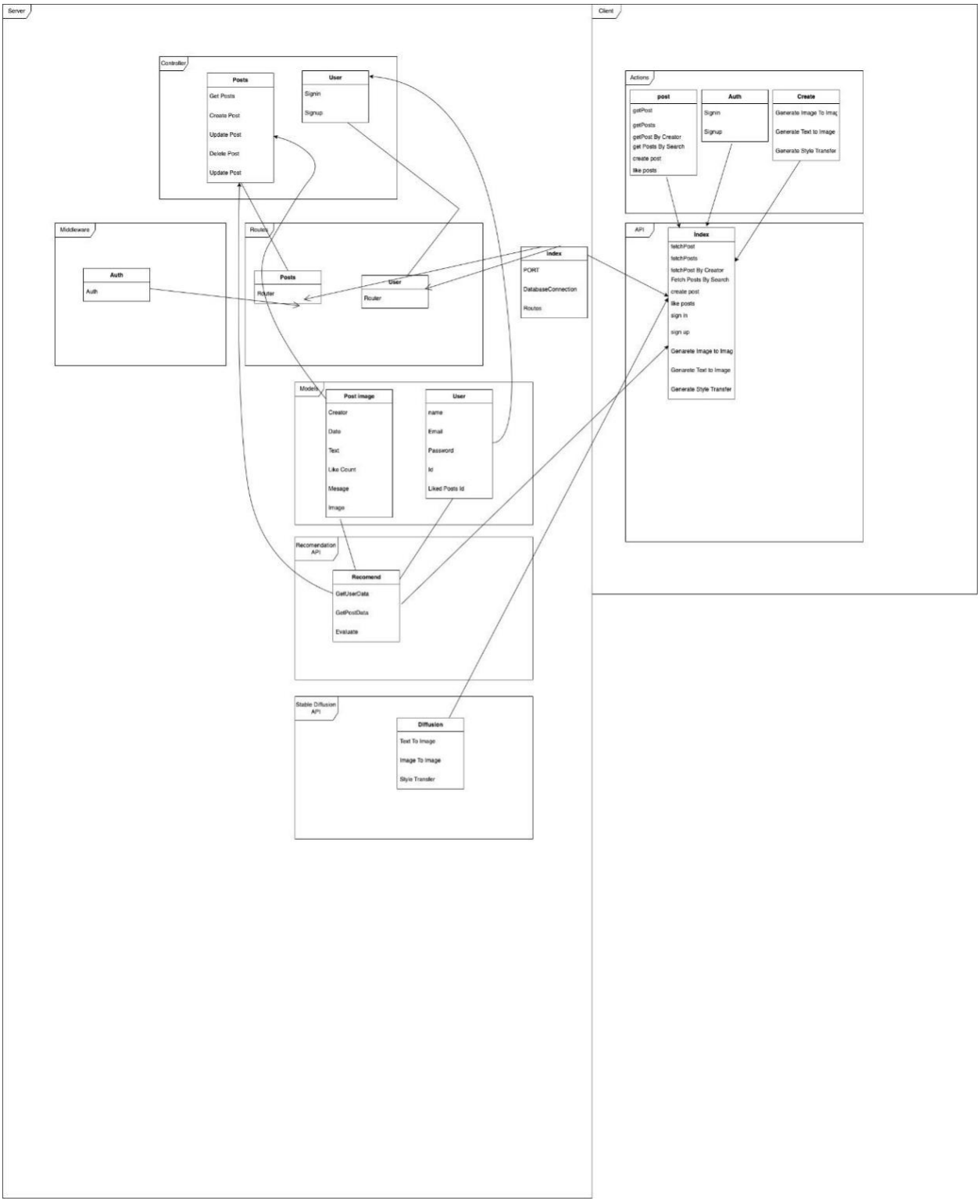
Purpose of the system



A UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system. In our UML diagram, we have an actor which is using Stable Diffusion Application. First the actor open the app and he/she has to choose the entry type as a visitor or as an user. If he/she is a visitor, app direct the user to the visitor page and let him/her to select the path register page or feed page. If he/she open the app as an user, app ask her/him for login. If he/she is an user, user must

verify user. If the username or password is wrong, user direct to the forgot password section. Otherwise, user direct to the feed page. In this point, user see the sections and he/she has to select one of these 2 sections. These are generating page and profile page. In profile page, user can view the followers and following users. In the other section (Generating Page), user has some creating options which are Image to Image, Text to Image and Art Transfer page. In Image-to-Image part, first the user uploads an image to the image upload page. After that the program gives the generated image to the user to evaluate. If the user like the generated image, he/she

can add it to the profile or just save it. If the user doesn't like the generated image and dislike it,program returns to upload an image part. Suppose that the user liked the generated image. In this point he/she has two options which are just save and add to profile. If the user just saves the generated image, program direct the user to the feed page. Otherwise, if the user selects the add to profile option, the program directs the user to the profile page. Our second creating option is Art Transfer. In this page, user enters text with style. If the input and the style are not valid, program direct the user to the text to image option. Suppose that the user entered valid text with style. In that point user goes to Text Box for Art Transfer. After that user enters a text and program shows the generated image for him/her to evaluate. Again, the program gives the user 2 options as a like or dislike. If the user dislike the generated image, it returns to enter text part. But if the user like the image, he/she may add to profile this image or just save it same as Image-to-Image part. If the user wants to add the generated image to profile, it directs to profile page. Otherwise, it directs to feed page. The last option is text to image part. In this option, user enters a text and program directs the user to the text box for text to image. User enters a text and see the generated image to evaluate. Remaining part is same as other options. If the user dislikes it, it returns to enter text part, if the user like the generated image, he/she see the question, add to profile or just save? If the user want to add the image to the profile program directs the user to the profile page and if the user want just save the image, program directs the user to the feed page.


Server
Controller
Posts
Get Posts
Create Post
Update Post
Delete Post
Update Post
User
Signin
Signup
Middleware
Auth
Auth
Routes
Posts
Router
User
Router
Index
PORT
DatabaseConnection
Routes
Models
Post image
Creator
Date
Text
Like Count
Mesage
Image
User
name
Email
Password
Id
Liked Posts Id
Recomendation API
Recomend
GetUserData
GetPostData
Evaluate
Stable Diffusion API
Diffusion
Text To Image
Image To Image
Style Transfer
Client
Actions
post
getPost
getPosts
getPost By Creator
get Posts By Search
create post
like posts
Auth
Signin
Signup
Create
Generate Image To Imag
Generate Text to Image
Generate Style Transfer
API
Index
fatchPost
fatchPosts
fatchPost By Creator
Fetch Posts By Search
create post
like posts
sign in
sign up
Genarete Image to Imag
Genarete Text to Image
Generate Style Transfer

2. Subsystem services

Server

1. docs():

The docs() endpoint is a GET endpoint with the path '/'. It is used to redirect users to the API documentation page. When a request is made to this endpoint, it returns a RedirectResponse that redirects the user to the '/docs' path where the API documentation is located. This helps users easily access the documentation to understand the available API endpoints and their functionalities.

2. create_user:

The create_user endpoint is a POST endpoint with the path '/signup'. It is used to create a new user. When a request is made to this endpoint, it expects a UserAuth object in the request body. This object contains user authentication details such as name, surname, username, profile picture, email, and password.

The endpoint first checks if a user with the given email already exists in the database. If a user with the same email is found, it indicates that the user already exists and returns an appropriate response indicating the failure to create a new user.

If the user doesn't exist, the endpoint proceeds to create a new user document. It saves the provided user details into the usersdb collection in the MongoDB database. Additionally, it hashes the user's password for security purposes before saving it.

Finally, the endpoint returns the created user document as a response, providing confirmation to the user that the account has been successfully created.

3. login:

The login endpoint is a POST endpoint with the path '/login'. It is used for user authentication and generating access and refresh tokens. When a request is made to this endpoint, it expects the user's username and password as form data.

