Part A (revised)

Motivation and Novelty:

Our project is a web application that perform deep-learning-based brain extraction for medical imaging researchers. In most research on brain images, skull needs to be removed from brain tissue to prevent interference and confusion in later analysis. Our application could allow medical researchers without coding or deep learning background to train a deep-learning model from a small number of training examples. Then, they could run brain extraction with the model on the rest of their brain images. This automated tool could reduce their work of manual brain segmentation while offering them a relatively good accuracy.

Currently, there are many automated brain extraction tools available to researchers. However, most automated brain extraction tools are using relative old, non-deep learning models and have relatively bad accuracy. As a result, researchers may have to refine all brain extraction results manually after using such tools. Deep learning models, on the other hand, are recently developed on this task and are proved to be very useful.

However, most researchers could not make full use of deep learning models because of two obstacles. Firstly, performance of deep learning models highly relies on the training examples. Thus, different types of medical images, MRI and CT for example, must be trained separately. Since it's almost impossible to prepare a trained model for each type of medical images, we have to allow the researchers to train new models on their image. This stops commercial software from using deep learning since it requires too much computation power and slow down the brain segmentation dramatically. Secondly, most medical researchers have limited programming and machine learning background. It would be unrealistic for them to perform preprocessing or modify network structure to adapt open source code for their images in a medical image format. Also, due to their limited access to computation power, the process of training new model would be very inefficient.

Our proposed project attacks these two problems by providing the preprocessing and other programming parts for the researchers with the needed computation power. The users could upload segmented(training) and raw medical images. They can also training pre-defined deep learning models on their training data or apply a model on raw medical images to perform brain extraction. A user without programming background could accomplish these tasks and take advantage of deep learning in the brain extraction task.

In addition, the proposed application would also allow sharing trained models. In this way, new users could search for existing pre-trained models and directly apply them on their images. We believe allowing community contribution could further reduce the average time needed for this task.

In conclusion, the proposed application would allow researchers with limited or no programming background to make use of deep learning on their brain extraction task. Shared model features allows community to contribute and could further boost the process. To the best of our knowledge, this proposed application is the first application for deep-learning based brain extraction.

BRAIN.AI
TEAM 5B
Section 1B

Hao Zhuang
Jiayu Hu
Sichen Song
Taixing Shi
Yash Choudhary, 704630134

Github: https://github.com/CS130-W20/team-B5

Requirements Analysis

The purpose of this section is to present a breakdown of the requirements for the Brain.ai application using various UML diagrams. The diagrams cover use cases, specific system features and hardware/software interfaces.

Use Case Diagram

The use case diagram below describes the actions our application will perform when interacting with a user (*actor*). A user can upload data, view data, download results/data, train models, view existing models, apply models to data, share models and delete models (*use cases*). All actions taken by the user are accompanied with a user authentication protocol to ensure security; also, the Train model & Apply model to data actions are accompanied with a View status action so users can see the status of the training/model.

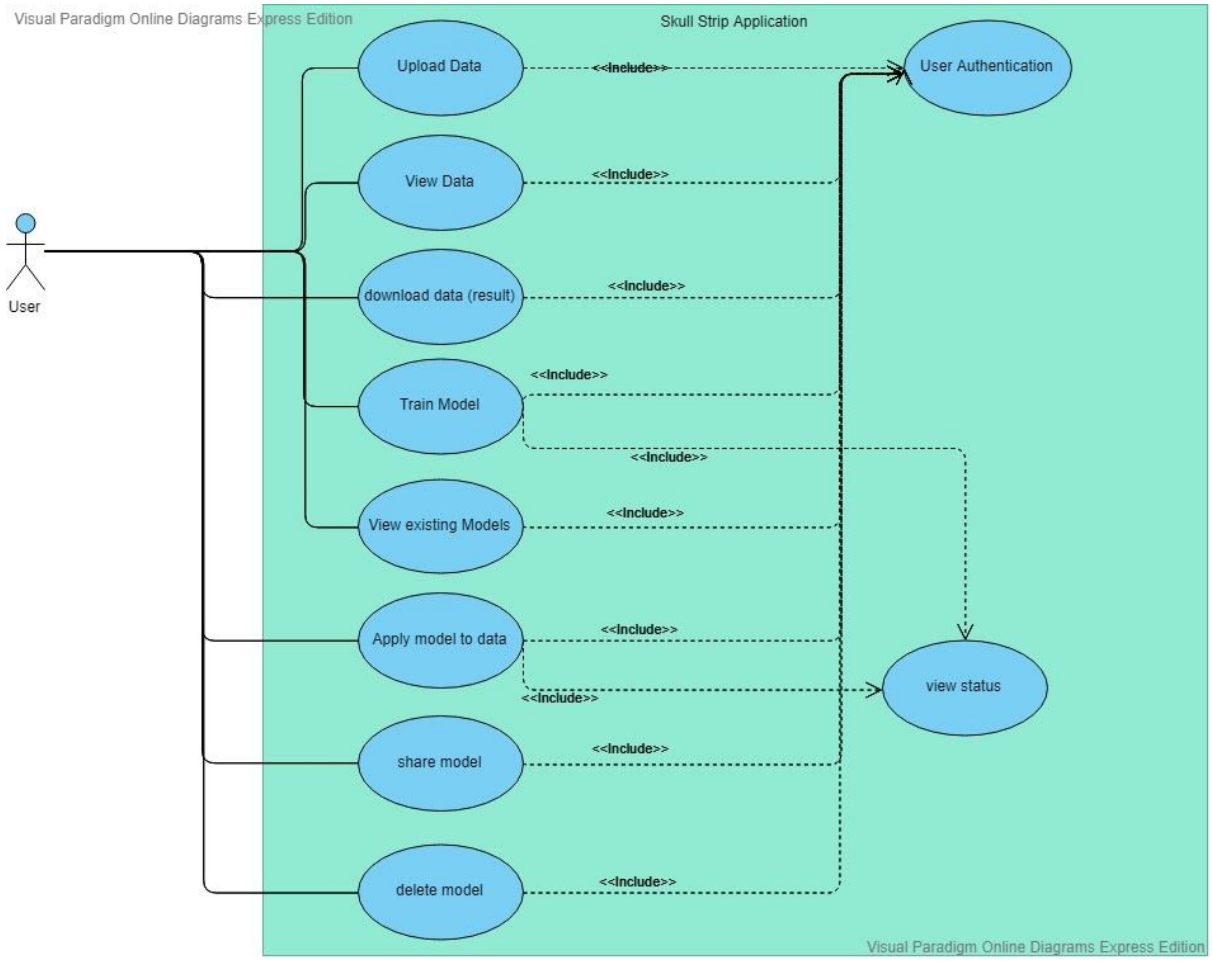


Figure 01: Use case diagram for user interacting with the Brain.ai application

Activity Diagrams

The activity diagram below gives a high-level overview of the actions a user can take and the operations the application can perform. The user can perform the same operations as described in the use case diagram above. The user interacts with the skull strip application which has functions to Manage Data which is used to handle all data related operations (View, Upload, Download), Model Training which is used to handle Train Model actions, Model Storage which is used to store models & handles View Model actions and Model Prediction which is called by the Run Skill Strip action.

