**Overview of App**

1. Capture Echocardiography Images: Use a device capable of capturing fetal echocardiography images. These images will serve as the input data for your classification system.

2. Labeling Images: Each image needs to be annotated with the appropriate labels. These labels will indicate the regions of interest, such as the fetal heart, and any defects present. This labeling process will assist in training your segmentation model.

3. Convert Annotations to JSON Format: Once the images are labeled, convert the annotations into JSON format. This format will include information about the coordinates of the labeled regions, which will be crucial for the segmentation process.

4. Semantic Segmentation: Utilize a semantic segmentation model to segment the fetal heart from the background and identify any abnormalities or defects. This process will create segmented images where different regions are labeled according to their content.

5. Upload Segmented Images: Set up a Flask application to create a web interface where users can upload the segmented images.

6. Integration with Classification Model: Develop a classification model that takes the segmented images as input and predicts whether the fetal heart is normal or has any defects. (FCN-ResNet)

7. Display Classification Results: Once the classification model processes the segmented image, display the results on the screen. This could include indicating whether the fetal heart is normal or specifying the type of defect present if any.

8. Flask and Ngrok Integration: Use Flask to create a web server for your application, and Ngrok for tunneling to make your Flask app accessible over the internet.

Here's a brief overview of how the process would flow: Users capture echocardiography images, label them, and convert the labels into JSON format. These annotated images are then used to train a segmentation model, which segments the fetal heart and any defects. Users upload these segmented images to the classification page, where the classification model predicts whether the fetal heart is normal or has any defects. The results are displayed on the screen for interpretation.

To implement this Flask web application for fetal heart classification using segmentation and classification, follow these steps:

| **Setup For Virtual Environment**  Create :**py -3.8 -m venv venv**  To Activate:**.\venv\Scripts\activate.ps1**  To deactivate:**deactivate** |
| --- |

**1. Install Required Packages:**

| Make sure you have Python installed on your system. Install the necessary packages using pip:  **pip install flask opencv-python numpy pyngrok tensorflow** |
| --- |

**2. Set Up Your Project Structure:**

| Create a directory for your project and within that directory, create the following structure:    project\_folder/  ├── app.py  ├── app1.py  ├── templates/  │ ├── index.html  │ └── upload.html  └── fetal\_heart\_classification\_model.h5 |
| --- |

**3. Write HTML Templates:**

| Add two HTML templates, `index.html` and `upload.html`, within the `templates` directory. These templates will define the structure of your web pages |
| --- |

**4. Write Python Code:**

| Copy the provided Python code and save it as `app.py` in your project folder. |
| --- |

**5. Run the Application:**

| Open a terminal, navigate to your project directory, and run the Flask application:  **python app.py** |
| --- |

**6. Access Your Web Application:**

| Once the Flask application is running, you can access it in your web browser using the Ngrok tunnel URL provided in the terminal. |
| --- |

**7. Upload and Test:**

| Upload an image of a fetal heart using the upload form in your web application. The application will segment the heart from the image, classify it, and display the predicted class label. |
| --- |

Ensure that the `**fetal\_heart\_classification\_model.h5**` file exists and contains a trained model for fetal heart classification.