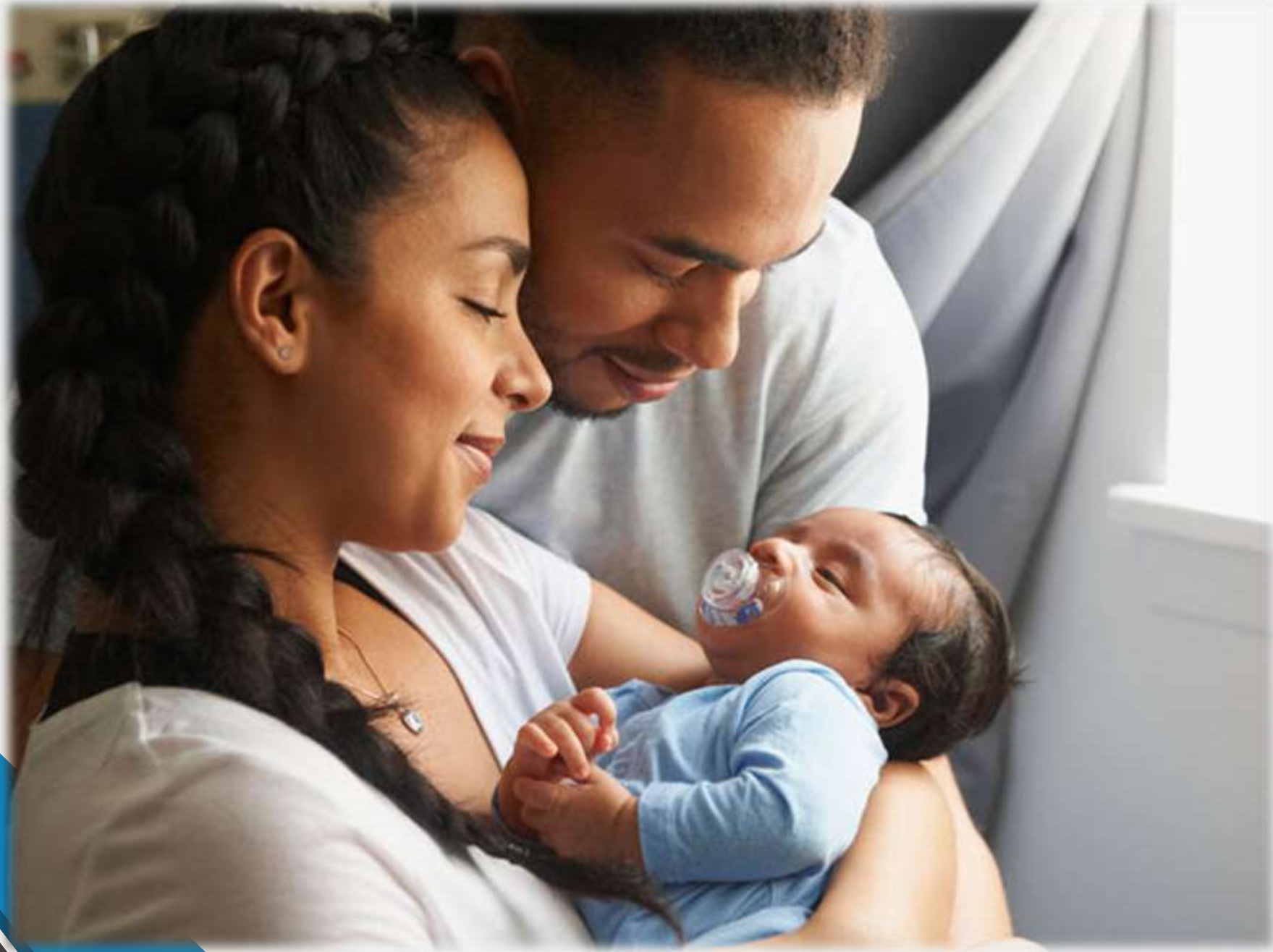


**A MECHANISM TO
KEEP THE
IMMUNIZATION
HIGH OF
NEWBORNS
WITH CRUCIAL
TECHNIQUES VIA
REMOTE
CONSULTATION
(TMP-23-103)**



Progress after proposal presentation

- Changes of the components
- Information gathering
 - ✓ Field visits
 - ✓ Interviews
 - ✓ Research papers, tutorials
- Implementations (basic models)

Changes of the components

Component Name	Before Proposal presentation	After Proposal presentation
Decentralized patient information system	<ul style="list-style-type: none">• Permission-based access	<ul style="list-style-type: none">• Permission-based access control to ensure that only authorized parties can access the data.• Improving Novelty
Chatbot Application	<ul style="list-style-type: none">• Medical knowledge base prediction for diseases• Suggestions of remedies• System notifications for the vaccinations• Chat history	<ul style="list-style-type: none">• Identify skin infections using image processing• As the novel feature, uploading an image option was introduced by the panel

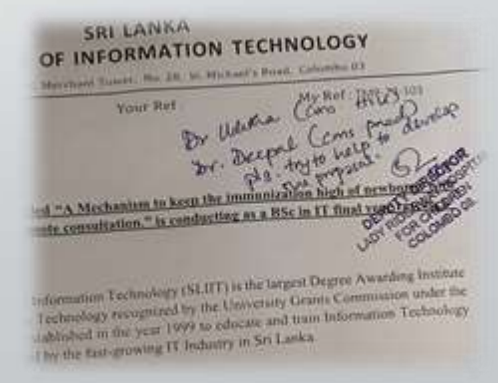
Changes of the components

Component Name	Before Proposal presentation	After Proposal presentation
Nutrition Level and malnutrition detection	<ul style="list-style-type: none">• Prediction of nutritional level using image processing and calculations.• Predict the nutritional level using height, weight, and head circumference	<ul style="list-style-type: none">• Predict the growth level without using images but behavioral aspects• Component name changed to "Growth Level Prediction"
Infant Sickness Identification through Video Processing	<ul style="list-style-type: none">• Remote consultation between the pediatrician and the parents with infants.• Predict the sickness and suggest the solutions and remedies.	<ul style="list-style-type: none">• Video processing participate only parents with the infants• Only identify the sickness and abnormal behavior and notify it through the system

Information Gathering

Interviews

- Deputy Director - Lady Ridgeway Hospital
- Dr. Uditha - IT Consultant (LRH)
- Dr. Deepal - Pediatrics (LRH)
- The pediatric clinic was visited at LRH
- Visited the Castle Hospital and handed over the necessary documentation for approval to do field visits



IT20015244 | Wijesinghe R.M.U.S



Decentralized Patient Registration.

Software Engineering

Background

Decentralized Patient (Newborns) Registration (DPR)

- After the proposal was submitted, the panel asked for an outstanding novelty to be highlighted. Information was gathered from experts and doctors to gather information on how to develop the registered patient, doctor and hospital. And if you want to view the patient records, you need to request the public key from the patient. They said to collect new novelty like this.
- I would implement a permission-based access control mechanism to ensure that only authorized parties can access the data.
- I implement the system to allow parents and health care providers to manage a child's medical and personal information in an efficient and accessible manner. For example, if a child needs to see a specialist, the specialist can quickly access the child's medical history and provide the necessary treatment.
- By encapsulating all of these features into a single component, the DPR component could be easily integrated into a larger decentralized healthcare system.

System Workflow

- **In this application is powered by IPFS, where patients' medical records are stored on the distributed file system, not owned by any centralized entity. A patient can access his or her records by interacting with a smart contract on the Ethereum blockchain, forming a digital identity of the patient on the decentralized network.**
- **The client first connects with MetaMask, and uses smart contract to mint a patient or doctor block, registered by the wallet address.**
- **The client can upload a record file to IPFS, which address is linked to a patient block in ETH chain. The client can get all record addressed stored in a patient block from smart contract, and get a record file by its address from IPFS.**
- **The health provider can search for a patient's records using the address, and upload a new record for the patient. The patient can also view his or her records.**
- **A patient or a doctor can access the patient's records by interacting with a smart contract on the Ethereum blockchain.**

Research Gap

Research paper title	year	Used Ethereum Blockchain for Decentralized system	Used interacting with the Ethereum network using Web3.js	Accessible only to authorized parties, such as parents or healthcare providers.
01-Centralised versus Decentralized Management of Patients' Medical Records	2009	No(only research paper)	No	Yes
02-Decentralized Patient-Centric Report and Medical Image Management System Based on Blockchain Technology and the Inter-Planetary File System.	2022	Yes (Only focuses in lab test , not implement and)	Yes	Yes
03-Decentralized Electronic Medical Records.	2019	Yes(Only focuses on the adults used in hyperledger)	-	Yes
04-Digital and Decentralized Management of Patient Data in Healthcare Using Blockchain Implementations.	2021	Yes(Only focuses on the adults not implement,only research paper)	Yes	Yes

Research Question

- The existing electronic health records management systems suffer from security and privacy vulnerabilities, as patient data is often stored in centralized databases prone to unauthorized access and data breaches. Our research project aims to address this critical issue by leveraging the decentralized and immutable nature of blockchain technology.
- How will the system ensure the privacy and security of the data collected?
- How will the system ensure that authorized parties have access to the necessary information?
- How will the system interface with existing healthcare systems and databases?

Research Novelty

- Permission-based access control to ensure that only authorized parties can access the data.
- Our solution incorporates advanced cryptographic techniques such as zero-knowledge proofs and homomorphic encryption to enable secure and privacy-preserving data sharing among authorized healthcare providers while keeping patient identities and sensitive information confidential.
- The use of blockchain technology ensures that the data is immutable and tamper-proof, providing a high level of security.
- We introduce a novel access control mechanism using smart contracts, allowing patients to grant granular access permissions to healthcare providers on a need-to-know basis. The access control rules are enforced autonomously, ensuring data integrity and preventing unauthorized data access.

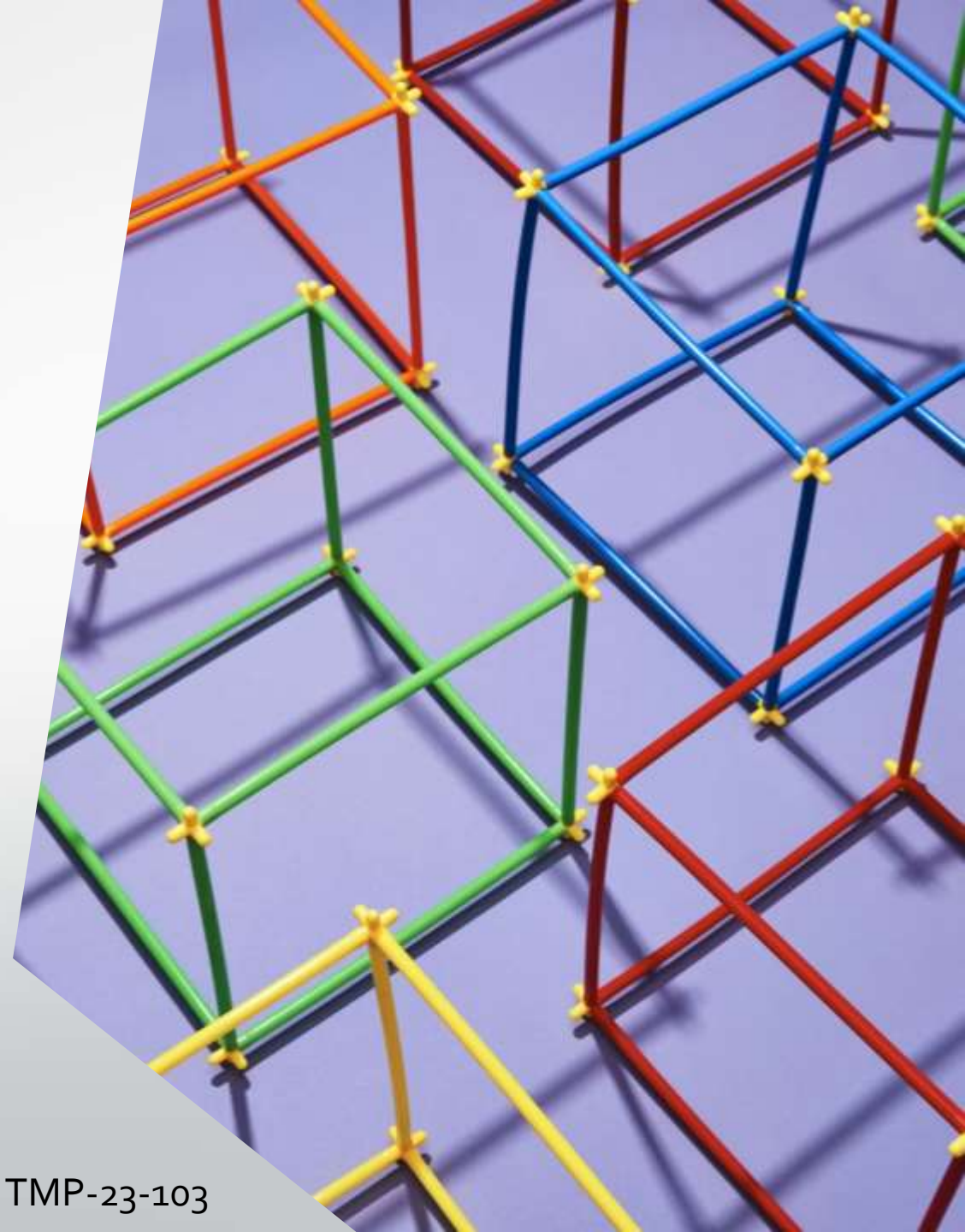
Specific Objectives & sub-Objectives

Specific Objective

The goal of using a decentralized system for the registration of newborns and parents' details is to provide a secure, transparent, and efficient way to manage critical information related to newborns and their parents.