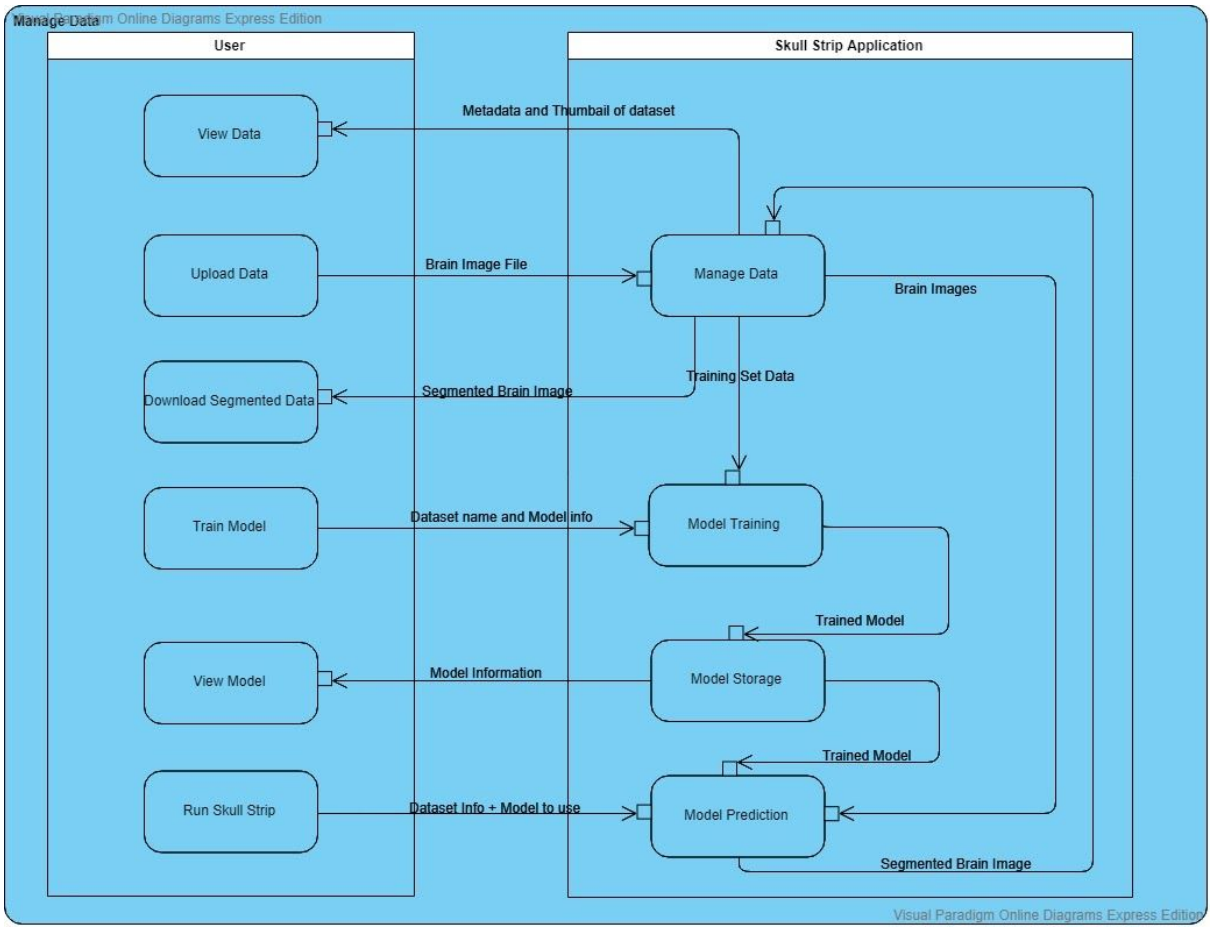


Figure 02: Activity diagram showing flow of control between user & application actions.

The activity diagram below shows the flow of action when the application is authenticating a user. It checks the credentials and returns an access token if validation is successful, otherwise it returns an error.

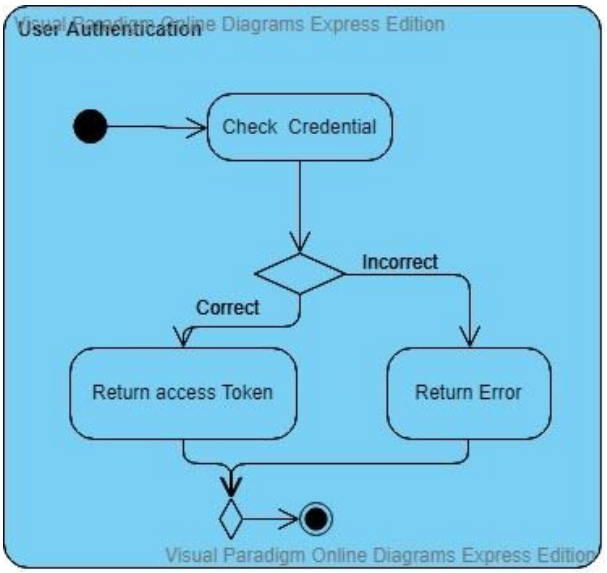


Figure 03: Activity diagram for User Authentication

Sequence Diagrams

The sequence diagram below illustrates the flow of control when the use case is a user training a model. First, the application ensures the right user is active by performing user authentication. Next, the user uploads training data which is sent to the application to be processed & in return the user gets a Data id. Once the user starts training with the Data id, the application assigns the training a Task id which is returned to the user. The user can then view the status of the training by sending the application their Task ID. Once the training is finished, the application sends the results back to the user.