The endpoint checks if a user with the given username exists in the database. If the username is not found, it returns a response indicating a failed login attempt.

If the username is found, the endpoint proceeds to verify the provided password. It compares the hashed password stored in the database with the password provided in the request. If the password is incorrect, it returns a response indicating an authentication failure.

If the username and password are correct, the endpoint generates an access token and a refresh token using the create_access_token() and create_refresh_token() functions respectively. These tokens are used for subsequent authentication and authorization purposes.

Finally, the endpoint returns the access and refresh tokens as a response, providing the user with the necessary tokens for accessing protected routes and performing authorized actions.

4. get_me:

The get_me endpoint is a GET endpoint with the path '/me'. It is used to retrieve the details of the currently logged-in user. To access this endpoint, the user needs to be authenticated and authorized, which is achieved by using the get_current_user() dependency.

Once the user is authenticated and authorized, the endpoint retrieves the user details from the SystemUser object obtained from the dependency. This object contains information such as the user's name, surname, username, profile picture, and email.

Finally, the endpoint returns the user details as a response, providing the logged-in user with their own information.

5. share_photo:

The share_photo endpoint is a POST endpoint with the path '/share'. It is used to share photos. When a request is made to this endpoint, it expects a Post object in the request body. This object contains the photo's prompt, the user's email, and the image in base64 string format.

The endpoint inserts the Post document into the post collection in the MongoDB database, which represents a shared photo. The document includes details such as the prompt, the user's email, and the image in base64 format.

If the insertion is successful, the endpoint returns a success message as a response, indicating that the photo has been successfully

6. Generate Image API Method:

The "Generate Image" API method receives a GET request with a text prompt and a scale value. It uses the prompt to generate an image by utilizing the Stable Diffusion Pipeline. The scale parameter adjusts the guidance scale for the generation process, influencing the level of

influence the prompt has on the generated image. The method returns the generated image in PNG format, encoded in base64.

7. Image-to-Image Transformation API Method:

The "Image-to-Image Transformation" API method handles a POST request that includes an input image, a text prompt, and a scale value. It transforms the input image based on the provided prompt using the Stable Diffusion Image-to-Image Pipeline. The method returns the transformed image as a response, encoded in base64.

8. Neural Style Transfer API Method:

The "Neural Style Transfer" API method processes a POST request containing a content image and a style image. It applies the style of the given image to the content image using a pre-trained model from the Magenta project. The method returns the stylized image as a response, encoded in base64.

9. Helper Functions:

The load_img function decodes a base64-encoded image, performs preprocessing such as resizing and normalization, and returns the preprocessed image as a TensorFlow tensor.
The tensor_to_image function converts a TensorFlow tensor representing an image back to a PIL Image object.
The neuralstyle function performs neural style transfer on a content image using a style image. It utilizes a pre-trained model from the Magenta project. The stylized image is returned as a PIL Image object.
These API methods, along with the helper functions, enable users to generate images based on prompts, perform image-to-image transformations, and apply style transfer to content images. The server-side implementation handles the requests from the frontend, processes the images using appropriate pipelines or models, and sends back the processed images in base64-encoded format as responses.

Client

On the client side, we aimed to maximize the user experience and to establish communication with the server as quickly as possible. For the user experience on the client side, we have coded a very good UI and added features that will allow users to spend a long time on our website.

There are many processes going on in the client part. First of all, if we talk about the JWT part, we have two pages called sign-in and sign-up. Users first create their memberships in the sing-up section. In this part, we get one e-mail and one password from the user. Then we send it to the api with the signUp function in our api.js file. We use Axios to do this because NextJS's and ReactJS' own fetch function may not work very well. Axios provides us both a much better performance and a much better speed. In this part, we send the userData and headers to the /signup part of our api. Since we use ngrok in the API part, we also send the "'ngrok-skip-browser-warning': '69420'" header as the recommended header to avoid CORS error. In this way, we create our user in the database with the user information we send to the API.

In the sign-in section, the user enters his e-mail address and password and presses the login button. After this process, the login function in our api.js file works and we send our request to the /login route of our api. Meanwhile, we send both the password and e-mail of the user to the backend and wait for the response in the api. Meanwhile, there is a JWT control in the backend and an access_token is returned to us. This returning access_token shows us that the user's information is correct. We save this in the cookies of the user's browser with the help of the Cookies library and automatically redirect the user to the /profile route. In this way, the user logs into the site.

In addition, we check the cookies of the user when he tries to enter the routes and addresses of the entire site, and see if he is logged in and if he is logged in, we take him to the page he wants to enter. In case he is not logged in, we are automatically redirected to the login page. In this way, we prevent the use of our site without logging in. We do this thanks to getProfile. In the getProfile function, we send the user's token to the API and check the access_token.

Apart from JWT, there are many processes on the client side. First of all, in our text2img feature, we get a prompt from the user in the part where we aim to generate images from the text prompts of the users. At the same time, we get our scale property with the help of slider. Then we send the sliderValue and prompt to the api with a get request. Then the base64 code of the generated image is returned to us. We convert this base64 code to image in the text2img page and show it to the page.

In addition, we receive one image and one prompt from the user on our img2img page. In this process, we first receive a prompt as text from the user and ask him to upload an image. Here we convert the uploaded image to base64 code. After that, we send this base64 code and prompt by sending a post to the api for the img2img generation process. Then we get the base64 code of the generated image from the api as a response. We show this to the user on our img2img page.

In addition to these, we ask the user to upload two images on our style-transfer page, which is another feature of ours. If he wants, he can choose one of the 3 tables we recommend. Then we send the base64 code of the two images uploaded to the api to generate. Afterwards, we show the generated base64 code returned to us on our page. Responses from API are always sent with base64 because it is much less costly and faster than sending images.

Apart from these, we have a bookmark feature on the client side. With this feature, users can bookmark the images they generate and share them with all other users, and they can also see them in the "My Arts" section of their profile. It uses this feature on the client side for the generated images that it will use later. In this feature, when the user presses the bookmark button for the generated image, we send the base64 code, which the user generates with the e-mail address where the access_token is checked with the getProfile function, as a post. Then these images are kept in the database depending on an e-mail address. This allows the user to see the art he has saved when he enters his profile page. At the same time, it can make all users see it on the explore page.

3. Implementation

Frontend

We used many different technologies on the frontend. We used the ReactJS framework, which can use Javascript in the Frontend area and outputs HTML. However, NextJS, which is a sub-framework of ReactJS, offers us many different features. It also eliminates many of the problems with ReactJS. It fixes many ReactJS problems like Routing, Image. For this reason, we decided to implement our project with NextJS.

In addition to NextJS, we used many libraries belonging to NextJS, such as next/router and next/image. We implemented the router part by setting up the pages folder structure in the frontend part. This gave us a very good performance and we had a much better folder structure.

We decided on ChakraUI, which is the most popular UI library among many libraries that can help in the UI part of NextJS, and by using ChakraUI, we have handled many CSS and style parts of the site in a much more responsive and well-performed way.

We organized the NextJS folders in our StableAI project as follows. We have Public, Src, Utils and node_modules folders. The node_modules folder is the folder that contains all the libraries we use from the node library, which is already known to everyone. In addition, in our public folder, we have various images that we need to access from all pages, our favicon and, in addition, our logo.

In our "src" folder, which is the most important part, we have pages and components folders. In the Components section, we keep various component structures that we use on multiple pages, which will generally be useful everywhere. For example, our Cards, Footer, HomePage, Navbar and UnNavbar components.

The Cards component creates a beautiful UI with various sample prompts and images that we use on the homepage. The footer component works as a page break component that covers the content that comes to the end of the page that we use on all pages. The HomePage component is actually a component structure that we use to not contain code blocks in index.js. Hosting code in index.js is generally not a very good structure. For this reason, we created a different component for HomePage and configured HomePage in this way. In addition, there are Navbar and UnNavbar sections. Navbar, on the other hand, is a component that aims to improve the user experience, which stands at the top of the entire page. Navbar is the Navbar that can be seen after the user login, and UnNavbar is the Navbar that the user can see without providing the authentication part without user login.