Sub Objectives

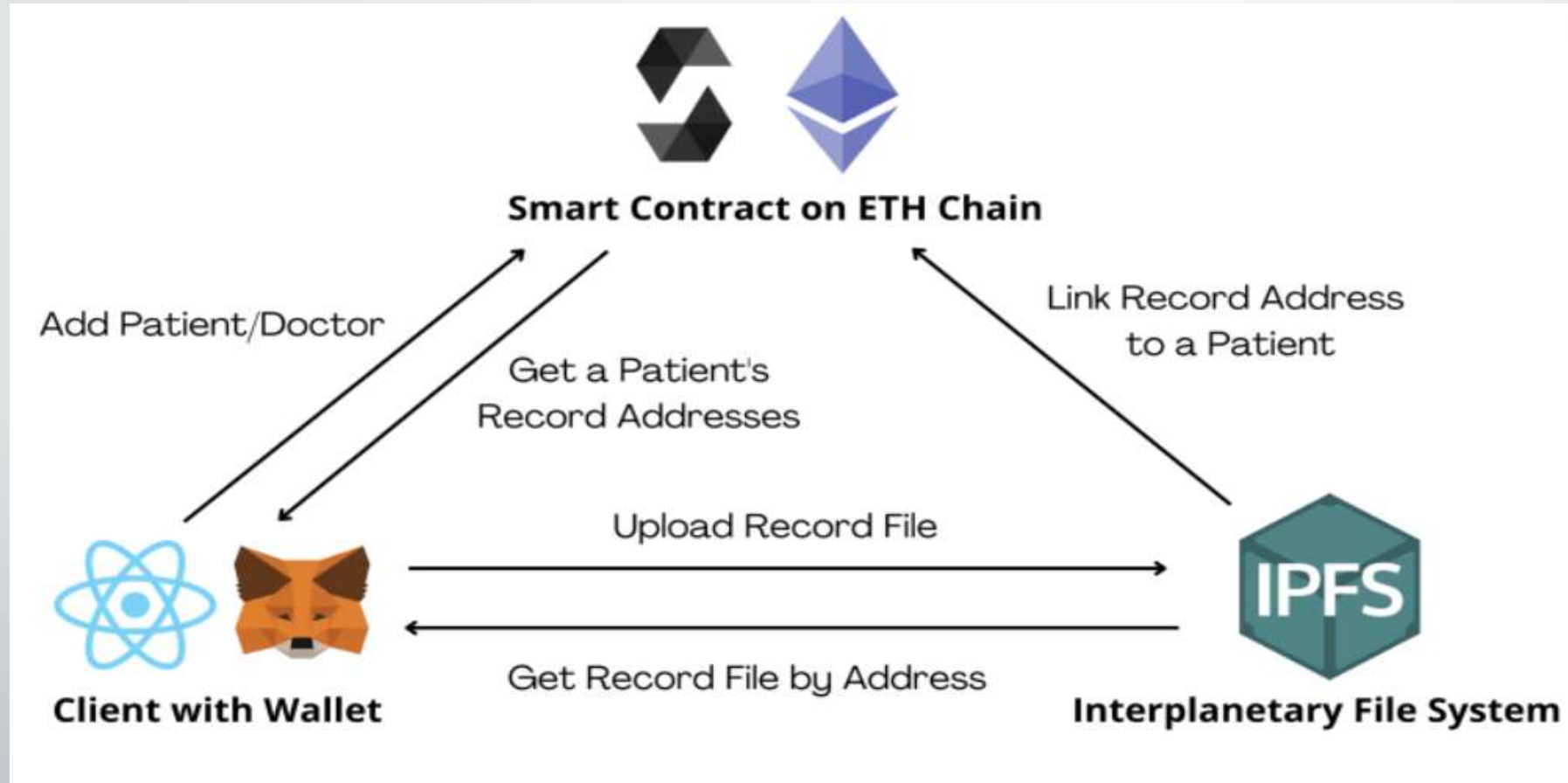
- By leveraging decentralized technologies such as blockchain or decentralized databases, this system can enhance security and privacy, prevent data tampering, and improve data accessibility.
- The smart contract stores the newborns' and parents' details on the Ethereum blockchain, ensuring that the data is secure and immutable.
- This system also allows parents and healthcare providers to manage the child's medical and personal information in a transparent and secure way, while providing access to relevant parties as needed.



Methodology

Component overview Diagram

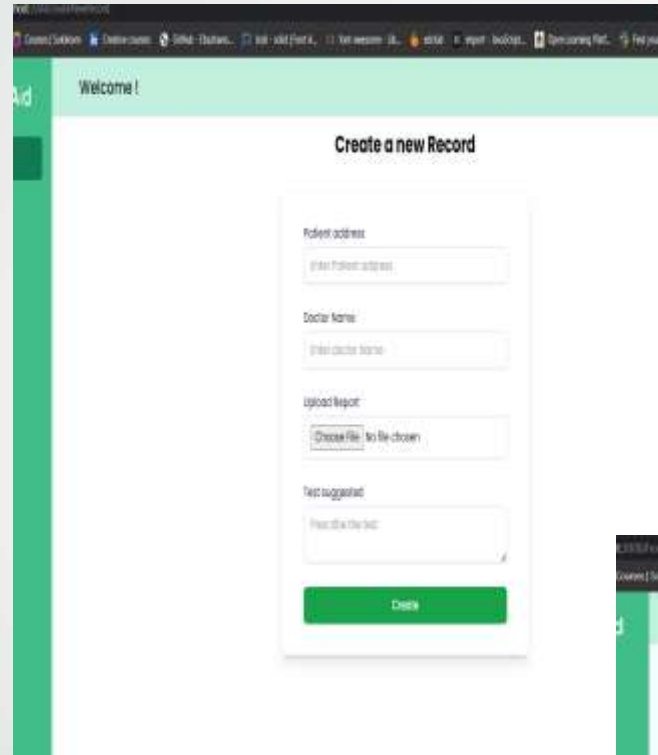
Component Overview Diagram



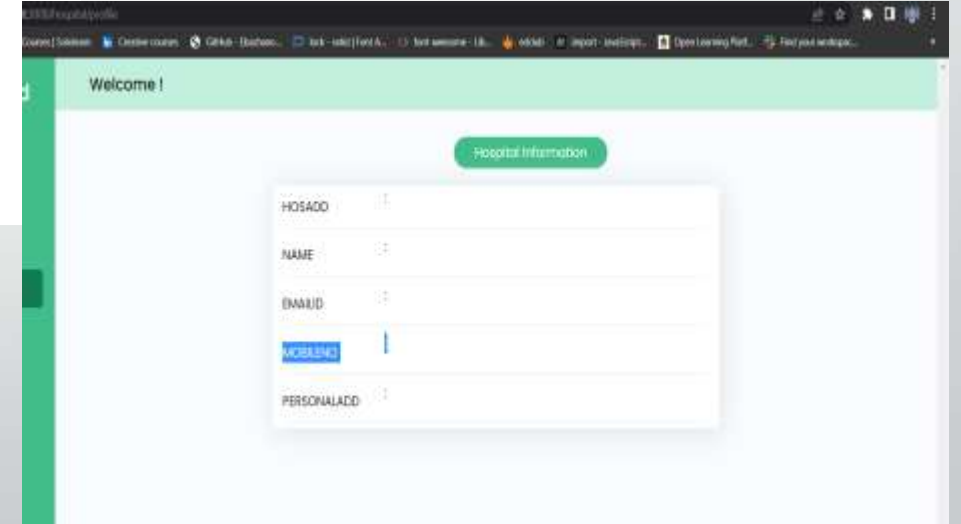
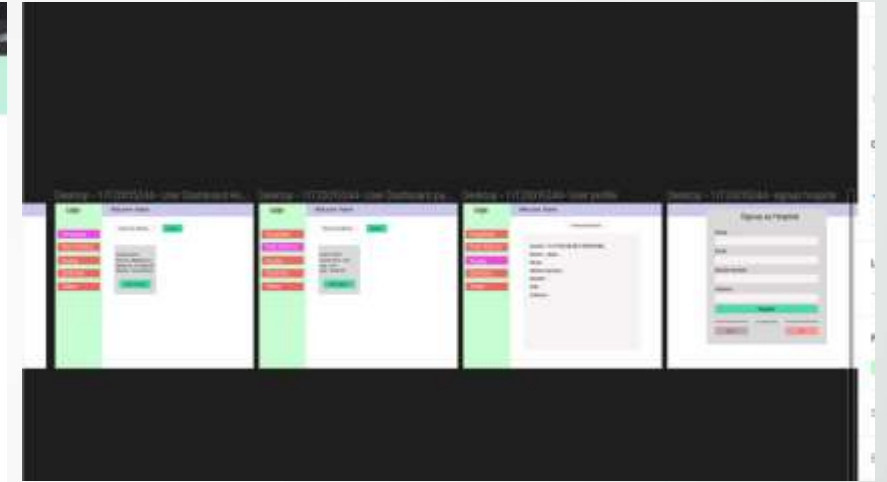
Implementation

Developing Frontend and Backend

- Frontend with: HTML, CSS, ReactJs, Tailwind.
- Backend with : Blockchain
- Storage : Web3.Storage, InterPlanetary File System(IPFS)



A screenshot of a web application interface showing a form titled "Create a new Record". The form is located on a page with a green header bar containing the text "Welcome!". The form fields include: "Patient address" (with a placeholder "Enter Patient address"), "Doctor Name" (with a placeholder "Enter doctor name"), "upload Report" (with a "Choose File" button and "No file chosen" text), and "Test suggested" (with a placeholder "Prescribe the test"). A green "Create" button is at the bottom of the form.

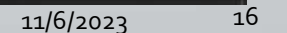


A screenshot of a web application interface showing a form titled "Hospital Information". The form is located on a page with a green header bar containing the text "Welcome!". The form fields include: "HOSPITAL" (with a placeholder "Enter hospital name"), "NAME" (with a placeholder "Enter name"), "EMAIL" (with a placeholder "Enter email"), "MOBILENO" (with a placeholder "Enter mobile number"), and "PERSONALADD" (with a placeholder "Enter personal address").

11/6/2023

15

Used Dashboard



Implementation

Hospital Dashboard

The screenshot shows the Visual Studio Code editor with the file explorer on the left, the editor window in the center, and the terminal at the bottom. The file explorer shows the project structure with folders like 'contracts', 'public', 'scripts', 'src', and 'components'. The 'components' folder is expanded, showing 'HospitalSidebar' and 'Sidebar'. The 'HospitalSidebar' folder contains 'hosSidebar.js', 'HSidebar.jsx', and 'Sidebar.mod...'. The 'hosSidebar.js' file is selected and its content is displayed in the editor window. The code defines a constant 'hosSidebar' array with four objects, each representing a sidebar item with a name, url, and icon. The terminal at the bottom shows the output of the webpack build process, indicating that the build was successful.

```
src > components > HospitalSidebar > hosSidebar.js > 60 hosSidebar
import { LocalHospitalIcon, HistoryEduIcon, BiotechIcon, Person2Icon } from '@mui/icons-material';
import styles from './Sidebar.module.css';

import styles from './Sidebar.module.css';
export const hosSidebar = [
  {
    name: "New Record",
    url: "/createNewRecord",
    icon: <LocalHospitalIcon className={styles.listIcon} />,
  },
  {
    name: "Past History",
    url: "/hospital/history",
    icon: <HistoryEduIcon className={styles.listIcon} />,
  },
  {
    name: "Users",
    url: "/hospital/users",
    icon: <BiotechIcon className={styles.listIcon} />,
  },
  {
    name: "Profile",
    url: "/hospital/profile",
    icon: <Person2Icon className={styles.listIcon} />,
  },
];
```

TERMINAL

You can now view **powering_stem** in the browser.