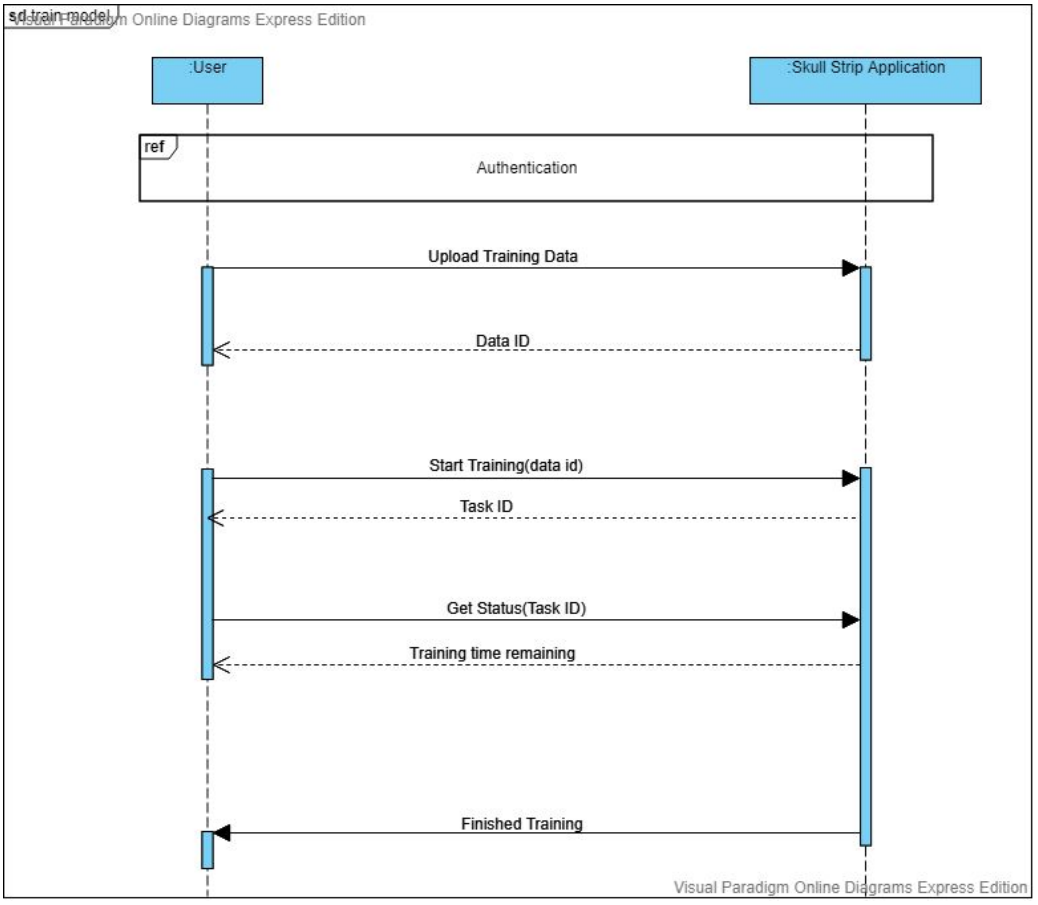


Figure 04: Sequence diagram of a user training data using the application.

The sequence diagram below illustrates the user authentication process that is performed as the first step of the application. The user credentials are first sent to the application, then depending on if the credentials are authentic (match with the original ones) either an access token or an error message is sent.

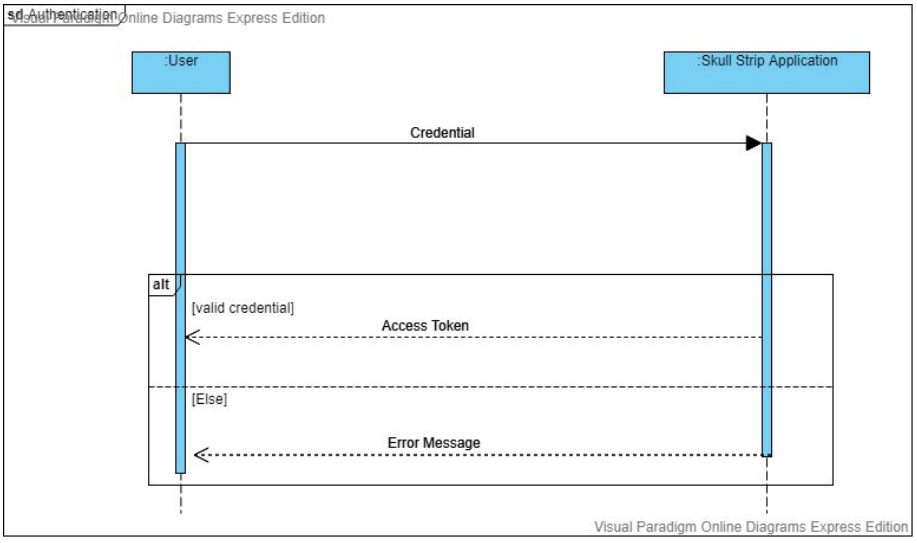


Figure 05: Sequence diagram of the user authentication process

The sequence diagram below illustrates the use case when a user wants to extract brain tissue from an image using the application. After authentication the user uploads the image, browses available models to train with, chooses a model & triggers the application to start segmentation. The user can view the status of the task/segmentation and once done the user can download the result.