We also have all our pages in our Pages folder. These are our explore, feed, img2img, profile, sign-in, sign-up, style-transfer, text2img pages. On our Explore page, there is a section where we can see the photos and prompts generated by all users. All users can see each other's generated images here. Our page with a nice UI that directs the feed part to all our generate pages. In addition, we have img2img, text2img, style-transfer pages. As we mentioned in our report, these are the pages of our features where we can generate an image from an image, an image from a text and transfer the style of an image to another image. In addition to all these, we can perform registration and login transactions using JWT on sign-in and sign-up pages.

Apart from these, we have a folder in the utils folder where the functions of various signup signin getprofile operations that we have done in the api.js file are kept. This is a part we use specifically for JWT. We can also see our API link here.

Backend

We wrote 2 different APIs in the back-end part of the project. We can separate them as image generation and CRUD APIs. These two APIs work separately from each other and they have separate tasks. The main reason for this is that our image generation models consume a lot of resources (GPU), so our image generation API works faster and because it works independently from the CRUD API, the processes do not conflict. We used fastapi in both apps, the main reason for this was that we were running our models over python. We used the uvicorn library for the apps to run on the local server and we preferred the ngrok library to open our localhost to the internet. Thanks to ngrok, we were able to share the applications running in our locale with the frontend. The ngrok library gives us a link when we run it, so our APIs can receive and respond to requests via this link.

Our generation api uses custom models in hugging face for text to image and image to image, we use transformers model for this, and because we also use diffusion model to work in other configurations, we imported StableDiffusionPipeline and StableDiffusionImg2ImgPipeline classes from diffusers library. We used base64 methods. At the same time, we have added this library to our code because the diffusers pipeline we use uses pytorch. We used tensorflow library for Image Style Transfer model and added tensorflow_hub library to download our model. We wrote our methods as asynchronous and we got help from the nest_asyncio library for the nested async methods to work. We used the pydantic BaseModel class to create the request models we received. We created our model classes as the child class of this BaseModel class and we took advantage of the inheritance feature of the programming language we use (python).

In our CRUD API, our file layout is different. In the app.py file, we have our functions for get and post methods and our database connection. In this file, we used the HTTPException class of fastapi. Our purpose was to suppress and resolve the errors that occur in the request in the log. At the same time, thanks to OAuth2PasswordRequestForm, which is the buildin authorization class of fastapi, we increased our security and we did not have to create our own model for authorization schemas We also used the models such as serOut, UserAuth, TokenSchema, SystemUser, Post that we created in the .app file while receiving requests or sending responses, and we used the CORSMiddleware class in order to reach our API from anywhere and not get cors errors. We wrote the deps.py file to check the jwt tokens we give to the users and we added the Depends method from fastapi to check it in app.py. In utils.py, we wrote our methods to create jwt acces token and hash passwords, we used the HS256 algorithm to generate acces tokens, and to hash our passwords, we got help from the CryptContext class from passlib. We used the os library to access the JWT secret key and JWT reference secret keys. We used the jose library for jwt encode work. We used pydantic for request and response fields similar to our generation api in the schemas.py file and created our own classes by inheriting from BaseModel.

Data-Base

MongoDB: MongoDB represents a type of database known as NoSQL. Unlike traditional SQL databases, MongoDB is document-based. That is, it stores data as documents, and these documents are typically stored in JSON format. MongoDB is designed for high scalability and performance, and is widely used for large data storage and analysis.

Base64: Base64 is a coding scheme used to represent binary data in text form. Text is more easily and securely transmitted over the network, and so binary data is often sent in this format. For instance, when sending an image with an email, the image is usually encoded into Base64 format.

JWT (JSON Web Token): JWT represents a standard used for secure information exchange between two parties. A JWT is typically used to authenticate a user's identity. A server creates a JWT after it has authenticated a user's identity and sends it back to the user. The user then sends this token back to the server with each request, which allows the server to authenticate the user's identity.

Speaking in more detail about our project:

In our MongoDB database named "StableAIdb", you have created two tables (in MongoDB these tables are called "collections"), named "post" and "user".

The "post" collection contains three fields named "prompt", "mail", and "image". "Prompt" and "mail" are probably each a type of text data. The "image" field holds image data and this data is stored in base64 format. That is, the image file is converted from binary format to text format and is stored in this way in MongoDB.

The "user" collection contains six fields: "name", "surname", "username", "profilpicture", "email", and "password". These fields represent a user's information. The "profilpicture" field is also likely holding image data in base64 format.

When users log in to your application, they receive a JWT to authenticate their identities. This token is then used when sending each request to the server, allowing the server to authenticate the user's identity. This method enables the user's identity to be authenticated without constantly sharing identity information with the server. This is important for security and also improves the user experience.

Architecture and Design

Features

Text to Image Page

- Visuals will be produced with the entered key Word.
- At the top of the page, there will be a text box for entering key Word or sentences
- There will be generate button under the text box.
- After clicking the generate button image will be generated and user routed to result page

Image to Image Page

- With an image to be uploaded to the system, new visuals will be produced based on that image.
- At the top of the page, there will be an upload area to upload sample images.
- It can be found on the Internet and a sample image can be uploaded with a link.
- An image in the computer can be uploaded to the system as an example visual.
- After clicking the generate button stable diffusion API will be executed, image will be generated, and user routed to result page

Transfer Art Page

- New visuals will be produced based on the input entered based on the visuals of other users or based on the style of a famous art.

- There will be an area on the page where the style art is uploaded to the system.
- There will be an area for the content image which is going to change.
- If user click the transfer button our style transfer API I is going to executed and user routed to result page.

Description of components

Component 1: Text Processor

The Text Processor class is responsible for processing the input text and converting it into a format that can be understood by the Image Generator. The following is a description of the interface for the Text Processor class:

Input: A string representing the user's input text.
Output: A list of keywords extracted from the embedded text.

The Text Processor class will extract relevant keywords from the input text that will be used by the Image Generator to generate the corresponding image.

Component 2: Image Generator

The Image Generator class is responsible for generating the image based on the input keywords provided by the Text Processor. The following is a description of the interface for the Image Generator class:

Input: A list of keywords extracted from the input text.
Output: An image corresponding to the input keywords.

The Image Generator class will use deep learning algorithms to generate a detailed image that corresponds to the input keywords provided by the Text Processor. The output image will be displayed to the user via the application's user interface.

Component 3: Data Loader

The Data Loader class is responsible for loading the pre-trained open-source data required for the Style Transfer component to generate images. The following is a description of the interface for the Data Loader class:

Input: User’s uploaded images
Output: A pre-trained model for generating images using style transfer.

The Data Loader class will load the pre-trained data required by the Style Transfer component to generate images based on the user's input.

Component 4: Style Transfer

The Style Transfer class is responsible for generating an image based on the user's input and the pre-trained model loaded by the Data Loader component. The following is a description of the interface for the Style Transfer class:

Input: A image representing the user's input image.
Output: An image corresponding to the user's input image generated using style transfer.

The Style Transfer class will use deep learning algorithms to generate an image that corresponds to the user's input image based on the pre-trained model loaded by the Data Loader component.

Component 5: Sign Up

The Sign Up class is responsible for creating a new user account in the system. The following is a description of the interface for the Sign Up class:

Input: A string representing the user's email address and a string representing the user's password.
Output: JSON Web Token.

The Sign Up class will create a new user account in the system using the provided email address and password.

Component 6: Login

The Login class is responsible for verifying a user's credentials and allowing access to the system. The following is a description of the interface for the Login class:

Input: A string representing the user's email address and a string representing the user's password.
Output: A boolean value indicating whether the login checking with JSON Web Token was successful.

The Login class will verify the user's credentials and allow access to the system if the provided email address and password are correct.

Component 7: Image Sharing

The Generated Image Sharing class is responsible for allowing users to share generated images with other users in the system. The following is a description of the interface for the Generated Image Sharing class:

Input: An image generated by the Style Transfer or text-to-image/image-to-image component.
Output: None.