Local: http://localhost:3000
On Your Network: http://192.168.8.176:3000

Note that the development build is not optimized.
To create a production build, use `npm run build`.

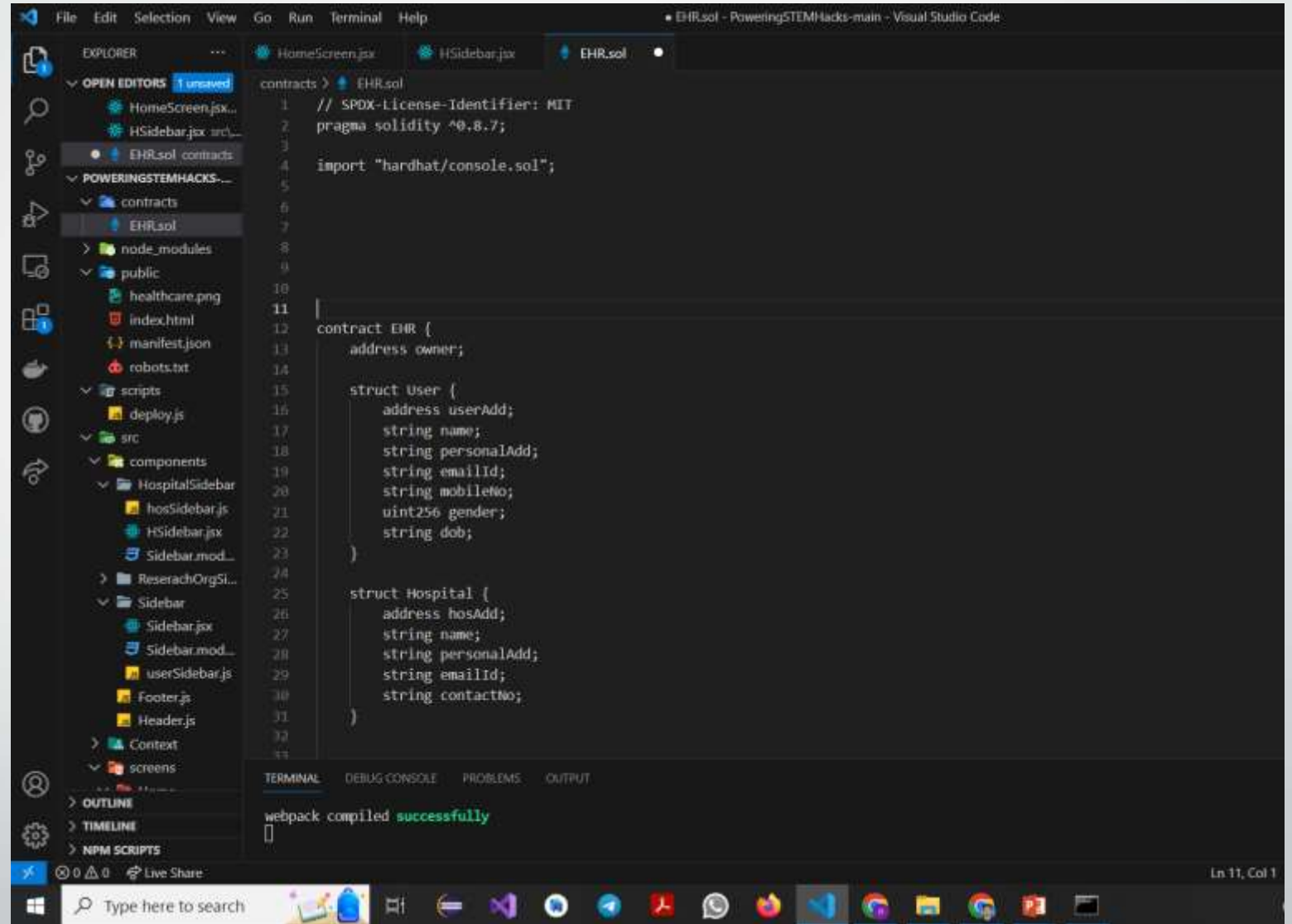
webpack compiled **successfully**

11/6/2023

17

Implementation

Smart Contract code



The screenshot displays the Visual Studio Code interface with the following components:

- EXPLORER:** Shows the project structure for 'PoweringSTEMHacks-main'. The 'contracts' folder is expanded, showing 'EHR.sol'.
- EDITOR:** Displays the Solidity code for 'EHR.sol'. The code includes a license identifier, pragma statement, and imports.
- TERMINAL:** Shows the output 'webpack compiled successfully'.

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.7;
3
4 import "hardhat/console.sol";
5
6
7
8
9
10
11
12 contract EHR {
13     address owner;
14
15     struct User {
16         address userAdd;
17         string name;
18         string personalAdd;
19         string emailId;
20         string mobileNo;
21         uint256 gender;
22         string dob;
23     }
24
25     struct Hospital {
26         address hosAdd;
27         string name;
28         string personalAdd;
29         string emailId;
30         string contactNo;
31     }
32
33 }
```


Technology Involved

Frontend

- React

Backend

- Node.js and Express
- Python, Springboots

Libraries

- Web3.js Ethereum network
- Mongoose and Axios

Database

- MongoDB, MySql
- Ethereum Based Blockchain



Project Requirements

- **Functional requirements**

- Distributed storage: The system must be able to store data in a distributed manner, with multiple copies of data spread across nodes in the network.
- Decentralized consensus: The system must use a consensus algorithm to agree on the state of the network without relying on a central authority. This could be achieved using a blockchain, DAG, or other consensus mechanisms.
- P2P communication: The system must be able to facilitate communication between nodes without relying on a centralized server. This could be achieved using a peer-to-peer (P2P) network.
- User authentication and access control: The system must provide secure authentication and access control mechanisms to ensure that only authorized users can access and modify data

- **Non-Functional requirements**

Accuracy, Reliability, Performance, Usability , Accessibility,

References

1. Secure decentralized electronic health records sharing system based on blockchains:
<https://www.frontiersin.org/articles/10.3389/fbloc.2021.732112/full>
2. Secure decentralized electronic health records sharing system based on blockchains:
<https://www.sciencedirect.com/science/article/pii/S1319157821001051>.
3. C C Darshan Thimmaiah, D. S. (2019). Decentralized Electronic Medical Records. IJRAR, 7.
4. Catherine Quantin, G. C. (2009). Centralised versus Decentralised Management of Patients' Medical Records. researchgate, 6.
5. Jihui Shi 1, S. K. (2022). A Novel Block Chain Method for Urban Digitization. International Journal of Environmental, 19.
6. Syed Agha Hassnain Mohsan, A. R. (2021). Decentralized Patient-Centric Report and Medical Image. International Journal of Environmental, 18.
7. Zhang P, White J, Schmidt DC, Lenz G and Rosenbloom (2018), FHIR Chain: Applying Blockchain to Securely and Scalably Share, Computational and Structural Biotechnology Journal, 16: 267-278
8. Crypt Bytes Tech (2017), Medical Chain - A blockchain for electronic health records. [https:// medium.com/crypt-bytes-tech/medicalchain-a-blockchain-for-electronic-health-recordseef181ed14c2](https://medium.com/crypt-bytes-tech/medicalchain-a-blockchain-for-electronic-health-recordseef181ed14c2).
9. Ekblaw A, Azaria A, Halamka JD and Lippman A (2016), A Case Study for Blockchain in Healthcare: “MedRec” prototype for electronic health records and medical research data. MIT Media Lab

IT20147778| Jayawickrama N.D.D



Health Informatics Chatbot Application

Software Engineering

Component Discussion

- **Introduction**

- The proposed component is to assist the parents of babies who are having a hard time managing their time.
- The system enables parents to ask questions about their infants' behaviors, any medicine, related remedies, or any other assumption or methods of how exactly to react to an emergency situation.
- By the image processing technique, the chatbot can identify the skin infections of the kid by analyzing a clear uploaded photo.
- Chat bot will also notify the vaccination days according to the data records it gets from the decentralized system.
- From this specific component, the main target is to introduce an assistant that is available beyond the time, helps to manage the time, to reduce the unnecessary cost of doctor appointments, and assist the parents in their problematic situations as a trained health care agent.

- **User Workflow**

- Parent approach to the chatbot provided the problematic situation they have faced.
- The chatbot system takes the input as data, asks relevant questions to understand the situation, and comes to a final prediction.
- Then the suggestions will be displayed as an answer to the problem. This suggestion can be advice, a prediction of a disease, a history record, a remedy, a medicine, or a contact detail of a relevant doctor.

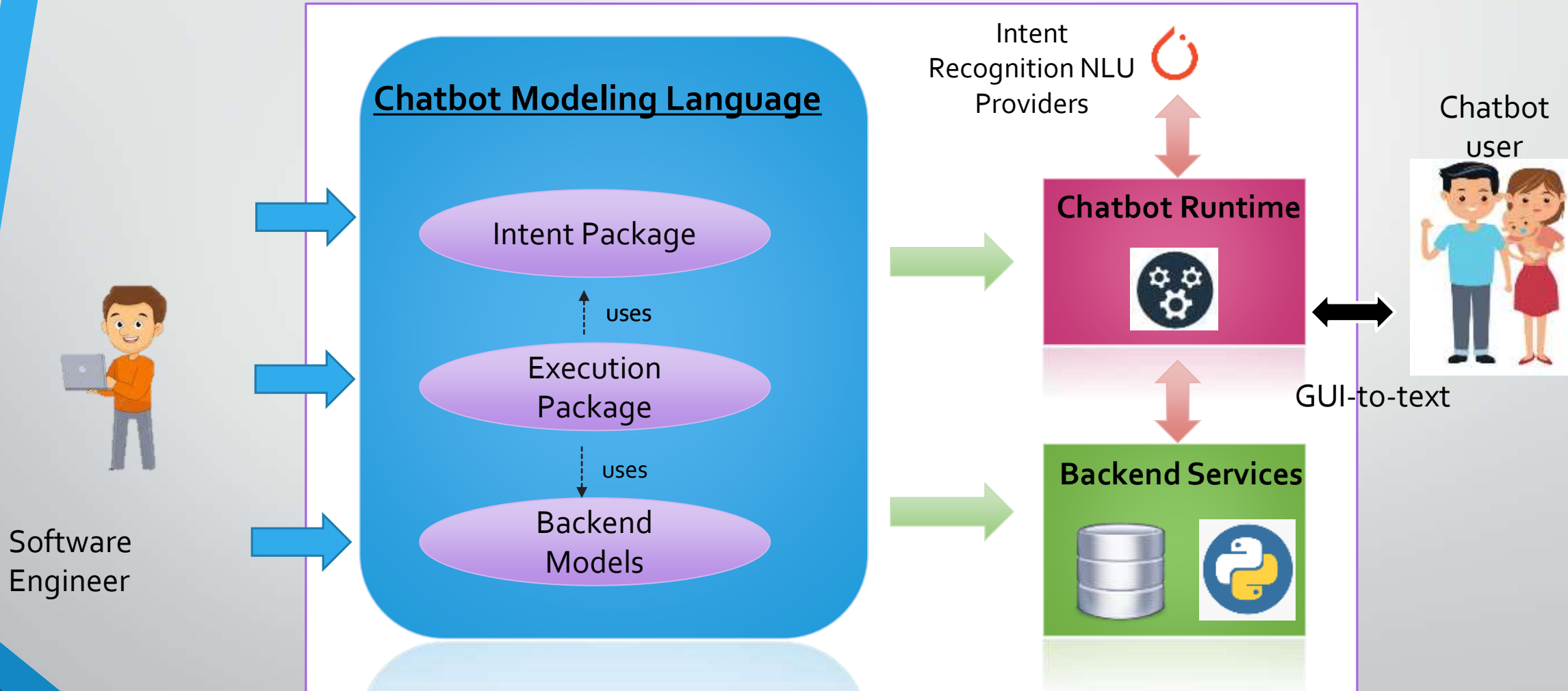
Component Discussion

- **Benefits for Parents**
- Time and cost Management: This system will reduce the unnecessary amount of time and the cost for the travel to hospital.
- Early Detection: By capturing and analyzing the parents' data, the system can predict the sicknesses and suggest solutions.
- Notification system: Parents receive notifications for the relevant vaccination dates.
- Personalization: Chatbot is created for each child, and it is unique for each user with the credentials
- Availability: Unlike doctors, the system is available for the user without any appointments and without time consideration.
- De-stress mental health: System existence is to decrease the problematic and emerging situation and to smoothly handle it.
- **Impact and Future Scope**
- The system touches the sensitive area of the health industry by predicting the sickness of infants.
- This method can be recognized as remote consultation via telecommunication, and with the reliability of the prediction, the system may perform an important role as an assistant for the parents of the newborn child.
- Through the training model that contributes with AI and ML algorithms, Image processing techniques, and NLP processing techniques, the accuracy, reliability, and user-friendliness of the system can be optimized.

Information Gathering

- **Number of interview sessions were conducted to finalize the component by the field specialists.**
- ✓ Dr. Christine Buddhini – Panadura Hospital
- ✓ Dr. Thilini Amarathunga - Muttur Hospital
- **Research was conducted after the comments of the panel and the specialists.**
 - Existing systems
 - Research papers
 - Video tutorials
- **For the implementation, tutorial videos, blogs, and chatbot documentation are followed.**

Component Overview Diagram



Model Implementation

Intents file

This file consists of the training dataset to train the model. All the data is hard-coded into a text file with tags, patterns, and corresponding responses