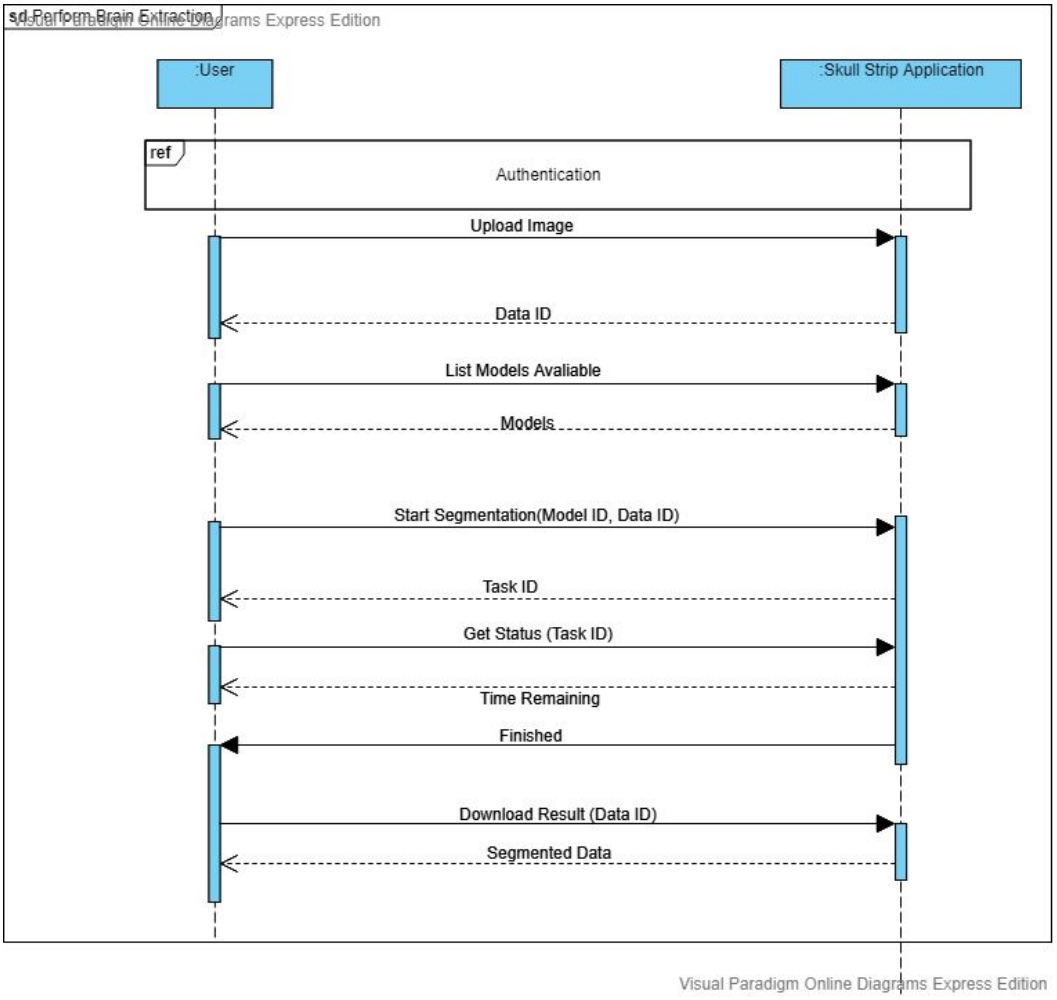


Figure 06: Sequence diagram of user segmenting data using models to extract brain tissue image.

State Machine Diagram

The state diagram below shows the high-level overview of the system in terms of flow of tasks. The states are Pending, Running and Finished.

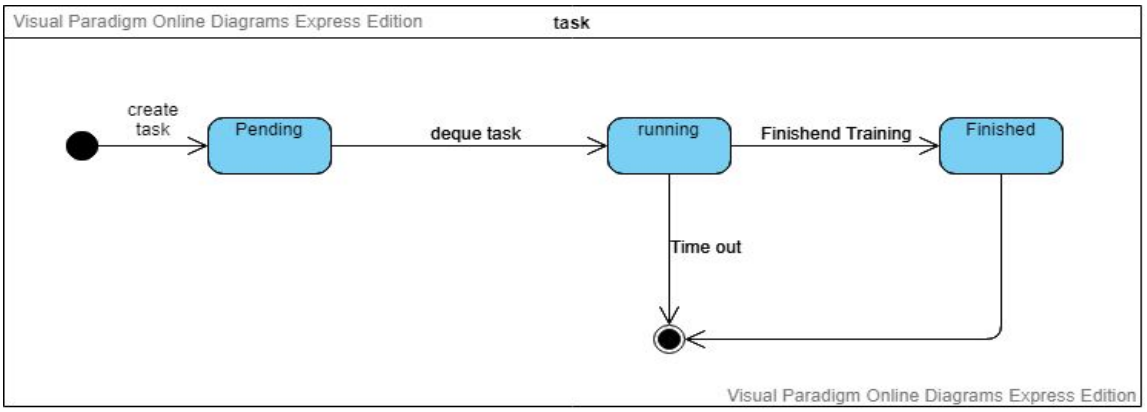


Figure 07: State Machine Diagram for the application

Class Diagram

The detailed class diagram describes the workflow between objects, interfaces and system components within the application. It takes into account the information from the “Architecture Description” section and is the final version of our class diagram.