The Generated Image Sharing class will allow users to share the generated image with other users in the system.

Component 8: Guidance Scale Processor

The Guidance Scale Processor class is responsible for processing the guidance scales for the image generation process. The following is a description of the interface for the Guidance Scale Processor class:

Input: An integer for guidance scales.
Output: None.

The Guidance Scale Processor class will process the guidance scales for the image generation process to ensure that the generated image meets the user's expectations.

Component 9: Recommendation

The Recommendation To User Depends on User's Keywords class is responsible for providing recommendations to users based on their input keywords. The following is a description of the interface for the Recommendation To User Depends on User's Keywords class:

Input: A list of keywords.
Output: A list of recommended post.

The Recommendation To User Depends on User's Keywords class will provide recommendations to users based on their input keywords to improve their image generation experience.

Component 10: Upload Image

The Image Uploader class is responsible for uploading images to the Image Generator for image-to-image translation. The following is a description of the interface for the Image Uploader class:

Input: An image file.
Output: A message indicating whether the upload was successful or not.

The Image Uploader class will allow users to upload images that can be used as input for image-to-image translations by the Image Generator.

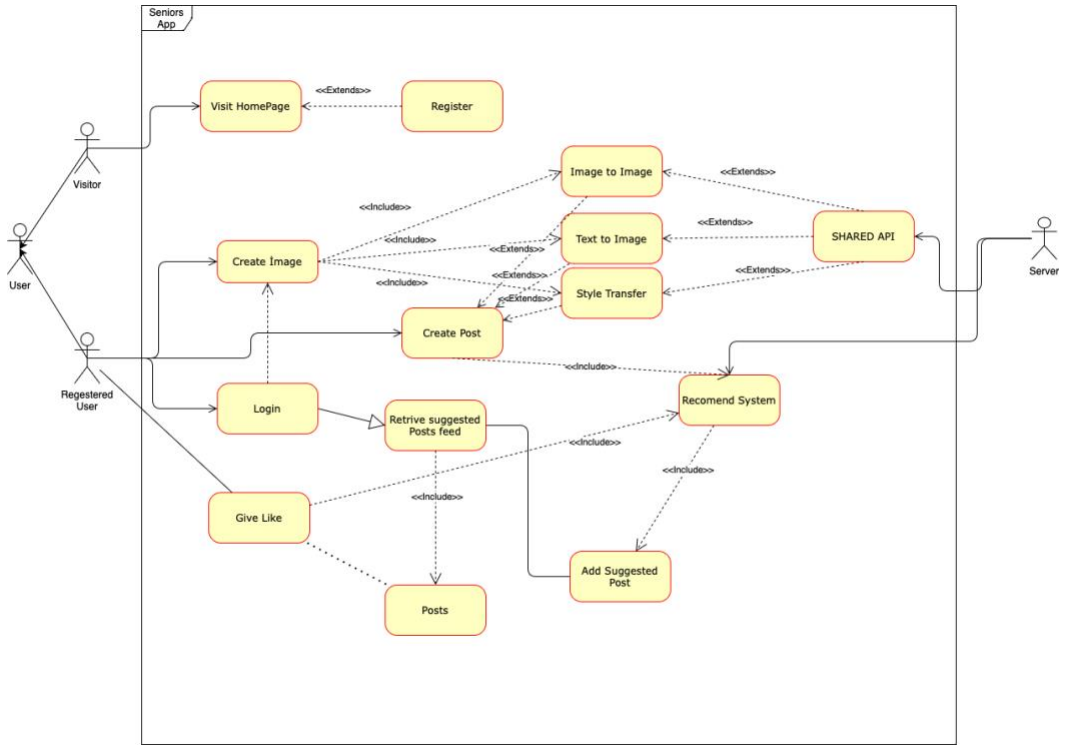
Component 11: Image Rating

The Image Rating class is responsible for allowing users to rate generated images as either like or dislike. The following is a description of the interface for the Image Rating class:
Input: An image ID and a rating (like or dislike).
Output: A message indicating whether the rating was successfully recorded or not(consol).

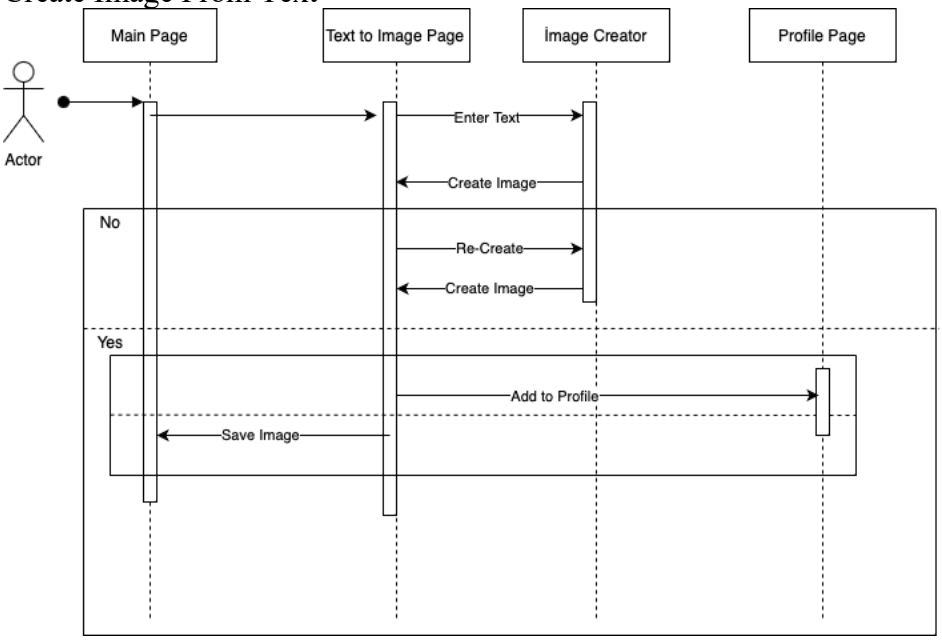
The Image Rating class will allow users to provide feedback on the generated images, which can be used to improve the system's performance in the future.

By following these class interfaces, the Stable Diffusion project will ensure that the different components of the system can communicate with each other and work together to provide a seamless user experience.