```
intents.json x chat.py
intents.json > [ ] intents > { } 6 > [ ] responses > 0
1 {
2   "intents": [
3     {
4       "tag": "greeting",
5       "patterns": [
6         "Hi",
7         "Hey",
8         "How are you",
9         "Is anyone there?",
10        "Hello",
11        "Good day",
12        "Hy",
13        "I need a favour",
14        "can you help me?",
15        "I need you assist",
16        "help me",
17        "i need a helo"
18      ],
19      "responses": [
20        "Hey :-)",
21        "Hello, how can I help you today?",
22        "Hi there, what can I do for you?",
23        "Hi there",
24        "Hi user! tell me how can i assist you?"
25      ]
26    },
27  ],
28  {
29    "tag": "goodbye",
30    "patterns": ["Bye", "See you later", "Goodbye"],
31    "responses": [
32      "See you later",
33      "Have a nice day"
```

Model

This code uses forward propagation of neural network technique and **relu** function to apply weights, and bias terms in terms to reduce the difference between actual output and predict output

```
intents.json  model.py  chat.py

model.py > NeuralNet > __init__
1  import torch
2  import torch.nn as nn
3
4
5  # feedforward neural set
6  class NeuralNet(nn.Module):
7
8      def __init__(self, input_size, hidden_size, num_classes):
9          super(NeuralNet, self).__init__()
10         self.l1 = nn.Linear(input_size, hidden_size)
11         self.l2 = nn.Linear(hidden_size, hidden_size)
12         self.l3 = nn.Linear(hidden_size, num_classes)
13
14         self.relu = nn.ReLU()
15
16     def forward(self, x):
17         out = self.l1(x)
18         out = self.relu(out)
19         out = self.l2(out)
20         out = self.relu(out)
21         out = self.l3(out)
22         # no activation and no softmax
23
24         return out
```


NLTK

NLTK stands for Natural Language Toolkit. This technique is used for tokenization, stemming, and getting the bag of words

Tokenization -Split strings into meaningful units

Stemming –Generate the root form by cutting the end off

Bag_of_words – the final outcome of binary values of stemmed words

```
intents.json  nltk_utils.py  model.py  chat.py
nltk_utils.py > ...
1  import nltk
2  from nltk.stem.porter import PorterStemmer
3
4  import numpy as np
5  #nltk.download('punkt') #download packed for pre train tokenizer (only need once)
6
7  import torch
8  #import neural network package
9  import torch.nn as nn
10
11  from torch.utils.data import Dataset, DataLoader
12
13  stemmer = PorterStemmer()
14
15  # method to get sentence
16  def tokenize(sentence):
17      return nltk.word_tokenize(sentence) #return the tokenized words
18
19  # method to stemming
20  def stem(word):
21      return stemmer.stem(word.lower())
22
23  # method for bag of words
24  def bag_of_words(tokenized_sentence, all_words):
25
26      tokenized_sentence= [stem(w) for w in tokenized_sentence]
27      bag = np.zeros(len(all_words), dtype = np.float32)
28
29
30      #enumerate is to get list number and the value individually
31      for idx, w in enumerate(all_words) :
32          if w in tokenized_sentence:
33              bag[idx]=1.0
34
35      return bag
36
37
```

Training file

- The intent file,
- NLTK,
- Numpy,
- Pytorch,
- The dataset
- the model,

of the system are imported into the training file. The file is to train the model by the dataset by using each technique

- **numpy**: A library for numerical operations and array manipulation.
- **Pytorch**: open-source machine learning library that provides a flexible framework for building and training deep learning models efficiently.
- **Epoch**: refers to a single pass through the entire training dataset during the training process of a model. It helps optimize the model's parameters by updating them based on the calculated loss and the chosen optimization algorithm.

```

66 batch_size = 8
67 hidden_size = 8
68 output_size = len(tags)
69 input_size = len(X_train[0])
70 learning_rate = 0.001
71 #complete iteration of a dataset during training phase
72 num_epochs = 1000
73
74 # print(input_size, output_size)
75
76 class ChatDataset(Dataset):
77
78     def __init__(self):
79         self.n_samples = len(X_train)
80         self.x_data = X_train
81         self.y_data = y_train
82
83     # support indexing such that dataset[i] can be used to get i-th sample
84     def __getitem__(self, index):
85         return self.x_data[index], self.y_data[index]
86
87     # we can call len(dataset) to return the size
88     def __len__(self):
89         return self.n_samples
90
91 dataset = ChatDataset()
92 train_loader = DataLoader(dataset=dataset,
93                             batch_size=batch_size,
94                             shuffle=True,

```

```

33         xy.append((w, tag))
34
35 # stem and lower each word
36 #omit the signs and get the words out from sentence
37 ignore_words = ['?', '.', '!']
38 all_words = [stem(w) for w in all_words if w not in ignore_words]
39
40 #sorting to ignore multiplication of words
41 all_words = sorted(set(all_words))
42 tags = sorted(set(tags))
43 #print(tags)
44
45 # print(len(xy), "patterns")
46 # print(len(tags), "tags:", tags)
47 # print(len(all_words), "unique stemmed words:", all_words)
48
49 # create training data
50 X_train = []
51 y_train = []
52 for (pattern_sentence, tag) in xy:
53     # x: bag of words for each pattern_sentence
54     bag = bag_of_words(pattern_sentence, all_words)
55     X_train.append(bag)
56     # y: PyTorch CrossEntropyLoss needs only class labels, not one-hot
57     # numbering the words in order
58     label = tags.index(tag)
59     y_train.append(label)
60
61 X_train = np.array(X_train)

```

```

trainpy > _
1 import numpy as np
2 import random
3 import json
4
5 import torch
6 import torch.nn as nn
7 from torch.utils.data import Dataset, DataLoader
8
9 from nltk_utils import bag_of_words, tokenize, stem
10 from model import NeuralNet
11
12 with open('intents.json', 'r') as f:
13     intents = json.load(f)
14     # print(intents) #-> to check
15
16 all_words = [] #for all the words from sentences
17 tags = [] # tags as in intents file
18 xy = [] #empty list that will be hold tags and patterns after the process
19
20 # loop through each sentence in our intents patterns
21 # with the key "intents", all there in intents file
22 for intent in intents['intents']:
23     tag = intent['tag']
24     # add to tag list
25     tags.append(tag)
26     for pattern in intent['patterns']:
27         # tokenize each word in the sentence
28         w = tokenize(pattern)
29         # add to our words list

```

```

intents.json  nltk_utils.py  train.py  X  chat.py
trainpy > _
31 dataset = ChatDataset()
32 train_loader = DataLoader(dataset=dataset,
33                             batch_size=batch_size,
34                             shuffle=True,
35                             num_workers=0)
36
37 device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
38
39 model = NeuralNet(input_size, hidden_size, output_size).to(device)
40
41 # Loss and optimizer
42 criterion = nn.CrossEntropyLoss()
43 optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
44
45 # Train the model
46 for epoch in range(num_epochs):
47     for (words, labels) in train_loader:
48         words = words.to(device)
49         labels = labels.to(dtype=torch.long).to(device)
50
51         # Forward pass
52         outputs = model(words)
53         # if y would be one-hot, we must apply
54         # labels = torch.max(labels, 1)[1]
55         loss = criterion(outputs, labels)
56
57         # Backward and optimize

```

```

intents.json  nltk_utils.py  train.py  X  chat.py
trainpy > _
121
122     if (epoch+1) % 100 == 0:
123         print(f'Epoch [{epoch+1}/{num_epochs}], Loss: {loss.item():.4f}')
124
125
126 print(f'final loss: {loss.item():.4f}')
127
128 # create dictionary to save the data
129 data = {
130     "model_state": model.state_dict(),
131     "input_size": input_size,
132     "hidden_size": hidden_size,
133     "output_size": output_size,
134     "all_words": all_words,
135     "tags": tags
136 }
137
138 FILE = "data.pth"
139 #serialized and save in to pickled file
140 torch.save(data, FILE)
141
142 print(f'training complete. file saved to {FILE}')

```

Chat file

This code uses by the user interaction

```
intents.json  nltk_utils.py  train.py  chat.py X
chat.py > ...
1  import random #to make random choice from possible answers
2  import json
3  import torch
4  from model import NeuralNet
5  from nltk_utils import bag_of_words, tokenize
6
7
8  device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
9
10 with open('intents.json','r') as a:
11     intents = json.load(a)
12
13 #load the saved data from data file
14 FILE = "data.pth"
15 data= torch.load(FILE)
16
17 input_size = data["input_size"]
18 hidden_size = data["hidden_size"]
19 output_size = data["output_size"]
20 all_words = data["all_words"]
21 tags = data["tags"]
22 model_state = data["model_state"]
23
24 model = NeuralNet(input_size, hidden_size, output_size).to(device)
25 model.load_state_dict(model_state)
26 model.eval()
27
28 bot_name = "Nanny_Bot"
29 print("Let's chat! type 'quit' to exit")
30
```

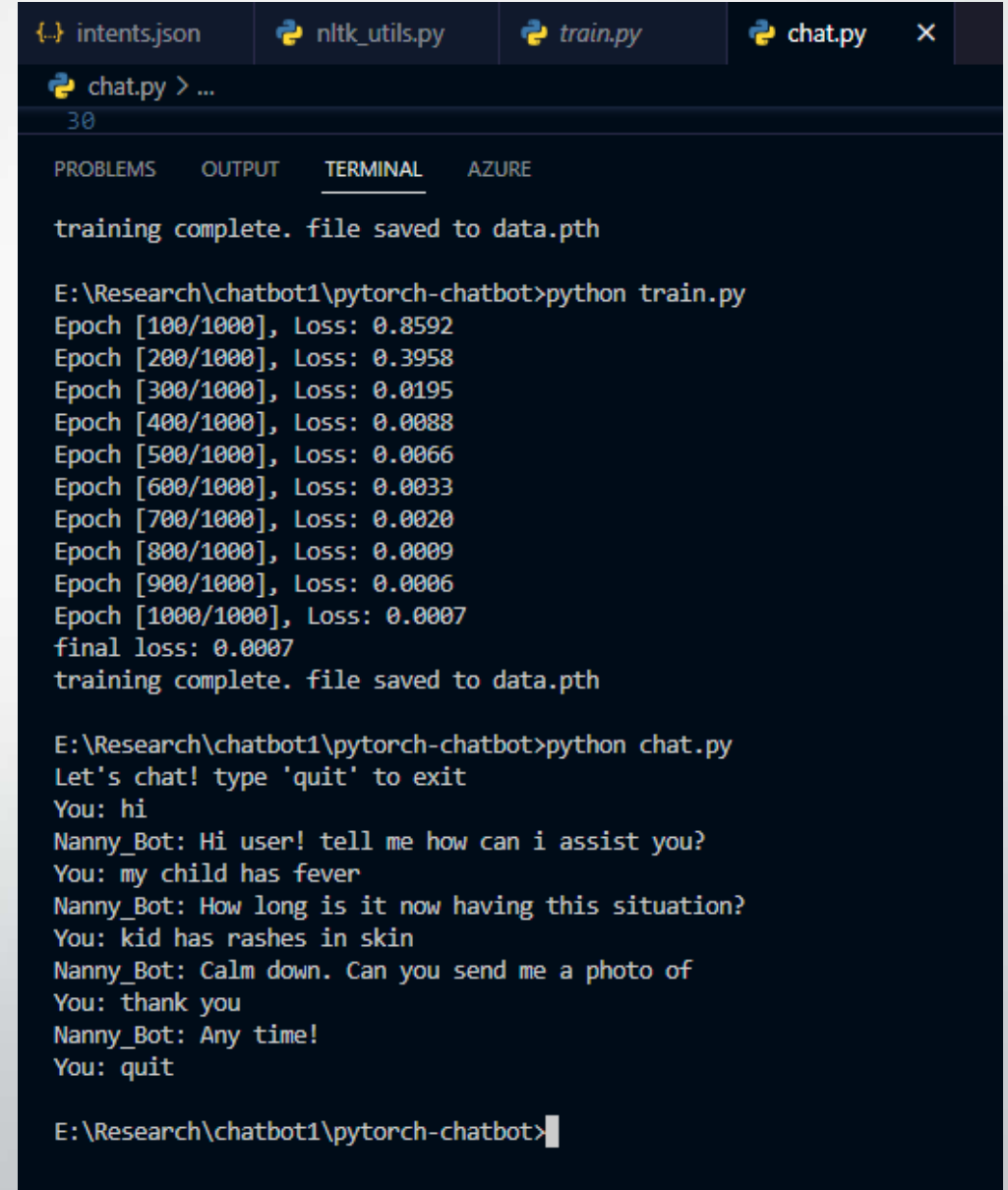
```
intents.json  nltk_utils.py  train.py  chat.py X
chat.py > ...
30
31 while True:
32     sentence = input('You: ')
33     if sentence == "quit": break
34
35     sentence = tokenize(sentence)
36     X = bag_of_words(sentence, all_words)
37     X = X.reshape(1, X.shape[0])
38     X = torch.from_numpy(X)
39
40     output = model(X)
41     _, predicted = torch.max(output, dim=1)
42     tag= tags[predicted.item()]
43
44     probs = torch.softmax(output, dim=1)
45     prob = probs[0][predicted.item()]
46
47
48     if prob.item() > 0.75:
49         for intent in intents["intents"]:
50             if tag == intent["tag"]:
51                 print(f"{bot_name}: {random.choice(intent['responses'])}")
52
53     else :
54         print(f"{bot_name}: I'm sorry, but im having problem understanding you... can you repeate again?")
```

Final output

- Training result