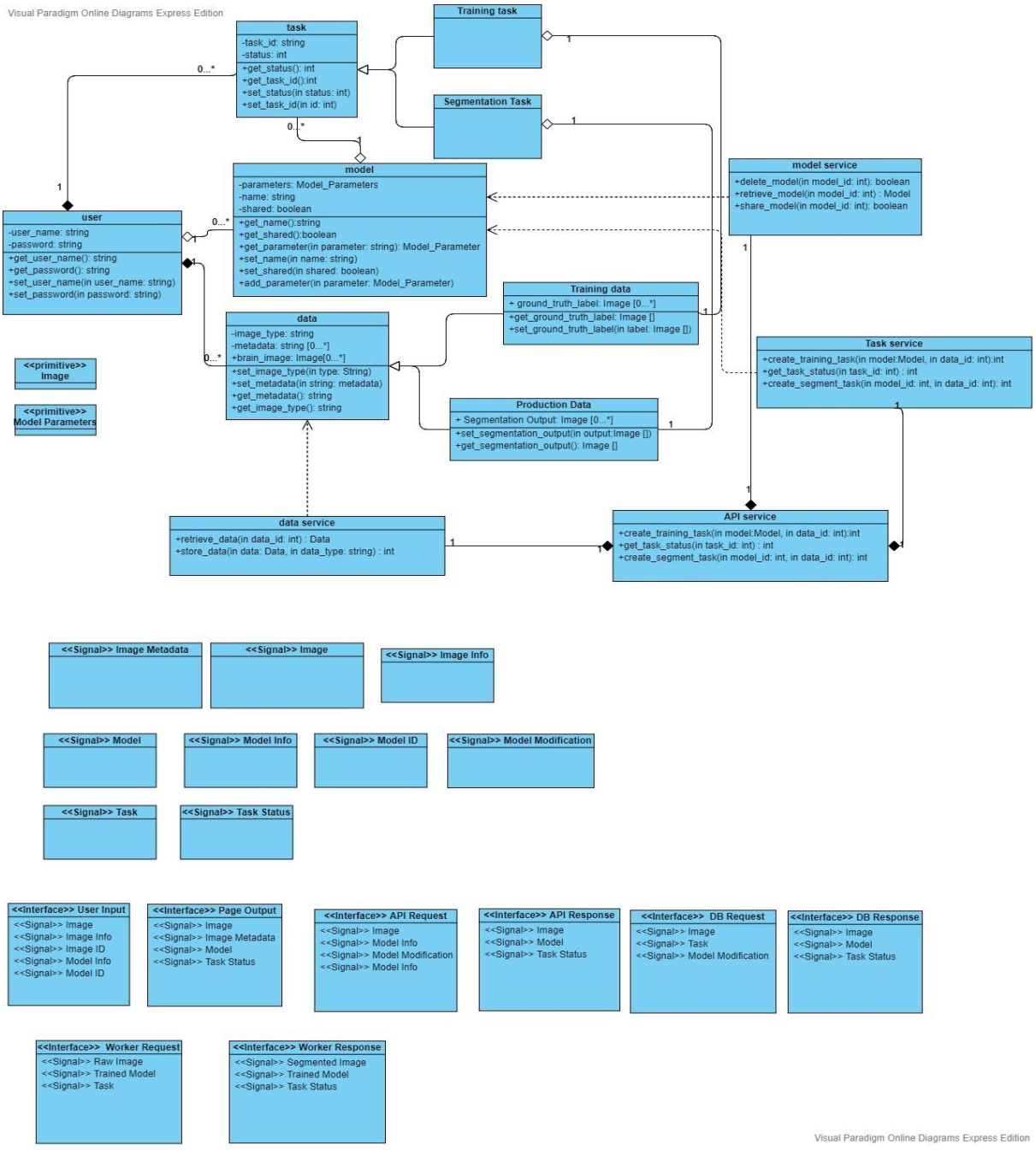


Figure 08: Class diagram of application

Architecture Description

This section presents out detailed analysis of the diagrams in the previous section in the form of diagrams.

Composite Structure Diagram

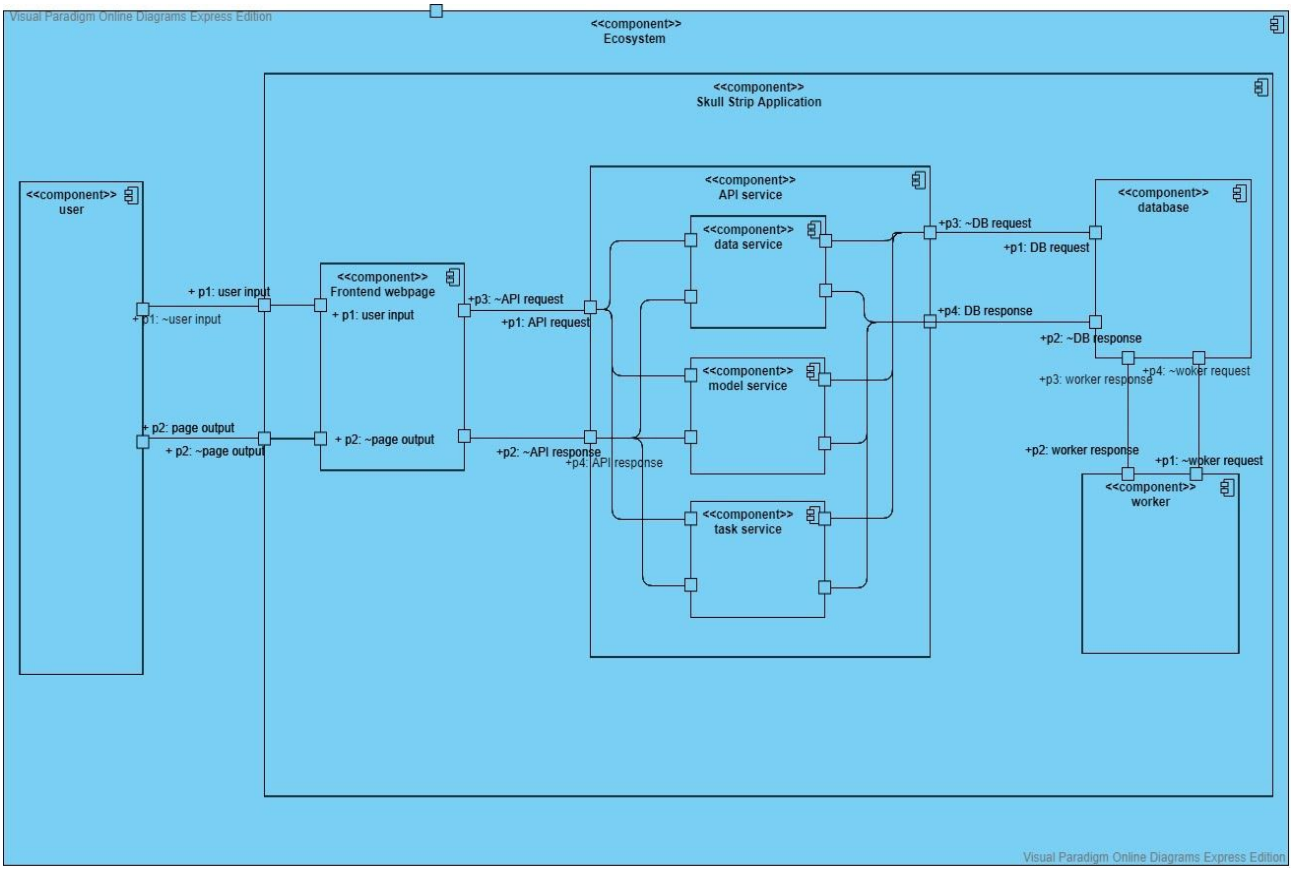


Figure 09: Composite Structure Diagram for the uses cases of the application

Activity Diagrams (with smaller functions detailed within system functions)