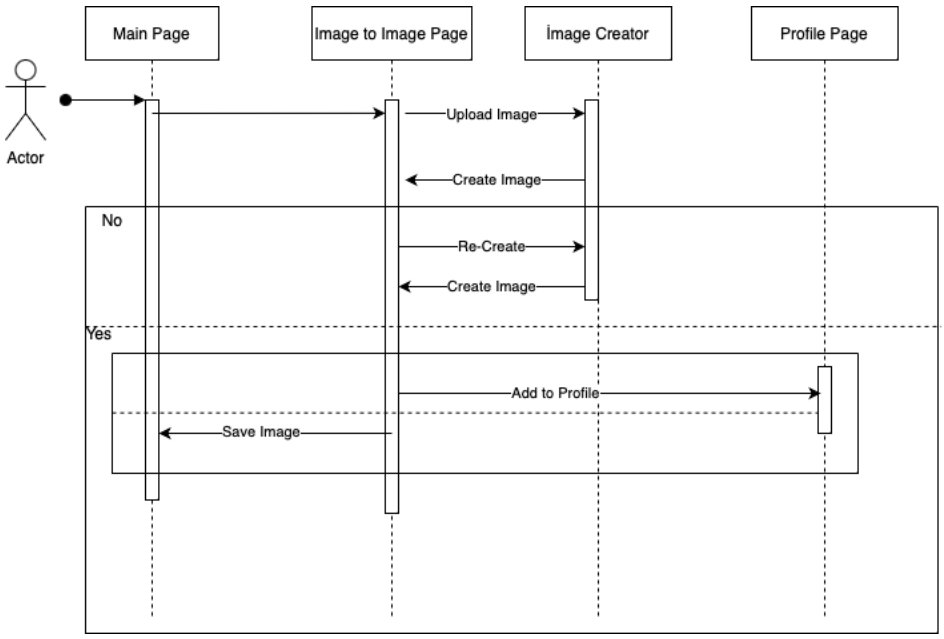
Subsystem decomposition



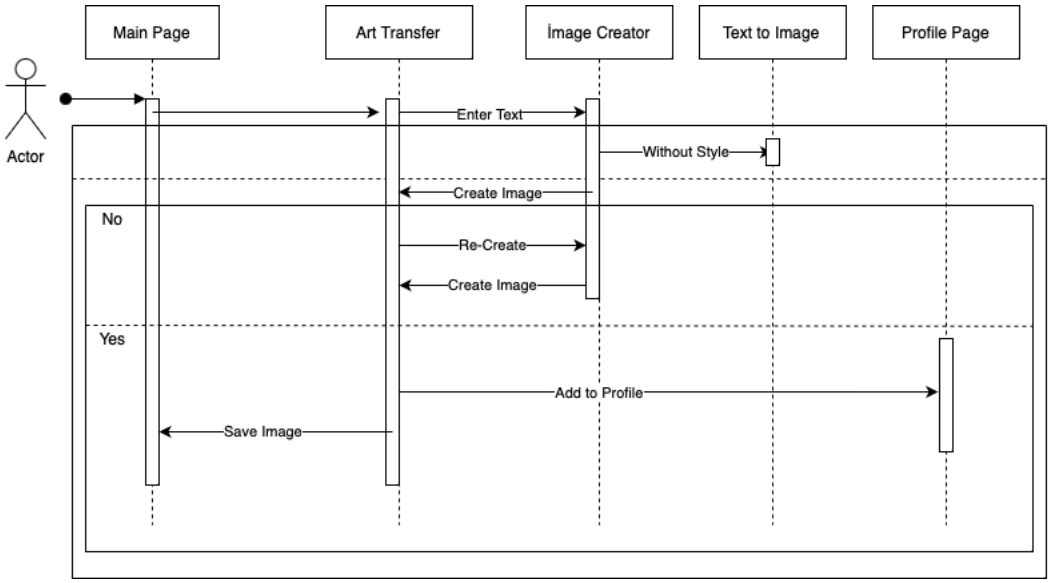
Create Image From Text



Create Image From Image



Create İmage From Art Transfer



4. Project Status

Summary of project progress

The StableAI project has made significant progress since its initiation. Over the course of several months, the project team has diligently worked on various aspects of application development, testing, and quality assurance to bring Stable Diffusion to life. Here is a summary of the project progress:

1. Initial Planning and Research:
    a. Conducted thorough research on stable diffusion, deep learning, and AI techniques to understand the underlying concepts and potential applications.
    b. Defined the project objectives, scope, and target audience.
    c. Developed a detailed project plan and timeline.
2. Application Development:
    a. Designed and implemented the Stable Diffusion application with a user-friendly interface.
    b. Integrated stable diffusion algorithms and deep learning models to enable text-to-image conversion, image-to-image transformation, and style transfer functionalities.
    c. Implemented social sharing features to allow users to showcase their created images within the application's community.

3. Testing and Quality Assurance:
    a. Conducted extensive testing to ensure the stability, performance, and reliability of the application.
    b. Performed functional testing to validate the core features and functionalities.
    c. Conducted usability testing to assess the user-friendliness and intuitive nature of the interface.
    d. Conducted security testing to ensure data protection and prevent unauthorized access.
    e. Performed performance testing to evaluate the application's responsiveness and scalability.
    f. Conducted reliability testing to validate its stability and robustness.
4. Iterative Development and Bug Fixes:
    a. Implemented necessary updates and bug fixes based on user feedback and testing results.
    b. Addressed any identified issues or defects to enhance the application's functionality and user experience.

c. Ensured the application meets the highest standards of quality and exceeds user expectations.

5. Evaluation and User Feedback:
    a. Gathered feedback from users to assess their experience with Stable Diffusion.
    b. Analyzed user feedback to identify areas of improvement and potential enhancements.
    c. Incorporated user suggestions and recommendations to enhance the application's capabilities and user satisfaction.
6. Documentation and Reporting:
    a. Prepared comprehensive documentation, including the test plan, final report, and user guides.
    b. Documented the development process, methodologies, and technical specifications.
    c. Provided clear and concise information on the project's background, objectives, scope, and outcomes.

Overall, the project has progressed successfully, meeting key milestones and objectives. The StableAI team has worked diligently to ensure the application's functionality, usability, security, performance, and reliability. The project has undergone thorough testing and quality assurance processes to deliver a stable and user-friendly application that empowers users to create visually appealing images and share their creations within the application's community.

Challenges and obstacles encountered

Throughout the development of the StableAI project, the team encountered several challenges and obstacles that required careful consideration and problem-solving. These challenges included:

1. Data Availability and Quality:

- Acquiring a diverse and high-quality dataset for training the deep learning models was a significant challenge. Finding relevant and annotated data that covered a wide range of image categories and styles required extensive research and data collection efforts.

2. Algorithm Complexity:

- Implementing stable diffusion algorithms and integrating them into the application was a complex task. Understanding the intricacies of stable diffusion and optimizing the algorithms for

efficient and accurate image generation posed technical challenges that required expertise in deep learning and AI.

3. User Experience Design:

- Designing an intuitive and user-friendly interface that accommodated the three main functions of StableAI was a challenge. Balancing the need for simplicity and functionality while ensuring a seamless user experience required iterative design and testing iterations.

4. Performance Optimization:

- Achieving real-time performance for image generation and style transfer was a challenge due to the computational demands of deep learning models. Optimizing the algorithms and implementing efficient processing techniques were essential to deliver a responsive and scalable application.

5. Security and Privacy:

- Ensuring the security and privacy of user data was a priority. Implementing robust security measures to protect user information, prevent unauthorized access, and comply with data protection regulations posed challenges that required careful implementation and testing.

6. Bug Fixes and Compatibility:

- Addressing and resolving software bugs and compatibility issues across different platforms and devices was an ongoing challenge. The team had to conduct thorough testing and debugging to identify and fix any issues that arose during development and ensure a smooth user experience.

7. Time Constraints:

- Meeting project deadlines within the allocated time frame required efficient project management and resource allocation. Balancing development, testing, and documentation tasks while ensuring quality outputs was a challenge that required careful planning and coordination

Solutions implemented to overcome challenges

To overcome the challenges and obstacles faced during the development of the StableAI project, the team implemented several solutions and strategies. These solutions were aimed at addressing specific challenges and ensuring the successful completion of the project. Here are some of the solutions implemented:

1. Data Availability and Quality:
    a. Conducted extensive research to identify and collect a diverse and high-quality dataset for training the deep learning models.
    b. Utilized data augmentation techniques to increase the variety and size of the dataset, ensuring better model performance.
    c. Collaborated with external resources and open-source datasets to supplement the available data and improve its quality.
2. Algorithm Complexity:
    a. Invested significant time in understanding the principles and intricacies of stable diffusion algorithms.
    b. Collaborated with experts in the field of deep learning and AI to gain insights and guidance on implementing the algorithms effectively.
    c. Conducted iterative testing and optimization to refine the algorithms and improve their performance and accuracy.

3. User Experience Design:
    a. Conducted user research and usability testing to understand user expectations and preferences.
    b. Iteratively designed and refined the application's interface to ensure ease of use and intuitive navigation.
    c. Incorporated user feedback and conducted iterative design iterations to enhance the user experience and address usability challenges.
4. Performance Optimization:
    a. Employed various optimization techniques, such as model pruning, quantization, and parallelization, to improve the performance and speed of the deep learning models.

b. Leveraged hardware acceleration technologies, such as GPU utilization, to expedite image generation and style transfer processes.
c. Conducted rigorous performance testing and profiling to identify and eliminate bottlenecks and optimize resource utilization.