```
E:\Research\chatbot1\pytorch-chatbot>python train.py
Epoch [100/1000], Loss: 1.0175
Epoch [200/1000], Loss: 0.0666
Epoch [300/1000], Loss: 0.0782
Epoch [400/1000], Loss: 0.0036
Epoch [500/1000], Loss: 0.0052
Epoch [600/1000], Loss: 0.0009
Epoch [700/1000], Loss: 0.0008
Epoch [800/1000], Loss: 0.0012
Epoch [900/1000], Loss: 0.0005
Epoch [1000/1000], Loss: 0.0003
final loss: 0.0003
training complete. file saved to data.pth

E:\Research\chatbot1\pytorch-chatbot>
```



The screenshot shows a code editor with four tabs: `intents.json`, `nlTK_utils.py`, `train.py`, and `chat.py`. The `chat.py` tab is active, displaying the output of the chatbot. The output shows the training process completed, with the file saved to `data.pth`. It then shows the chatbot's response to a series of user inputs.

```
chat.py > ...
30

PROBLEMS OUTPUT TERMINAL AZURE

training complete. file saved to data.pth

E:\Research\chatbot1\pytorch-chatbot>python train.py
Epoch [100/1000], Loss: 0.8592
Epoch [200/1000], Loss: 0.3958
Epoch [300/1000], Loss: 0.0195
Epoch [400/1000], Loss: 0.0088
Epoch [500/1000], Loss: 0.0066
Epoch [600/1000], Loss: 0.0033
Epoch [700/1000], Loss: 0.0020
Epoch [800/1000], Loss: 0.0009
Epoch [900/1000], Loss: 0.0006
Epoch [1000/1000], Loss: 0.0007
final loss: 0.0007
training complete. file saved to data.pth

E:\Research\chatbot1\pytorch-chatbot>python chat.py
Let's chat! type 'quit' to exit
You: hi
Nanny_Bot: Hi user! tell me how can i assist you?
You: my child has fever
Nanny_Bot: How long is it now having this situation?
You: kid has rashes in skin
Nanny_Bot: Calm down. Can you send me a photo of
You: thank you
Nanny_Bot: Any time!
You: quit

E:\Research\chatbot1\pytorch-chatbot>
```




IT20405090 | Vanhoff R. L.

Growth Level Prediction

Software Engineering

Background

- After the proposal presentation the component was asked to change to a growth level predictor instead of a nutritional level identification and the functionality was changed from image processing to training a model to predict the growth level, by the panel.
- Information was gathered from the specialists and doctors in order to gather information for the predictions to make it more accurate and to discuss on how the functionality can be done.
- But after the process, it was stated that the growth level cannot be measured without the height, weight and age-appropriate other measures with just the behavioral milestones.
- As the final decision, the component was changed to a prediction function where according to the given measures of a child, the growth level can be predicted by the model.

Information Gathering

- Several interviews were held with the doctors to discuss about the changed function before finalizing the component.
- ✓ **Dr. Udara Ariyasinghe - Arogya Family Medical Center**
- ✓ **Dr. Thilini Amarathunga - Muttur Hospital**
- Investigations were done by exploring the available systems, past research papers, tutorials and other related materials to get a clear understanding about doing predictions using a model.
- Referred online materials such as YouTube videos, online surveys, WHO documentation and other medicine related documentation to use the correct information to train the model and start the implementations.

Component Discussion

The Function's Primary Goal - The purpose is to give parents an automated method for determining and tracking their infants' growth levels. The function estimates the developmental stage based on several factors about a kid, including weight, height, age, and other development milestones.

The following are the function's main objectives:

- **Development Level Prediction:** The Decision Tree Regressor method, a machine learning approach, is used by the function to forecast newborns' development levels. The function trains a regression model to precisely forecast the growth level based on the input measurements by examining the provided dataset, which comprises development milestones data for children aged 0–5 years.
- **Precautionary Measures:** If the expected growth level suggests overgrowth or undergrowth, the function is intended to help parents take the appropriate safety measures. Parents can better comprehend their baby's development and seek the necessary medical advice or intervention by having insights about the child's growth trajectory.
- **User-Friendly Interfaces:** The function is incorporated into a user-friendly front-end web application that is a full-stack web application. Parents may easily input their baby's measurements into the interface to get a prediction of growth level. To make it simple for parents to understand and respond to the results, the interface also displays the forecast as a % along with corresponding categories of poor, fair, good, and excellent.

By emphasizing these crucial elements, the function intends to give parents a useful and instructive tool for tracking their child's development and encouraging proactive healthcare management.

Component Discussion

User Workflow

- **Enter Child's Information:** Provide the child's age, height, weight, and select or input information about their developmental milestones in various domains, such as motor skills, problem-solving, communication, and emotional development.
- **Predict Growth Level:** Click the "Predict the Growth" button to initiate the prediction process. The system will process the entered data and calculate the child's growth level.
- **View Result:** Once the prediction is complete, view the predicted growth level displayed on the screen. Take note of the result and any recommendations provided by the application.

Impact and Future Scope

Impact:

- Empowers parents with valuable insights into their child's progress.
- Enables early detection of developmental delays or potential issues.
- Facilitates timely interventions and support.
- Promotes proactive parenting by offering personalized recommendations.
- Contributes to informed decision-making for child development.

Future Scope:

- Calculation of different milestones and personalized recommendations.
- Expansion of assessed milestones and incorporation of machine learning techniques.
- Integration of a comprehensive growth database for accurate comparisons.
- Inclusion of educational resources and interactive features.
- Continuous evolution and advancements as a valuable parenting tool.

System Workflow

- **Data collection:** The system begins by gathering a dataset that includes information on the developmental stages of children aged 0 to 5. The growth prediction model is trained using the dataset as its foundation.
- **Data preprocessing:** To extract pertinent characteristics, the collected dataset is preprocessed. The algorithm in this instance chooses particular metrics like weight, height, age, emotional development, social skills, emotional issues, and growth rate. In addition to addressing missing values, categorical variables must be encoded using LabelEncoder, and the data must be split into training and testing sets using train_test_split.
- **Model Training:** The system trains the growth predictor model using the DecisionTreeRegressor method from the scikit-learn package. The selected characteristics and accompanying growth levels are included in the training dataset, which is used to train the model.
- **Model Evaluation:** Using the testing dataset, the model is assessed after training. To evaluate the model's success, the system predicts the growth levels for the testing data and contrasts them with the actual growth levels. The accuracy_score measure is used to determine the model's accuracy.

System Workflow

- **Model Persistence:** After the model has been trained and assessed, it is stored using the pickle library in a serialized manner. The model is saved in a file along with the label encoders required to encode categorical variables.
- **Front-End Integration:** A full-stack online application incorporates the growth prediction model. Parents can pick certain observations or activities pertaining to their baby's growth and development using checkboxes on the front-end interface.
- **Prediction Calculation:** The system determines the score based on the number of selected actions when the user checks the appropriate checkboxes. The ratio of the number of chosen activities to the total number of potential actions is multiplied by 100 to get the score, which is then expressed as a percentage.
- **Forecast Interpretation:** Based on the estimated %, the algorithm chooses one of four growth forecast categories (poor, fair, good, or great). For example, 80% for exceptional, 60% for good, and 40% for fair, are used as specific thresholds to determine which category a forecast belongs in.
- **Results Presentation:** The system shows the user the growth projection category. Based on the chosen observations, the category represents the baby's growth stage. On the front-end interface, the results are displayed in an approachable manner, giving parents useful information about their child's development and the essential safety measures to be implemented.

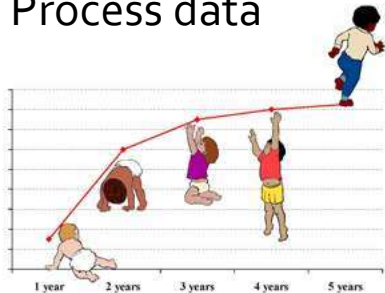
Component Overview Diagram



Input data will be entered by the parents



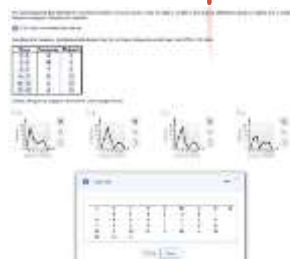
Process data



Output will be displayed as a prediction



Machine Learning Model



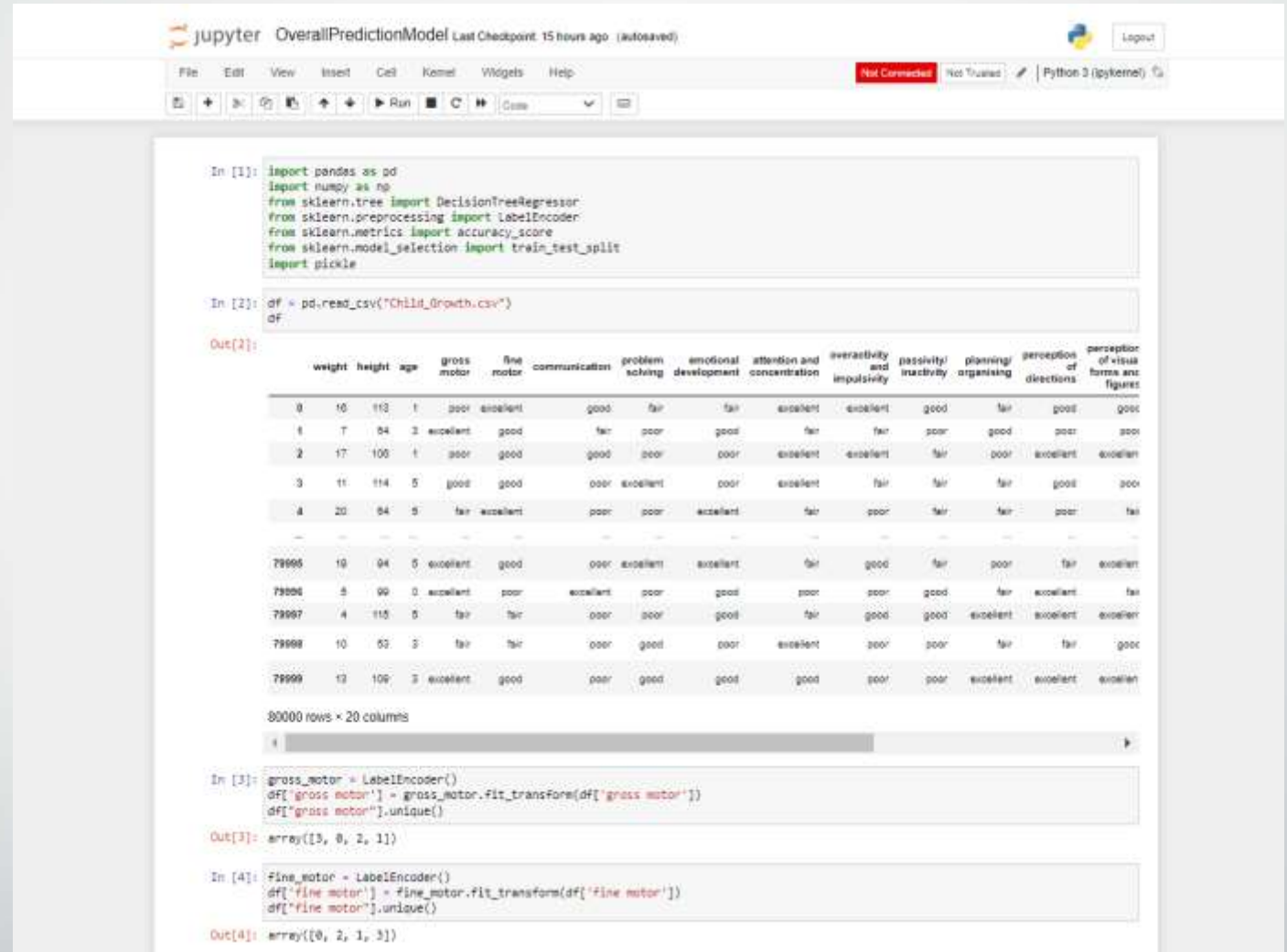
Train the Machine Learning model using a dataset

Implementation

Developing and training a model

Libraries:

- **numpy**: A library for numerical operations and array manipulation.
- **pandas**: A library for data manipulation and analysis.
- **sklearn.tree.DecisionTreeRegressor** or: A class from scikit-learn library for implementing decision tree regression.
- **sklearn.preprocessing.LabelEncoder**: A class from scikit-learn library for label encoding categorical variables.
- **sklearn.metrics.accuracy_score**: A function from scikit-learn library for computing accuracy scores.
- **sklearn.model_selection.train_test_split**: A function from scikit-learn library for splitting data into training and testing sets.
- **pickle**: A module for serializing and deserializing Python objects.



The screenshot shows a Jupyter Notebook titled "OverallPredictionModel" with a last checkpoint 15 hours ago. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a status bar indicating "Not Connected" and "Python 3 (ipykernel)".

The notebook contains the following code cells:

```
In [1]: import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeRegressor
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
import pickle
```

```
In [2]: df = pd.read_csv("Child_Growth.csv")
df
```

The output of In [2] is a DataFrame with 80,000 rows and 20 columns. The columns are: weight, height, age, gross motor, fine motor, communication, problem solving, emotional development, attention and concentration, overactivity and impulsivity, passivity/inactivity, planning/organizing, perception of directions, and perception of visual forms and figures. The first few rows of the DataFrame are displayed:

	weight	height	age	gross motor	fine motor	communication	problem solving	emotional development	attention and concentration	overactivity and impulsivity	passivity/inactivity	planning/organizing	perception of directions	perception of visual forms and figures
0	10	113	1	poor	excellent	good	fair	fair	excellent	excellent	good	fair	good	good
1	7	84	3	excellent	good	fair	poor	good	fair	fair	poor	good	poor	poor
2	17	100	1	poor	good	good	poor	poor	excellent	excellent	fair	poor	excellent	excellent
3	11	114	5	good	good	poor	excellent	poor	excellent	fair	fair	fair	good	good
4	20	94	5	fair	excellent	poor	poor	excellent	fair	poor	fair	fair	poor	fair

Below the DataFrame preview, it states "80000 rows x 20 columns".

```
In [3]: gross_motor = LabelEncoder()
df['gross motor'] = gross_motor.fit_transform(df['gross motor'])
df['gross motor'].unique()
```

```
Out[3]: array([3, 0, 2, 1])
```

```
In [4]: fine_motor = LabelEncoder()
df['fine motor'] = fine_motor.fit_transform(df['fine motor'])
df['fine motor'].unique()
```

```
Out[4]: array([0, 2, 1, 3])
```

```
jupyter OverallPredictionModel Last Checkpoint: 16 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Not Connected Not Trusted Python 3 (ipykernel)

In [5]: communication = LabelEncoder()
df['communication'] = communication.fit_transform(df['communication'])
df["communication"].unique()

Out[5]: array([2, 1, 3, 0])

In [6]: problem_solving = LabelEncoder()
df['problem solving'] = problem_solving.fit_transform(df['problem solving'])
df["problem solving"].unique()

Out[6]: array([1, 3, 0, 2])

In [7]: emotional_dev = LabelEncoder()
df['emotional development'] = emotional_dev.fit_transform(df['emotional development'])
df["emotional development"].unique()

Out[7]: array([1, 2, 3, 0])

In [8]: attention = LabelEncoder()
df['attention and concentration'] = attention.fit_transform(df['attention and concentration'])
df["attention and concentration"].unique()

Out[8]: array([0, 1, 2, 3])

In [9]: overactivity = LabelEncoder()
df['overactivity and impulsivity'] = overactivity.fit_transform(df['overactivity and impulsivity'])
df["overactivity and impulsivity"].unique()

Out[9]: array([0, 1, 3, 2])

In [12]: passivity = LabelEncoder()
df['passivity/ inactivity'] = passivity.fit_transform(df['passivity/ inactivity'])
df["passivity/ inactivity"].unique()

Out[12]: array([2, 3, 1, 0])

In [13]: planning = LabelEncoder()
df['planning/ organising'] = planning.fit_transform(df['planning/ organising'])
df["planning/ organising"].unique()

Out[13]: array([1, 2, 3, 0])

In [14]: perception = LabelEncoder()
df['perception of directions'] = perception.fit_transform(df['perception of directions'])
df["perception of directions"].unique()

Out[14]: array([2, 3, 0, 1])
```

Data set reference -

<https://www.kaggle.com/datasets/salmanahmad1980/child-growth-measurements>

- The code loads a dataset from a CSV file using `pd.read_csv` from pandas library.
- The dataset is then filtered to select specific columns relevant to the model.
- Missing values are dropped using **dropna** function.
- Categorical variables are encoded using **LabelEncoder**.

```
jupyter OverallPredictionModel Last Checkpoint: 16 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Not Connected Not Trusted Python 3 (ipykernel)

Out[14]: array([2, 3, 0, 1])

In [15]: perception_vf = LabelEncoder()
df['perception of visual forms and figures'] = perception_vf.fit_transform(df['perception of visual forms and figures'])
df["perception of visual forms and figures"].unique()

Out[15]: array([2, 3, 0, 1])

In [16]: memory = LabelEncoder()
df['memory'] = memory.fit_transform(df['memory'])
df["memory"].unique()

Out[16]: array([2, 3, 0, 1])

In [17]: spoken = LabelEncoder()
df['spoken language'] = spoken.fit_transform(df['spoken language'])
df["spoken language"].unique()

Out[17]: array([2, 3, 1, 0])

In [18]: reading = LabelEncoder()
df['reading/writing'] = reading.fit_transform(df['reading/writing'])
df["reading/writing"].unique()

Out[18]: array([1, 0, 3, 2])

In [19]: social_skills = LabelEncoder()
df['social skills'] = social_skills.fit_transform(df['social skills'])
df["social skills"].unique()

Out[19]: array([3, 1, 2, 0])

In [20]: emotional_prob = LabelEncoder()
df['emotional problems'] = emotional_prob.fit_transform(df['emotional problems'])
df["emotional problems"].unique()

Out[20]: array([0, 1])

In [21]: growth_r = LabelEncoder()
df['growth'] = growth_r.fit_transform(df['growth'])
df["growth"].unique()

Out[21]: array([0, 1, 2])
```


- The code splits the preprocessed data into input features (X) and the target variable (y).
- The `train_test_split` function is used to divide the data into training and testing sets, with a test size of 20% and a random state of 0.
- **A decision tree regressor model** is instantiated using **DecisionTreeRegressor**.
- The model is trained using the training data with the `fit` method.

```
In [23]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
dec_tree_reg = DecisionTreeRegressor(random_state=0)
```

```
In [24]: dec_tree_reg.fit(X_train, y_train)
y_pred_train = dec_tree_reg.predict(X_train)
y_pred_test = dec_tree_reg.predict(X_test)
```

```
In [25]: train_accuracy = accuracy_score(y_train, y_pred_train.round())
test_accuracy = accuracy_score(y_test, y_pred_test.round())
```

```
In [26]: print("Training Accuracy:", train_accuracy)
print("Testing Accuracy:", test_accuracy)
```

```
Training Accuracy: 1.0
Testing Accuracy: 0.335625
```

← Accuracy

```
In [28]: # Save the model and LabelEncoders in a dictionary
data = {
    "model": dec_tree_reg,
    "gross_motor": gross_motor,
    "fine_motor": fine_motor,
    "communication": communication,
    "problem_solving": problem_solving,
    "emotional_dev": emotional_dev,
    "attention": attention,
    "overactivity": overactivity,
    "passivity": passivity,
    "planning": planning,
    "perception": perception,
    "perception_vf": perception_vf,
    "memory": memory,
    "spoken": spoken,
    "reading": reading,
    "social_skills": social_skills,
    "emotional_prob": emotional_prob,
    "growth": growth_r
}
```

```
In [29]: # Save the dictionary as a pickle file
with open('overAllModel.pkl', 'wb') as file:
    pickle.dump(data, file)
```

Selection of Decision Tree Regressor

- Highest accuracy score and lowest error compared to other algorithms.

```
In [12]: #trying the different machine learning models
from sklearn.linear_model import LinearRegression
linear_reg = LinearRegression()
linear_reg.fit(X, y.values)
```

```
Out[12]: <LinearRegression>
LinearRegression()
```

```
In [13]: y_pred = linear_reg.predict(X)
```

```
In [14]: from sklearn.metrics import mean_squared_error, mean_absolute_error
import numpy as np
error = np.sqrt(mean_squared_error(y, y_pred))
```

```
In [15]: error
```

```
Out[15]: 0.8156768147782453
```

```
In [21]: from sklearn.model_selection import GridSearchCV

max_depth = [None, 2,4,6,8,10,12]
parameters = {"max_depth": max_depth}

regressor = DecisionTreeRegressor(random_state=0)
gs = GridSearchCV(regressor, parameters, scoring='neg_mean_squared_error')
gs.fit(X, y.values)
```

```
Out[21]: <GridSearchCV>
GridSearchCV
  estimator: DecisionTreeRegressor
    > DecisionTreeRegressor
```

```
In [17]: regressor = gs.best_estimator_
regressor.fit(X, y.values)
y_pred = regressor.predict(X)
error = np.sqrt(mean_squared_error(y, y_pred))
print(error)
```

```
0.8156377061858167
```

```
In [19]: from sklearn.ensemble import RandomForestRegressor
random_forest_reg = RandomForestRegressor(random_state=0)
random_forest_reg.fit(X, y.values)
```

```
Out[19]: <RandomForestRegressor>
RandomForestRegressor(random_state=0)
```

```
In [20]: y_pred = random_forest_reg.predict(X)
error = np.sqrt(mean_squared_error(y, y_pred))
print(error)
```

```
0.4164121319965191
```

```
In [6]: from sklearn.tree import DecisionTreeRegressor
dec_tree_reg = DecisionTreeRegressor(random_state=0)
dec_tree_reg.fit(X, y.values)
```

```
Out[6]: <DecisionTreeRegressor>
DecisionTreeRegressor(random_state=0)
```

```
In [7]: y_pred = dec_tree_reg.predict(X)
```

```
In [8]: import numpy as np
from sklearn.metrics import mean_squared_error, mean_absolute_error
error = np.sqrt(mean_squared_error(y, y_pred))
print(error)
```

```
0.3083555172199778
```



Infant Growth Level Prediction

We need some information to predict the growth level of the child according to the development milestones



Age of the Infant

 0 5

Height (to the nearest integer in centimeters)

 - +

Weight (to the nearest integer in kilograms)

 - +

Gross Motor Development Status

 ▼

How Organized And Planned?

 ▼

Fine Motor Development Status

 ▼

How Good At Perception Of Directions?

 ▼

How Good At Solving Problems?

 ▼

How Good At Perception Of Visual Forms And Figures?

 ▼

How Good At Communication?

 ▼

How Good Is The Memory?

 ▼

Emotional Development Status

 ▼

How Good At Reading Skills?

 ▼

Attention And Concentration

 ▼

How About Speaking Skills?

 ▼

Overactivity and Impulsivity

 ▼

Social Skills Status

 ▼

Passivity Or Inactivity

 ▼

Are There Any Emotional Problems?

 ▼

Predict the Growth

Growth level can be predicted as : Under Growth