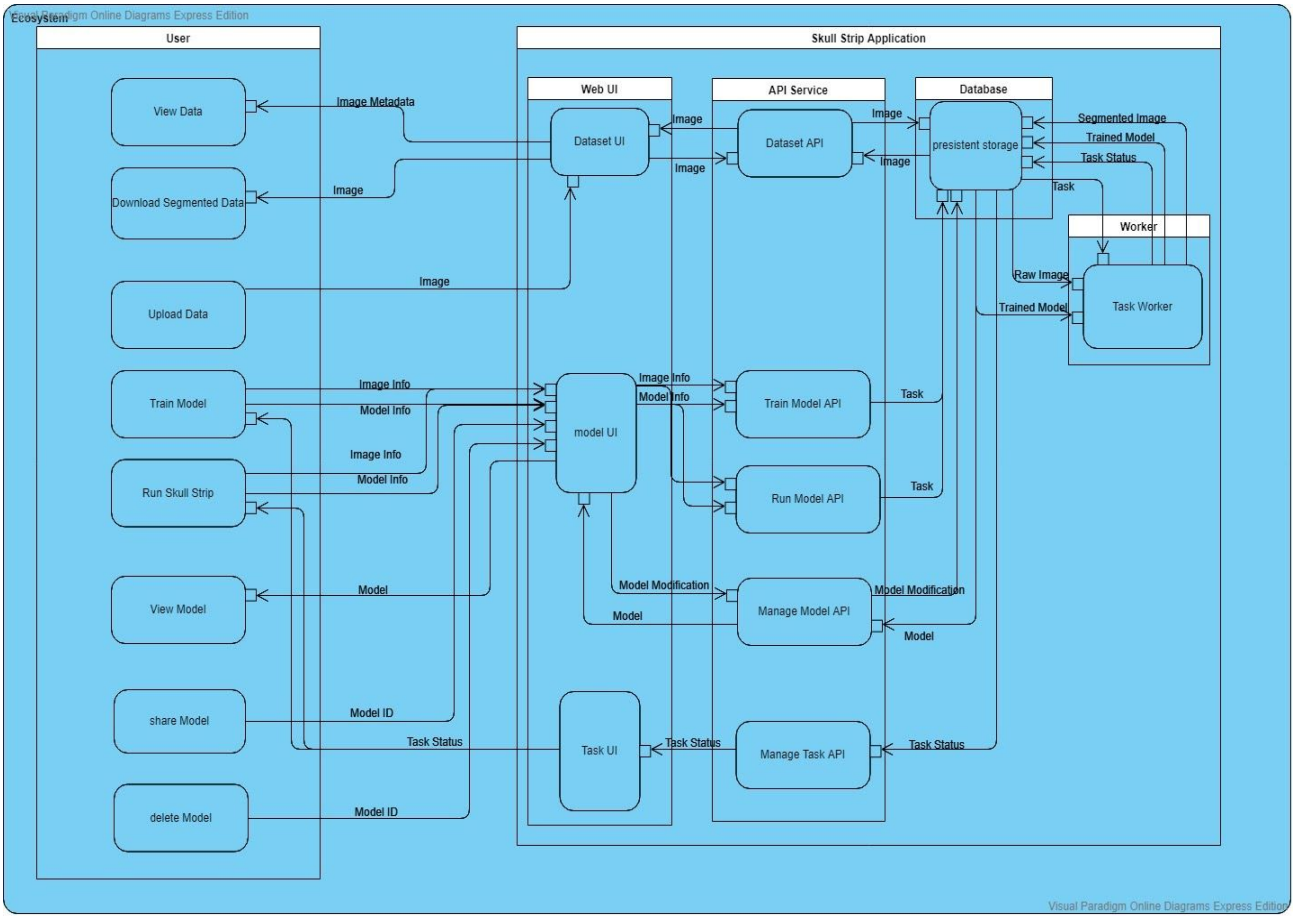


Figure 10: Revised activity diagram after decomposing system functions

UI Mockups

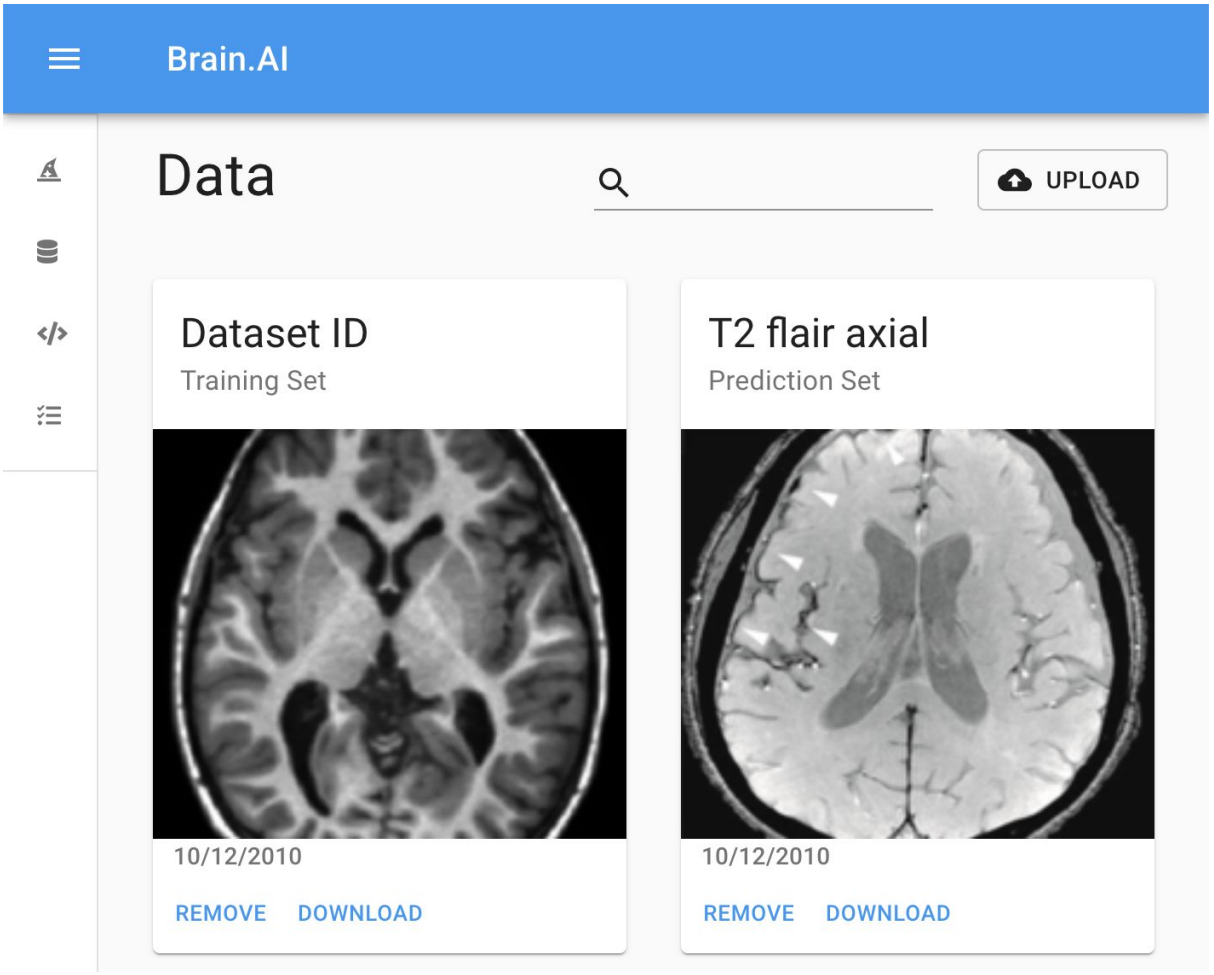
Our application allows researchers to create neural network models in order to extract brain from a medical imaging. To begin with the process, our users need to upload the training data set to our databases and then create a neural network model assigned with the training data. Then, the back-end server will use the data provided to train the corresponding model. After the training process is completed, users can upload a prediction data set to the database, and then designate a trained model to make the prediction (performing the brain extraction). When the prediction task is finished, our user can use the website to download the result.