5. Security and Privacy:
    a. Implemented robust security measures, including encryption of user data, secure authentication mechanisms, and secure data storage practices.
    b. Conducted regular security audits and vulnerability assessments to identify and address potential security risks.
    c. Ensured compliance with relevant data protection regulations, such as GDPR, to protect user privacy and data rights.
6. Bug Fixes and Compatibility:
    a. Established a thorough bug tracking and reporting system to document and prioritize reported issues.
    b. Conducted extensive testing across different platforms and devices to identify and resolve compatibility issues.
    c. Collaborated closely with users and gathered feedback to quickly address bugs and provide timely updates and bug fixes.
7. Time Constraints:
    a. Employed effective project management techniques, such as agile methodologies, to plan and allocate resources efficiently.
    b. Conducted regular progress tracking and milestone reviews to ensure timely completion of tasks.
    c. Prioritized tasks based on criticality and impact to ensure that the project's objectives were achieved within the allocated time frame.

Comparison of actual progress against initial plan/schedule

Throughout the development of the StableAI project, it is essential to compare the actual progress made with the initial plan and schedule to assess the project's adherence to timelines and milestones. Here is a comparison of the actual progress against the initial plan:

1. Development Phase:
    a. Initial Plan: The development phase was estimated to span 4-5 months, with each team member devoting a minimum of 10-15 hours per week.
    b. Actual Progress: The team adhered to the estimated timeline and allocated hours, completing the development phase within the planned duration.

c. Comparison: The actual progress aligned with the initial plan, indicating that the team effectively managed their time and resources during the development phase.

2. Testing and Quality Assurance:
    a. Initial Plan: The test plan included comprehensive testing in various areas, such as functionality, usability, security, performance, and reliability.
    b. Actual Progress: The team conducted thorough testing and quality assurance activities, addressing the defined testing areas and ensuring the application's stability, security, and user-friendliness.
    c. Comparison: The actual progress aligned with the initial plan, indicating that the team successfully executed the planned testing and quality assurance activities.
3. Challenges and Obstacles:
    a. Initial Plan: The initial plan acknowledged potential challenges and obstacles that could arise during the project's execution.
    b. Actual Progress: The team encountered various challenges and obstacles, as mentioned in the previous section, and implemented appropriate solutions to overcome them.
    c. Comparison: The team proactively addressed the challenges and obstacles, indicating effective risk management and problem-solving capabilities.
4. Milestones and Deliverables:
    a. Initial Plan: The initial plan outlined specific milestones and deliverables for each phase of the project, such as prototype development, user testing, and final application release.
    b. Actual Progress: The team achieved the planned milestones and delivered the specified deliverables within the defined timeframes.
    c. Comparison: The actual progress aligned with the initial plan, demonstrating successful milestone achievements and deliverable completion.

5. Iterative Development and Updates:
    a. Initial Plan: The initial plan acknowledged the need for iterative development and updates based on user feedback and testing results.
    b. Actual Progress: The team incorporated user feedback, conducted iterative design iterations, and implemented necessary updates and bug fixes throughout the project lifecycle.
    c. Comparison: The team effectively executed the iterative development and update process, ensuring continuous improvement and meeting user expectations.

5. Impact of Engineering Solutions

Global impact of the project

The StableAI project has the potential to have a significant global impact by introducing innovative deep learning and artificial intelligence technology to the field of design. By enabling users to convert text descriptions into visually appealing images and explore new design ideas effortlessly, the project empowers designers and users worldwide. It opens up new possibilities for creative expression and expands the boundaries of design imagination. The global impact of the project lies in providing a user-friendly and accessible tool that allows people from diverse backgrounds and skill levels to bring their design visions to life.

Economic implications and benefits

The StableAI project has economic implications and benefits for various stakeholders. By offering a unique and powerful design tool, the project can attract a broad user base, including designers, artists, content creators, and individuals seeking visually appealing content. This user demand can lead to commercial opportunities, such as paid subscriptions, licensing agreements, or advertising partnerships, generating revenue streams for the project. Additionally, the project's success can drive economic growth by fostering creativity and innovation in design-related industries, leading to job creation and increased productivity.

Environmental considerations and sustainability

The StableAI project has environmental considerations and contributes to sustainability in several ways. By providing a digital platform for design creation, the project reduces the need for physical materials and resources traditionally associated with design processes. This digital approach minimizes waste and environmental impact. Moreover, by enabling users to explore design ideas virtually before executing them physically, the project promotes efficient resource utilization and reduces the environmental footprint of design iterations. The emphasis on digital workflows and reduced material consumption aligns with sustainability principles.

Societal impact and user benefits

The StableAI project has a significant societal impact and offers various benefits to its users. By democratizing the design process and making it accessible to a wide range of users, the project promotes inclusivity and empowers individuals to express their creativity. It eliminates barriers to entry by providing an intuitive and user-friendly interface, allowing even non-designers to generate visually appealing images. The project's user benefits include saving time and effort by

automating the image creation process, stimulating inspiration and ideation, and providing a platform for users to showcase their creations and gain recognition within the community.

6. Contemporary Issues

Discussion of current issues related to the project topic

The StableAI project, focusing on text-to-image generation and AI-assisted design, operates within a dynamic and evolving landscape. Here are some current issues related to the project:

1. Data Bias and Representation: AI models trained on large datasets may inherit biases present in the data. This can result in biased or unfair outcomes when generating images. It is crucial to address issues of bias in training data and ensure the models are capable of generating diverse and inclusive visuals, avoiding reinforcing stereotypes or marginalizing certain groups.

2. Creative Control and Originality: With AI-generated designs, there is a fine balance between using AI as a tool to enhance creativity and maintaining originality. It is important to consider how AI-generated designs can be a source of inspiration rather than completely replacing human creativity. Encouraging users to leverage AI as a tool to explore new ideas while retaining their unique artistic perspectives is a key challenge.

Analysis of potential challenges and future developments

1. Technical Advancements: As technology progresses, there will be opportunities for improving the stability, quality, and performance of the StableAI application. Advancements in deep learning architectures, data augmentation techniques, and training methodologies can enhance the accuracy and realism of the generated images.
2. User Experience and Interface Design: Ensuring a seamless and intuitive user experience is crucial for the success of the StableAI application. Future developments may focus on refining the user interface, incorporating user feedback, and providing customization options to cater to individual user preferences.
3. Scalability and Resource Management: As the user base grows, scalability becomes a challenge. Managing the computational resources required for training and generating

images at scale will be essential. Optimizing algorithms, leveraging cloud infrastructure, or exploring distributed computing approaches may be potential solutions.

Ethical considerations and social implications

1. Privacy and Data Security: Collecting and processing user data for the StableAI application raises concerns about privacy and data security. It is essential to implement robust data protection measures, obtain user consent for data usage, and ensure compliance with relevant privacy regulations.

2. Accountability and Transparency: As AI generates images, there is a need for transparency and accountability in how the system functions. Users should have an understanding of how the AI model works, its limitations, and any biases it may have. Providing transparency can help build trust and mitigate potential ethical concerns.

3. Intellectual Property and Copyright: AI-generated designs raise questions about intellectual property rights and copyright. Clear guidelines should be established to determine ownership and usage rights for the generated images. Respecting intellectual property and ensuring fair use of AI-generated designs are important considerations.

4. New Tools and Technologies

1. ReactJS: React is a popular JavaScript library for building user interfaces, especially for single-page applications. It's used for handling the view layer in web and mobile apps. React allows you to design simple views for each state in your application, and it will efficiently update and render the right components when your data changes.
2. HTML, CSS, JS: These are the cornerstone technologies for building Web pages:
   2.1. HTML (HyperText Markup Language) provides the structure of a webpage.
   2.2. CSS (Cascading Style Sheets) is used to control the layout and look of the webpage.
   2.3. JavaScript is the scripting language used to create dynamic content on webpages.