```
predict_page.py - Implementation - Visual Studio Code
File Edit Selection View Go Run Terminal Help
predict_page.py X explore_page.py two_months.py four_months.py
EXPLORER
  IMPLEMENTATION
    > __pycache__
    > .ipynb_checkpoints
    > .vscode
    1.jpg
    2.jpg
    3.jpg
    app.py
    baby2.jpg
    baby3.jpg
    Child_Growth.csv
    Child_Growth.xlsx
    DecisionTreeRegressor.ipynb
    emotional.webp
    explore_page.py
    four_months.py
    newModel.pkl
    overAllModel.pkl
    OverallPredictionModel.ipynb
    predict_page.py
    saved_steps.pkl
    Test1.ipynb
    two_months.py
    Untitled3.ipynb
    Untitled4.ipynb
    Untitled5.ipynb
    Untitled6.ipynb
    WebAppFromScratch.ipynb
predict_page.py > show_predict_page
1 import streamlit as st
2 import pickle
3 import numpy as np
4 from sklearn.preprocessing import LabelEncoder
5
6 # Load the trained model and data
7 def load_model():
8     with open('overAllModel.pkl', 'rb') as file:
9         data = pickle.load(file)
10        growth_r = data["growth"] # Load the "growth" label as "growth_r"
11        data["growth_r"] = growth_r # Add the loaded label to the dictionary
12        del data["growth"] # Remove the original "growth" key
13    return data
14
15 # Show images on the page
16 def show_images():
17     # Set page layout
18     col1, col2, col3 = st.columns(3)
19
20     # Column 1 - First image
21     with col1:
22         st.image("1.jpg", use_column_width=True)
23     # Column 2 - Second image
24     with col2:
25         st.image("3.jpg", use_column_width=True)
26     # Column 3 - Third image
27     with col3:
28         st.image("2.jpg", use_column_width=True)
29
30 # Show the growth predictor page
31 def show_predict_page():
32     data = load_model()
33
```

The libraries required for the code are imported in this part, including **Streamlit** for building the web application, **Pickle** for loading the trained model, **NumPy** for performing mathematical operations, and **LabelEncoder** for encoding categorical variables.

The trained model is loaded using the `load_model()` method from the "overAllModel.pkl" file. The data dictionary is read from the file, the original "growth" key is deleted, the "growth" label is assigned to `growth_r`, and the "growth_r" label is added to the dictionary. It then gives back the modified data dictionary

predict_page.py > show_predict_page

```
29
30 # Show the growth predictor page
31 def show_predict_page():
32     data = load_model()
33
34     regressor = data["model"]
35     gross_motor_encoder = data["gross_motor"]
36     fine_motor_encoder = data["fine_motor"]
37     problem_solving_encoder = data["problem_solving"]
38     communication_encoder = data["communication"]
39     emotional_dev_encoder = data["emotional_dev"]
40     attention_encoder = data["attention"]
41     overactivity_encoder = data["overactivity"]
42     passivity_encoder = data["passivity"]
43     planning_encoder = data["planning"]
44     perception_encoder = data["perception"]
45     perception_vf_encoder = data["perception_vf"]
46     memory_encoder = data["memory"]
47     spoken_encoder = data["spoken"]
48     reading_encoder = data["reading"]
49     social_skills_encoder = data["social_skills"]
50     emotional_prob_encoder = data["emotional_prob"]
51
52
53 # Create a mapping dictionary for numerical labels to string labels
54 label_mapping = {
55     2: "Under Growth",
56     0: "Normal Growth",
57     1: "Over Growth"
58 }
59
```

predict_page.py > show_predict_page

```
60 # widgets-title
61 st.title("Infant Growth Level Prediction")
62 # widgets-text
63 # h3 tag - ###
64 st.write("""### We need some information to predict the growth level of the child according to the development milestones""")
65
66 # Call the function
67 show_images()
68
69 #choices for the dropdowns
70 gross_motor = (
71     "poor",
72     "fair",
73     "good",
74     "excellent"
75 )
76
77 fine_motor = (
78     "poor",
79     "fair",
80     "good",
81     "excellent"
82 )
83
84 problem_solving = (
85     "poor",
86     "fair",
87     "good",
88     "excellent"
89 )
90
```

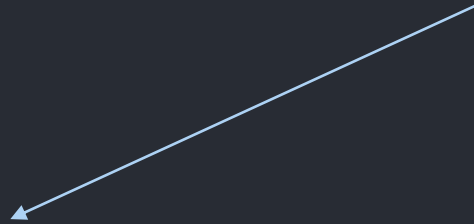
The primary function that renders the growth predictor page is `show_predict_page()`. It uses the `load_model()` method to initially load the model and any other necessary data. Then, it divides up the supplied data into distinct variables for further usage.

```

predict_page.py > show_predict_page
162     )
163
164     emotional_prob = (
165         "yes",
166         "no"
167     )
168
169
170     # creates a slider for age selection
171     age = st.slider("Age of the Infant", 0, 5, 0)
172
173     # creates number input fields for height and weight
174     height = st.number_input("Height (to the nearest integer in centimeters)", min_value=0)
175     weight = st.number_input("Weight (to the nearest integer in kilograms)", min_value=0)
176
177     # creates select boxes
178     col1, col2 = st.columns(2)
179
180     with col1:
181         gross_motor_status = st.selectbox("Gross Motor Development Status", gross_motor)
182
183         fine_motor_status = st.selectbox("Fine Motor Development Status", fine_motor)
184
185         problem_solving_status = st.selectbox("How Good At Solving Problems?", problem_solving)
186
187         communication_status = st.selectbox("How Good At Communication?", communication)
188
189         emotional_development_status = st.selectbox("Emotional Development Status", emotional_dev)
190
191         attention_status = st.selectbox("Attention And Concentration", attention)
192
193         overactivity_status = st.selectbox("Overactivity and Impulsivity", overactivity)
194

```

The slider, select boxes and other components to take the user input, are indicated here.