Data page:

The data view of the web page allows users to upload the dataset for training or prediction. When clicking the “Upload” button in the top right corner, there will be a popup that guide users to select file to be uploaded and choose the type, which is either training data or prediction data, of the dataset.

After that, the data view of the website will display all the datasets with their IDs, types, and visualizations. At the bottom of each of the data, users can also choose to remove or download the corresponding data.

For users with many datasets, the user could search by the name of the dataset with the interactive search text field.

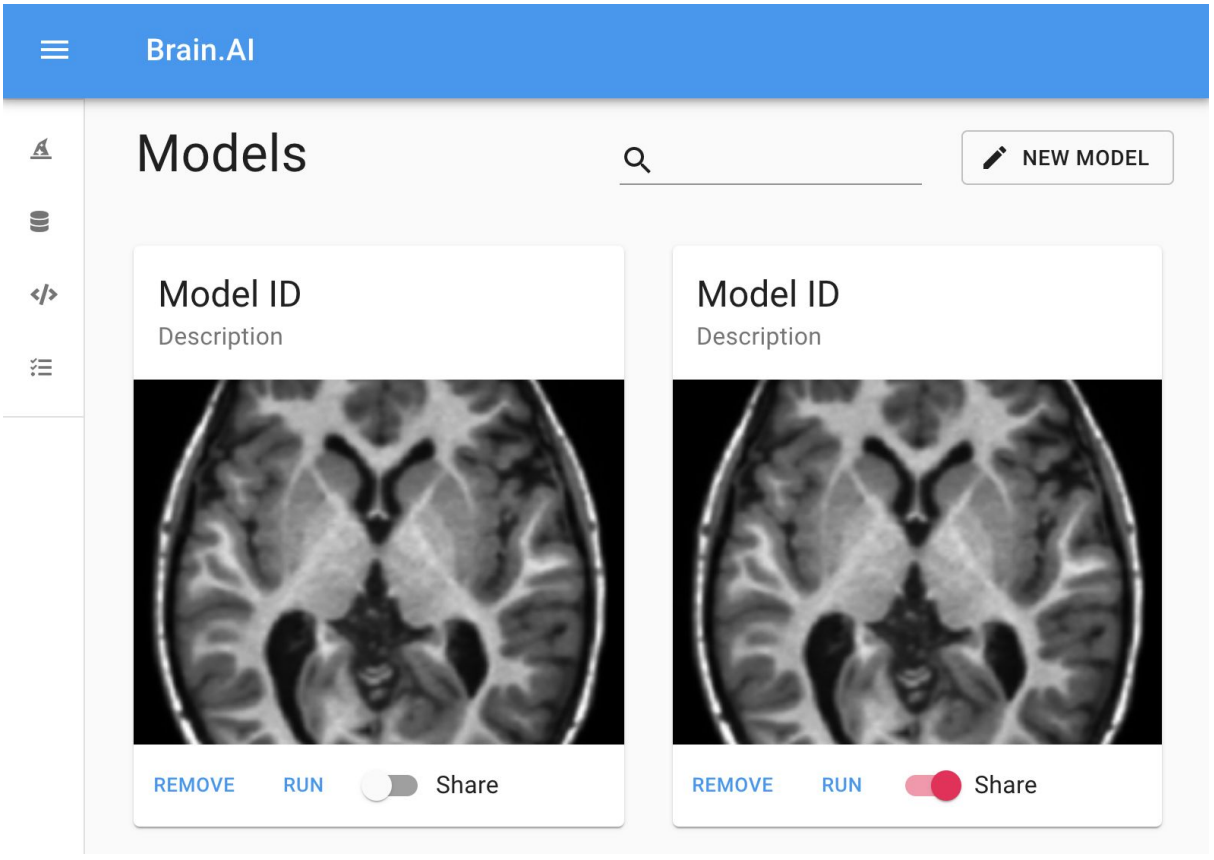


Model page

The model view is for helping users to create or manage models that will be used for prediction. At the beginning, users need to click “New Model” in the top right corner in order to create a model for training. After clicking, there will be a popup that asks users to provide a Model ID, a description, and an associated training dataset in order to train the model.