3. NextJS: This is a React framework for JavaScript that enables functionality such as server-side rendering and generating static websites for React-based web applications. It's a production-ready framework that allows developers to quickly build static and dynamic JS and TypeScript apps with less setup.
4. Chakra UI: This is a simple, modular, and accessible component library that gives you the building blocks to build React applications. It follows a style system that allows developers to customize components to maintain consistency in design.
5. Axios for React: Axios is a Promise-based HTTP client for JavaScript which can be used in the browser and Node.js. It has many features like making XMLHttpRequests from the browser, automatic transforms for JSON data, and client-side protection against XSRF.
6. NextJS Router: The NextJS Router allows you to route to different pages in your NextJS application. It enables navigation within a Next.js app and supports parameters and query strings.
7. Cookies: 'js-cookies' is a small library to manipulate cookies in the browser. It's often used for storing JWTs on the client-side after successful authentication. However, storing JWTs in cookies can have security implications and needs to be handled carefully to avoid attacks such as CSRF.
8. React-Icons: This library includes a collection of popular icons that can be easily used in React applications. It utilizes ES6 imports that allow you to include only the icons that your project is using.
9. NextJS-Image: This is a feature of Next.js that optimizes image loading for your application. It allows for optimization such as lazy loading images, resizing images, and serving modern formats to modern browsers.
10. FastAPI: This is a modern, fast (high-performance), web framework for building APIs with Python 3.6+ based on standard Python type hints.
11. Response, File (FastAPI): These are classes in FastAPI used to create HTTP responses. File is used for reading file uploads in a request.
12. CORSMiddleware: A FastAPI middleware class that manages Cross-Origin Resource Sharing (CORS), which is necessary when you want to allow cross-domain interactions in your API.
13. HTTPException: This is a FastAPI class that allows you to create HTTP responses with specific status codes and content.
14. TensorFlow (imported as tf): TensorFlow is a free and open-source software library for machine learning and artificial intelligence.
15. Uvicorn: A lightning-fast ASGI server implementation, using uvloop and httptools.
16. PyTorch (imported as torch): PyTorch is an open-source machine learning library based on the Torch library. It provides a flexible deep learning framework and encourages a seamless research-to-production transition.

17. autocast (PyTorch): Provides a context manager for mixed precision. When enabled, each of a model's forward pass is run in mixed precision.
18. StableDiffusionPipeline, StableDiffusionImg2ImgPipeline: These appear to be specific implementations, perhaps custom classes, related to an image to image diffusion model. The names suggest they provide pipeline mechanisms for stable diffusion processes.
19. BytesIO (Python Standard Library): An in-memory stream for binary data. It operates like a file object and is used when you have data in memory that behaves like a file.
20. base64 (Python Standard Library): Provides functions to encode binary data to a printable string and to decode such encodings back to binary data.
21. JSONResponse (FastAPI): A FastAPI class that creates an HTTP response with JSON content.
22. ngrok: A service that exposes local servers behind NATs and firewalls to the public internet over secure tunnels.
23. nest_asyncio: This Python module patches asyncio to allow nested use of asyncio.run and loop.run_until_complete.
24. tensorflow_hub: A library for the publication, discovery, and consumption of reusable parts of machine learning models.
25. BaseModel (FastAPI/pydantic): A Pydantic base class that provides runtime checking and validation of complex data schemas.
26. Image (PIL): Provides a class for storing and manipulating image data.
27. load_learner (fastai): A fastai function to load a Learner object saved with export().
28. os (Python Standard Library): Provides a portable way of using operating system-dependent functionality.
29. numpy: A package for scientific computing in Python.
30. Keras: A high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano.
31. UUID (Python Standard Library): This module provides immutable UUID objects (the UUID class) and the functions uuid1(), uuid3(), uuid4(), uuid5() for generating version 1, 3, 4, and 5 UUIDs.
32. Field (Pydantic): Allows customisation of Pydantic models' definition, used to provide extra information about the field.
33. datetime (Python Standard Library): Supplies classes for manipulating dates and times.
34. Union, Any (Python Standard Library): These are used for Python's static typing. Union is a way to say a value could be one of many types. Any is used when a value could be of any type.
35. Depends, HTTPException, status (FastAPI): These are used for handling dependencies, exceptions, and HTTP status codes in FastAPI.
36. OAuth2PasswordBearer (FastAPI): This is a simple OAuth2 authentication utility for password grant flow.

37. jwt: JSON Web Tokens are an open, industry-standard RFC 7519 method for representing claims securely between two parties.
38. ValidationError (Pydantic): Raised when data fails to match a Pydantic model.
39. db, pymongo: pymongo is a Python driver for MongoDB, a cross-platform document-oriented database system. 'db' is typically used to reference a database instance.
40. OAuth2PasswordRequestForm (FastAPI): A form used to receive username and password for OAuth2 password flow.
41. RedirectResponse (FastAPI): A FastAPI response class that redirects the client to a different URL.
42. UserOut, UserAuth, TokenSchema, SystemUser, Post: These are probably Pydantic models or Python classes used in your application. The exact purpose would depend on the implementation in your code.
43. nest_asyncio: A library that patches asyncio to allow nested event loops.

Benefits and advantages of adopting these tools/technologies

The adoption of these new tools and technologies provides several benefits and advantages for the StableAI project:

1. Enhanced Performance: Deep learning frameworks and GANs allow for the development of advanced models capable of generating high-quality and realistic images. This results in a more satisfying user experience and increases the overall value of the application.
2. Efficient Text Processing: NLP libraries facilitate the extraction of relevant information from text descriptions, improving the accuracy and coherence of the generated images. These tools help in understanding the context and semantics of the input text, leading to more accurate image generation.
3. Scalability and Cost-Effectiveness: Cloud computing services provide the necessary scalability and computational resources to handle large-scale training and image generation tasks. This ensures the application can handle increased user demand without compromising performance or incurring excessive costs.

Challenges encountered and lessons learned

Throughout the development of the StableAI project, several challenges were encountered, leading to valuable lessons learned. One significant challenge was the technical complexity involved in working with deep learning frameworks and implementing Generative Adversarial Networks (GANs). Overcoming this challenge required assembling a skilled and knowledgeable team with expertise in machine learning and neural networks. Additionally, efficiently managing computational resources, particularly when dealing with large datasets and complex models, proved to be a challenge. This necessitated optimizing resource utilization, employing distributed

computing approaches, and closely monitoring performance. Another challenge was fine-tuning the text-to-image generation algorithms to achieve better performance and image quality. This required iterative experimentation and algorithmic refinements based on user feedback and evolving techniques. Lastly, the adoption of AI technologies raised ethical considerations, including bias, privacy, and copyright. Addressing these concerns required proactive measures to mitigate biases, ensure data privacy and security, and adhere to legal and ethical guidelines. These challenges provided valuable lessons, emphasizing the importance of building a skilled team, efficiently managing computational resources, continuously refining algorithms, and considering ethical implications throughout the project's lifecycle.

5. Use of Library and Internet Resources

Overview of library resources utilized

During the development of the StableAI project, various library resources were utilized to support research, gain insights, and gather relevant information. These resources included academic libraries, online databases, and digital repositories. The team accessed scholarly journals, conference proceedings, research papers, and books related to deep learning, image generation, natural language processing, and AI technologies. These library resources provided foundational knowledge, theoretical frameworks, and technical guidance to inform the project's development.

Description of relevant Internet resources accessed

In addition to library resources, the project team accessed relevant Internet resources to gather information, stay updated on advancements, and explore practical applications. Online platforms such as arXiv, researchgate.net, and Google Scholar were valuable sources for accessing preprints, published articles, and conference papers in the field of AI and computer vision. Online forums, blogs, and technical documentation related to deep learning frameworks, NLP libraries, and GANs were also explored. These Internet resources offered a wealth of information, insights, and practical examples that contributed to the project's understanding and implementation.