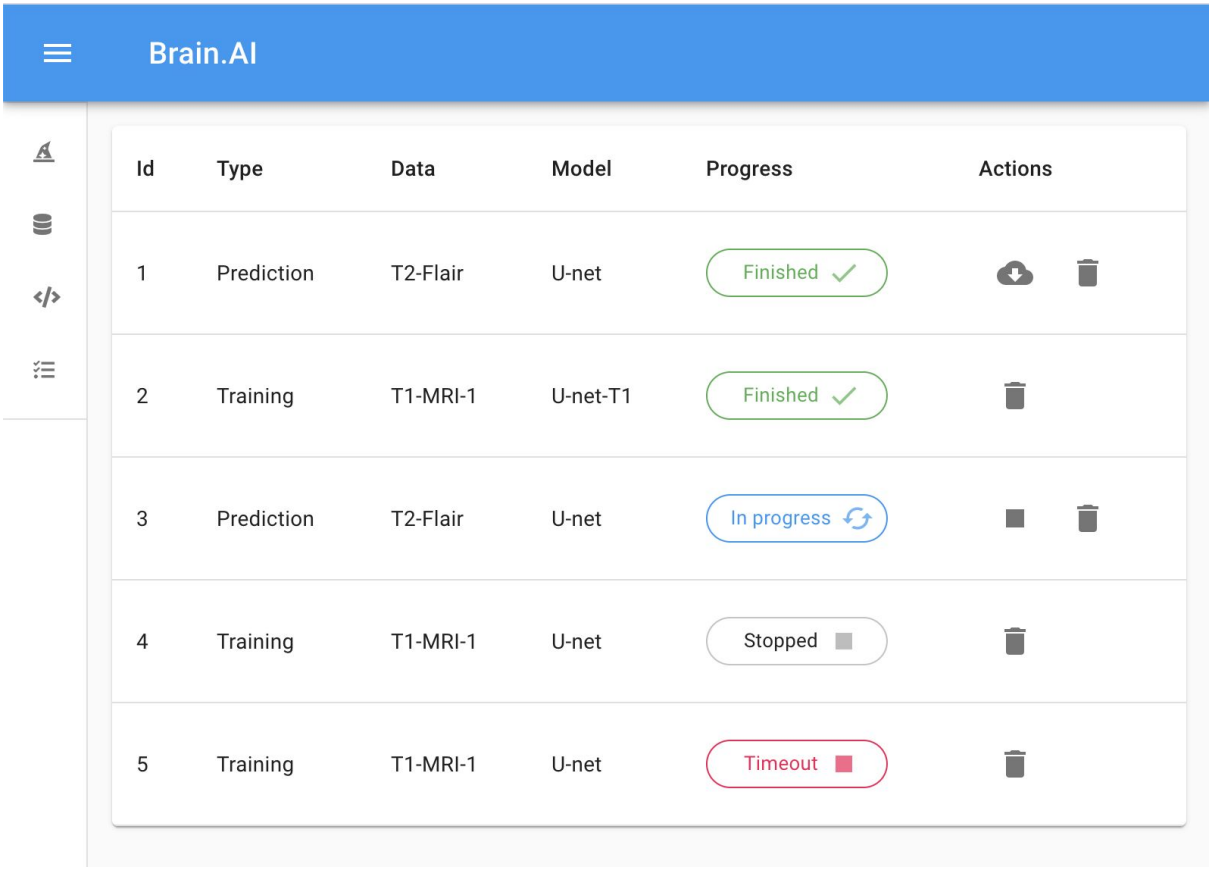
Then, the model will be automatically trained. When finished, the model view will display all the models that have been trained. For each trained model, the page will display its ID and description. At the bottom of each model, there are three controls, which are “Remove”, “Run”, and “Share”.

Remove: Remove the corresponding model.
Run: Use the corresponding model to predict a dataset. When clicked, a popup will guide users to select a prediction dataset in order to make the prediction.
Share: Allow users to share the model in the global model collection, which will allow other users to reuse the model without having to train a new one.



Task page

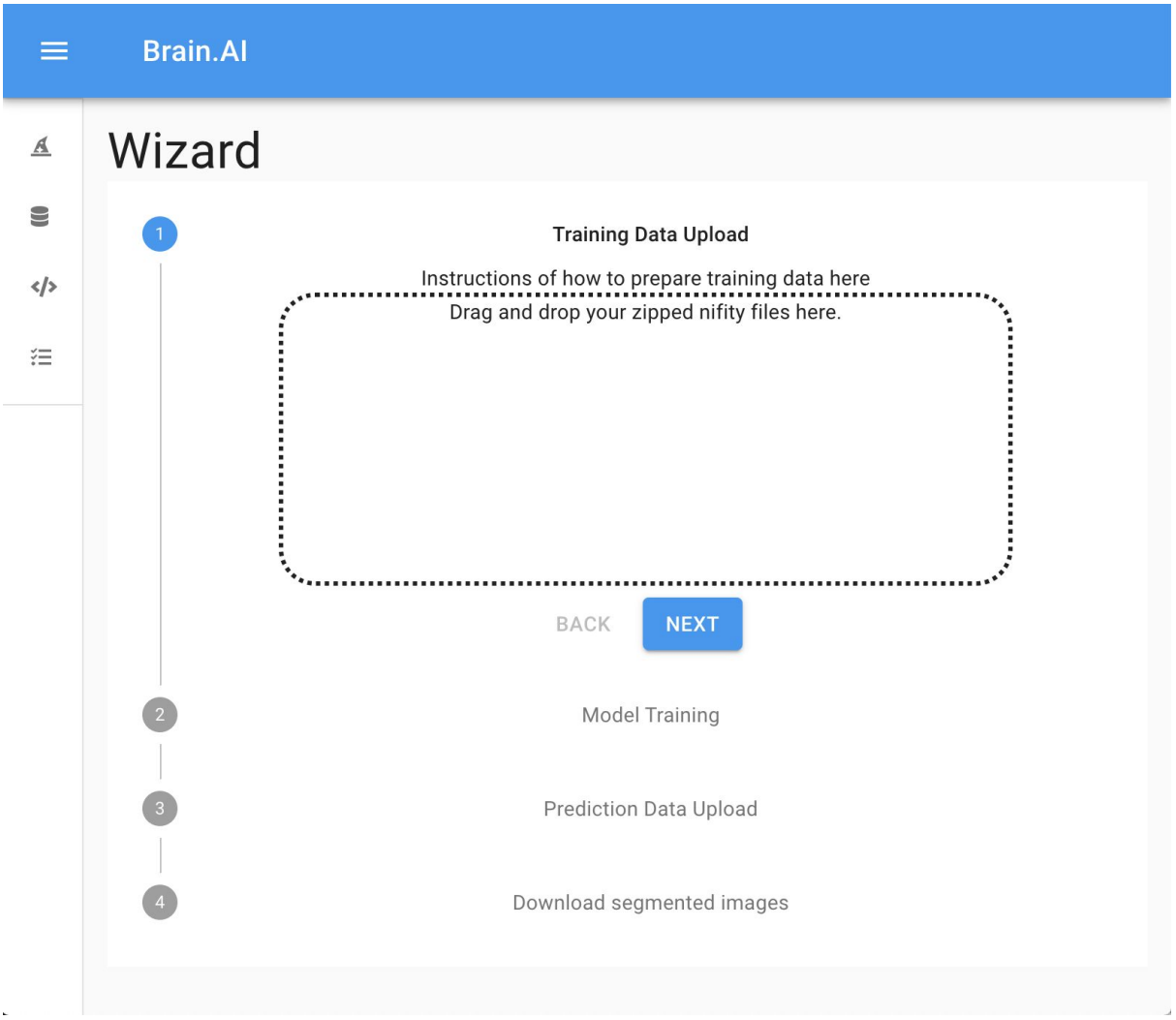
In the task view, user can see all the tasks they have created.
Training Task: The task for training, which is created when a new model has been created with an assigned training dataset.
Prediction Task: This is the task for predicting, which is created when users click “run” in the model view and assigned a prediction data set for predicting. When the task is finished, users will be able to download the predicted result using the download button shown in the right of each table entry.



Other pages

The Wizard

This is a guide to help users to complete the entire process in a single run.



Sign In

This allows users to sign in using their account.