Role of background information in project development

Background information played a critical role in the development of the StableAI project. It provided the necessary foundation and understanding of key concepts, algorithms, and techniques in the areas of deep learning, image generation, and natural language processing. The team extensively studied existing research, academic literature, and technical documentation to gain insights into text-to-image generation, style transfer, and AI-assisted design. This background information guided the project's methodology, model architecture, and implementation decisions. It also helped in evaluating the feasibility, effectiveness, and limitations of different approaches and algorithms. By leveraging background information, the

project team was able to make informed decisions, adapt best practices, and push the boundaries of text-to-image generation.

6. Test Results

Summary of the test plan

The test plan for the StableAI project aimed to ensure the quality and functionality of the application across various aspects, including functionality, usability, security, performance, and reliability. It covered different types of tests such as functional testing, usability testing, security testing, performance testing, and reliability testing. The goal was to validate the core features, assess user-friendliness, ensure data protection, evaluate responsiveness and scalability, and validate stability and robustness.

Description of each test case

Each test case in the test plan was designed to target specific aspects of the application. The functional testing focused on validating the text-to-image conversion, image-to-image transformation, and style transfer functionalities. Usability testing assessed the intuitiveness of the user interface and the overall user experience. Security testing aimed to identify and address any vulnerabilities in data handling and authentication processes. Performance testing evaluated the application's responsiveness and scalability under different load conditions. Reliability testing aimed to validate the stability and robustness of the application by stress testing and identifying potential failure points.

Pass/fail status for each test

The pass/fail status for each test case was recorded based on the results obtained during the testing phase. The test cases that met the expected criteria and produced the desired outcomes were marked as "Pass." On the other hand, if a test case failed to meet the expected criteria or resulted in unexpected behavior, it was marked as "Fail."

Assessment of test results

The overall test results were assessed to determine the application's performance and adherence to the defined standards. The assessment involved analyzing the number of passed and failed test cases, identifying any patterns or trends in the failures, and evaluating the impact of the failures on the application's functionality and user experience. The test results provided insights into the strengths and weaknesses of the application, highlighting areas that required further improvement or bug fixing.

Statistics on failed tests

Statistics on failed tests were compiled to gain a better understanding of the areas that needed attention. These statistics included the number of failed tests, the severity of the failures, and the frequency of specific failure types. This analysis helped prioritize bug fixing and enhancement efforts, ensuring that critical issues were addressed first.

Discussion of identified bugs and potential enhancements

During the testing phase, various bugs and issues were identified and documented. These ranged from minor usability issues to critical functional failures. Each identified bug was thoroughly analyzed to understand its root cause and impact on the application. Discussions were held to determine the priority of bug fixing and potential enhancements. The discussions also focused on identifying areas for improvement, such as enhancing the accuracy of text-to-image conversions, optimizing performance, refining user interface elements, and incorporating user feedback to enhance the overall user experience.

User Test

1. Users must be able to register on our website.
2. Users should be able to log in with the e-mail and password they registered on our website.
3. Users should not use any feature of the site without logging into our website.
4. Login users should be able to generate images from text prompt.
5. Login users should be able to upload an image, add a prompt and produce a new image.
6. Login users should be able to transfer the style from one image to another with the style transfer feature.
7. Login users should be able to save the generated images and see them later in their profiles.
8. Login users should be able to see the images generated by the users with the explore feature and be able to surf there.

We have achieved all these milestones, and we have encountered many problems and produced many new solutions in this process. We have generated access_token for each user using JWT so that users can register on the website. In this part, we have eliminated many problems by using the cookies feature of the browser in order to store the access_tokens of the users on the client side.

In order for users registered to our website to log in, we ensured that the access_token generated by JWT is returned to the user if the user's e-mail and password match. We overcame this with cookies so that we can control users' access to the site.

Challenges

We have done numerous tests to ensure that users can generate images from text prompts. We have worked diligently to prevent both +18 and racist content from showing, and we have succeeded, we have conducted many tests. In addition, we have prevented this by checking the access_tokens of users in all generate operations so that users who are not logged in cannot generate images.

In the feature of generating images from Image, we also enabled logged in users to be able to generate them with the help of access_token. In addition, we had problems such as too much file size while sending base64 code to the api. But we solved all this and tested it many times to make sure there was no problem.

We thought that it would be difficult for users to upload two images for the style-transfer feature, and we uploaded 3 table images as a solution to this, and we thought that users would prefer them more. We also tested this feature and ensured that the base64 codes of these images were generated together with the other image uploaded by the user. In addition, as with other features, we have provided access_token control so that only logged in users can use our features.

The most challenging problem we experienced was that we were getting CORS errors in the responses returned from the API to the client and the requests sent from the client to the API, and we learned that the reason for this was the Ngrok that our API was connected to, and we found the solution by doing many tests. We solved our problem by adding a header on the headers that we can solve the warnings of ngrok, by doing all our post and get operations and we tested it many times.

7. Conclusion

Recap of project achievements and contributions

Throughout the development of the StableAI project, significant achievements and contributions have been made. The project successfully implemented three core functionalities: text-to-image conversion, image-to-image transformation, and style transfer. These functionalities empower users to unleash their creativity and bring their ideas to life through AI-assisted design. The application provides a user-friendly interface, allowing users to easily generate and share images based on their textual input. The project has also demonstrated the successful integration of deep learning, artificial intelligence, and image processing techniques, showcasing the potential of these technologies in the design domain.

Lessons learned and future recommendations

We are planning to move the StableAI project to an area where even our competitors cannot reach, in order to surpass them. As one of the few projects that combine Artificial Intelligence with Web 3.0, we intend to take our StableAI project to the Web 3.0 environment. We envision a project where users can log in with their own Metamask Wallets, and all authentication processes are done with wallet IDs. This will ensure that all the images generated by the users are unique to them. We will make sure that all generated images become NFTs, enabling users to sell and buy them on platforms like Opensea. This will provide both us and the users with an opportunity to profit. As a result, we will be able to reach a much larger user base. All user data and NFT prompts will be stored on IPFS data storage, making the entire artificial intelligence project decentralized through the use of blockchain. This will set us apart from our competitors in the current web world and enable us to achieve greater success and advancement.

Closing remarks

In conclusion, the StableAI project has successfully developed an application that utilizes deep learning and AI to convert text descriptions into visually appealing images. It has provided designers and users with a powerful tool to explore new design ideas and perspectives effortlessly. The project's achievements and contributions have opened up exciting possibilities in the field of AI-assisted design and have showcased the potential of stable diffusion in generating detailed visuals conditioned on text prompts. By delivering a high-quality, user-friendly application, the project has made a positive impact on the design community, allowing users to bring their dreams and visions to life with just a few words or sentences. As the project concludes, it is essential to continue supporting and maintaining the application, incorporating user feedback, and staying at the forefront of advancements in AI and design to ensure its continued success and relevance in the ever-evolving technological landscape.

8. References

- https://standards.ieee.org
- https://www.umb.edu/it/project_management_office/methodology/project_roles_responsibilities
- https://www.acm.org/code-of-ethics
- https://www.lucidchart.com/blog/types-of-UML-diagrams
- https://www.howtogeek.com/830179/how-to-run-stable-diffusion-on-your-pc-to-generate-ai-images/#autotoc_anchor_1
- https://dataconomy.com/2022/09/stable-diffusion-ai-art-generator/
- @InProceedings{Rombach_2022_CVPR,
  author = {Rombach, Robin and Blattmann, Andreas and Lorenz, Dominik and Esser, Patrick and Ommer, Bj"orn},
  title = {High-Resolution Image Synthesis With Latent Diffusion Models},
  booktitle = {Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)},
  month = {June},
  year = {2022},
  pages = {10684-10695}
  }
- Golnaz Ghiasi, Honglak Lee, Manjunath Kudlur, Vincent Dumoulin, Jonathon Shlens. Exploring the structure of a real-time, arbitrary neural artistic stylization network. Proceedings of the British Machine Vision Conference (BMVC), 2017.
- https://tugrulbayrak.medium.com/jwt-json-web-tokens-nedir-nasil-calisir-5ca6ebc1584a