```

213
214 # Button to trigger prediction
215 ok = st.button("Predict the Growth")
216 if ok:
217     # Prepare input data for prediction
218     X = np.array([[weight, height, age, gross_motor_status, fine_motor_status, problem_solving_status, communication_status, emotional_development_status, attention_s
219
220     X[:, 3] = gross_motor_encoder.transform(X[:, 3])
221     X[:, 4] = fine_motor_encoder.transform(X[:, 4])
222     X[:, 5] = problem_solving_encoder.transform(X[:, 5])
223     X[:, 6] = communication_encoder.transform(X[:, 6])
224     X[:, 7] = emotional_dev_encoder.transform(X[:, 7])
225     X[:, 8] = attention_encoder.transform(X[:, 8])
226     X[:, 9] = overactivity_encoder.transform(X[:, 9])
227     X[:, 10] = passivity_encoder.transform(X[:, 10])
228     X[:, 11] = planning_encoder.transform(X[:, 11])
229     X[:, 12] = perception_encoder.transform(X[:, 12])
230     X[:, 13] = perception_vf_encoder.transform(X[:, 13])
231     X[:, 14] = memory_encoder.transform(X[:, 14])
232     X[:, 15] = reading_encoder.transform(X[:, 15])
233     X[:, 16] = spoken_encoder.transform(X[:, 16])
234     X[:, 17] = social_skills_encoder.transform(X[:, 17])
235     X[:, 18] = emotional_prob_encoder.transform(X[:, 18])
236
237
238 growth = regressor.predict(X)
239 y_pred_string = [label_mapping[label] for label in growth] # Map numerical labels to string labels
240 st.subheader(f"Growth level can be predicted as : " + ", ".join(y_pred_string))
241

```

Here creates a 2D numpy array X containing the input data for prediction. The values are taken from various variables representing the selected options and input values.

The next set of lines from `X[:, 3] = gross_motor_encoder.transform(X[:, 3])` to `X[:, 18] = emotional_prob_encoder.transform(X[:, 18])` encode the categorical features in the input data to numerical values using the respective encoders. This is necessary for the machine learning model to make predictions. `growth = regressor.predict(X)`: This line uses the trained regressor model to make predictions on the prepared input data X. The predicted growth levels are stored in the growth variable.

Work In Progress

Your Baby At Four Months

Social and Emotional Milestones

- ☒ Smiles On Their Own To Get The Attention
- ☐ Chuckles (Not Yet A Full Laugh) When Trying To Make Them Laugh
- ☐ Looks At The Parent, Moves Or Make Sounds To Keep The Attention

---Social and Emotional Development: Poor---

Language And Communication Milestones

- ☐ Makes Sounds Like 'oooh', 'aaahh' (cooing)
- ☐ Makes Sounds Back When You Talk To Them
- ☒ Turns Head Towards The Sound Of The Voice

---Language And Communication Development: Poor---

Cognitive Milestones (Learning, Thinking, Problem Solving)

- ☒ If Hungry, Opens The Mouth When Sees A Bottle Or The Breast
- ☒ Look At The Hands With Interest

---Cognitive Development: Excellent---

Movement / Physical Development Milestones

- ☐ Holds Head Up Steady Without Support When Holding
- ☐ Holds A Toy When Put In The Hands
- ☐ Uses The Arm To Swing At Toys
- ☐ Brings Hands To Mouth
- ☐ Pushes Up Onto Elbows / Forearms When On Tummy

The Development Milestones of a child

There are four different development milestones in a child's life cycle.

- Physical Development - Gross motor and Fine motor skills
- Cognitive Development - Ability to think, reason and solve problems
- Language Development - Communicate using words and gestures, understand language and expressions
- Social and Emotional Development - Express emotions, interactions and attachments



IT20156374| Fernando A.P

Infant Sickness Identification through Video Processing



Software Engineering

Component Discussion

- **Introduction**

- The focus of this research is to develop a system that utilizes video processing techniques to identify abnormal behaviors in infants, aiming to assist parents in recognizing potential illnesses.
- The system enables parents to record videos of their infants and provides guidance on capturing specific behaviors for analysis.
- By employing image processing and video processing techniques, the system analyzes the recorded videos to identify signs of sickness in infants.
- The primary objective of this system is to offer valuable support to parents with babies up to five years old.

- **System Workflow**

- Parent records a video of their infant, following the system's instructions.
- The system processes the recorded video using advanced image and video processing techniques.
- The system detects abnormal behaviors or signs of sickness in the video frames.
- If any abnormal behavior is identified, the system prompts a command or displays a notification message to the parent.

Component Discussion

- **Benefits for Parents**
 - Enhanced Monitoring: The system allows parents to actively monitor their infants for potential illnesses.
 - Early Detection: By capturing and analyzing abnormal behaviors, the system aids in early detection of sicknesses.
 - Prompt Notifications: Parents receive immediate alerts or commands from the system, providing timely guidance and support.
 - Peace of Mind: This system empowers parents with a valuable tool to ensure the well-being of their infants.
- **Impact and Future Scope**
 - By developing a reliable system for sickness identification in infants, this research aims to contribute to the field of child healthcare and parental support.
 - Future enhancements may involve integrating artificial intelligence algorithms to improve the accuracy and efficiency of sickness detection.
 - The system could be expanded to include a wider range of sicknesses and developmental abnormalities, catering to a broader age group of children.

Information Gathering

Interviews and discussions about the overall system and individual component

- ✓ Dr.Supun Peris - Ragama Hospital

Resources and study materials to enhance the components

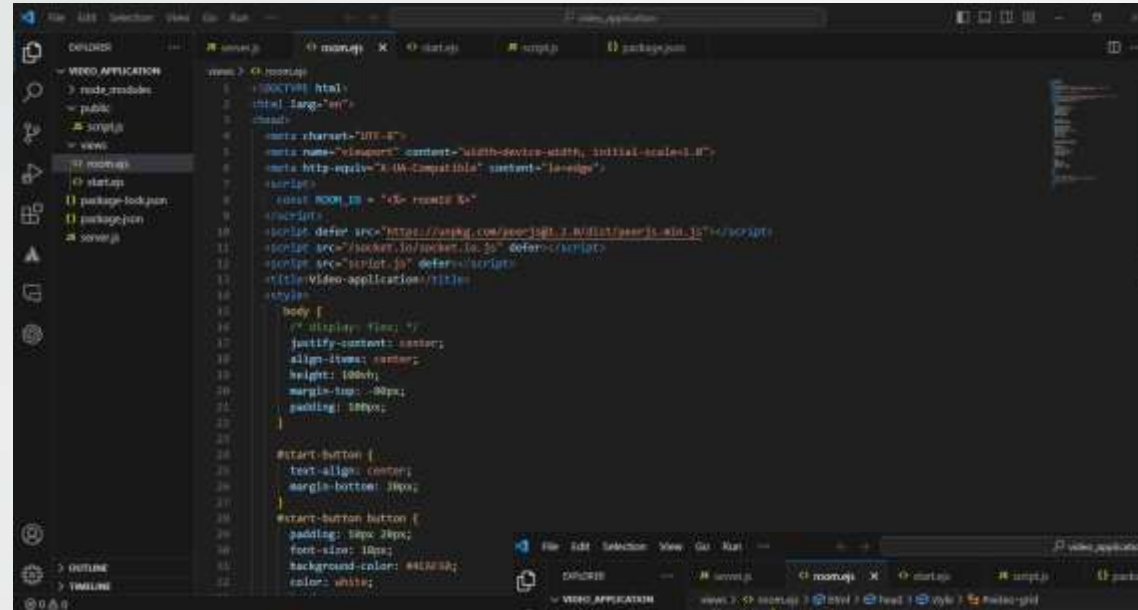
- ✓ Existing Research papers
- ✓ Articles
- ✓ Video tutorials for technologies

Component Overview Diagram

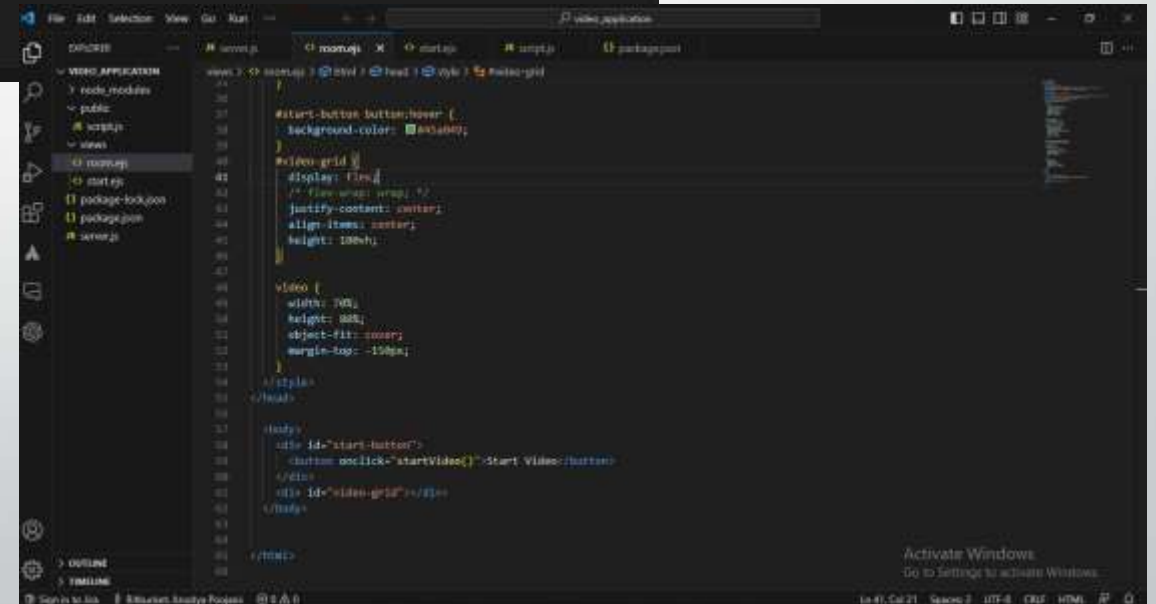
Completion of the project

Screenshot of the implementation

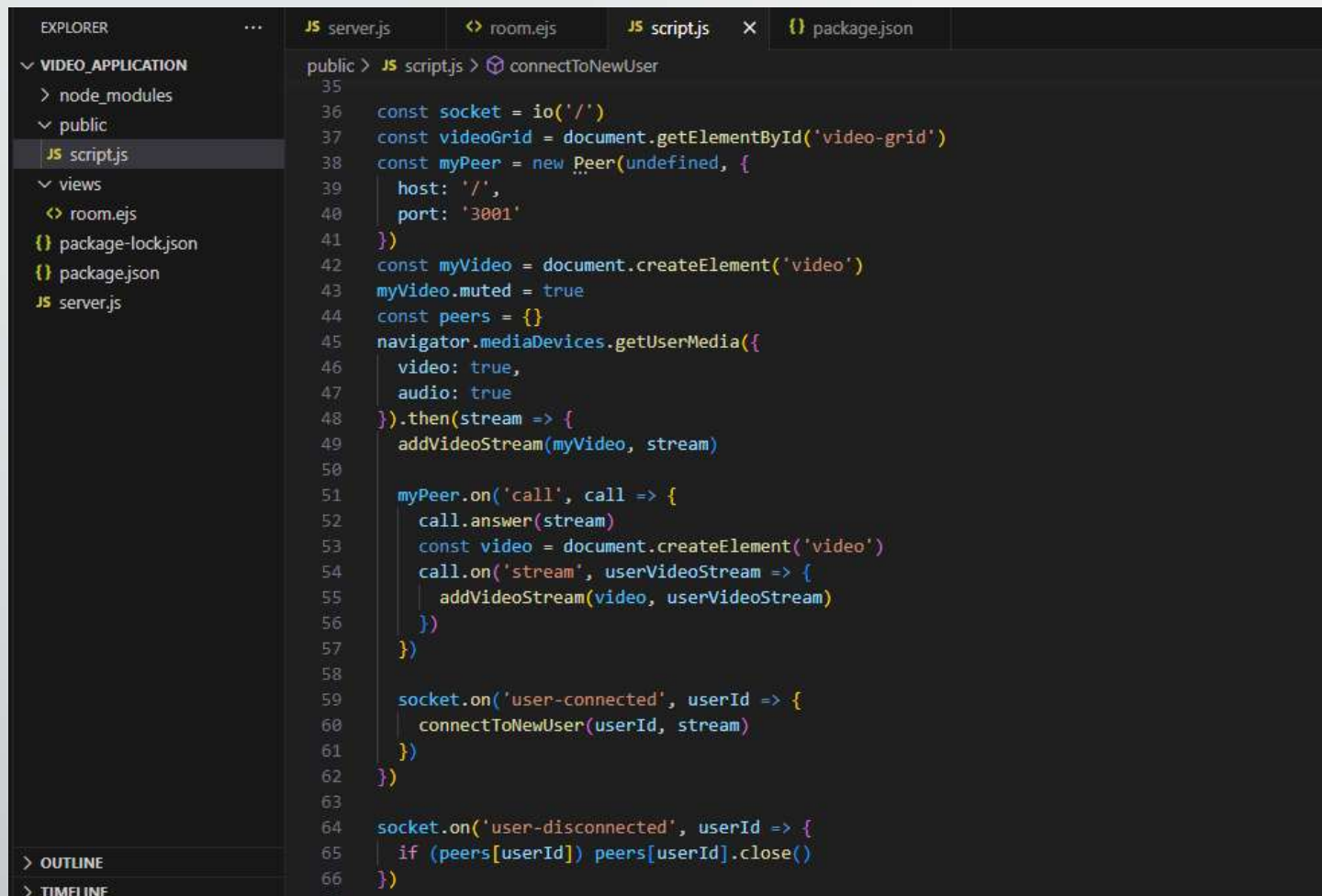
Room.ejs – this is one of frontend implementation code segment. This will load the video to the browser.



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <meta http-equiv="X-UA-Compatible" content="ie=edge">
7   <script>
8     const ROOM_ID = "00- room1 1a"
9   </script>
10   <script defer src="https://unpkg.com/peerjs@1.3.0/dist/peerjs.min.js"></script>
11   <script src="/socket.io/socket.io.js" defer=<script>
12   <script src="script.js" defer=<script>
13   <title>Video-application</title>
14 </head>
15 <body>
16   <div display="flex">
17     justify-content: center;
18     align-items: center;
19     height: 100vh;
20     margin-top: -80px;
21     padding: 10px;
22   </div>
23   <div id="start-button">
24     text-align: center;
25     margin-bottom: 10px;
26   </div>
27   <div id="start-button">
28     padding: 10px 20px;
29     font-size: 18px;
30     background-color: #4CAF50;
31     color: white;
32   </div>
```



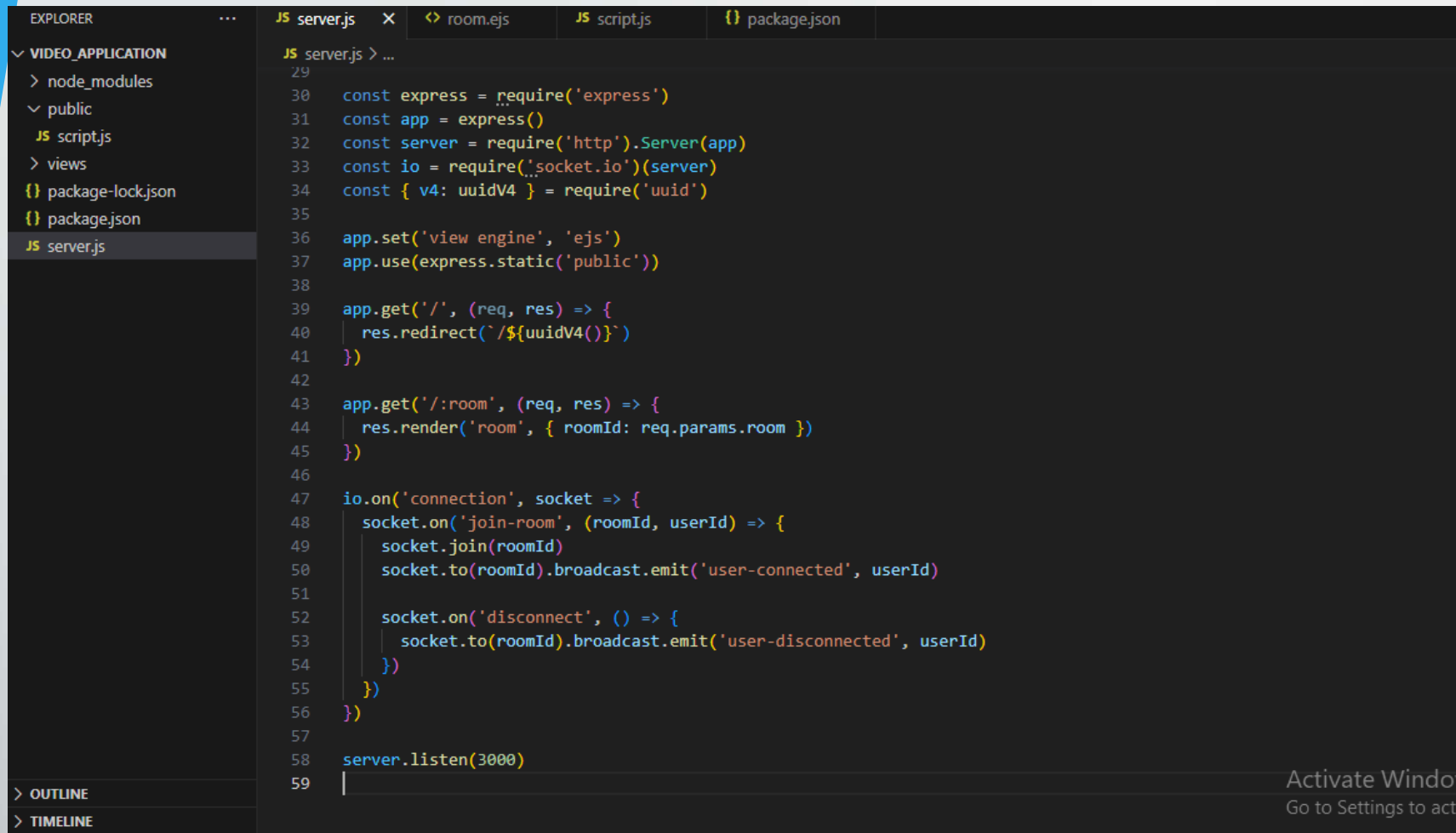
```
1 <div id="start-button">
2   background-color: #4CAF50;
3 </div>
4 <div id="video-grid">
5   display: flex;
6   flex-wrap: wrap;
7   justify-content: center;
8   align-items: center;
9   height: 100vh;
10 </div>
11 <div id="video">
12   width: 100%;
13   height: 80%;
14   object-fit: cover;
15   margin-top: -150px;
16 </div>
17 </body>
18 </html>
```



```
EXPLORER
  VIDEO_APPLICATION
    node_modules
    public
      JS script.js
    views
      room.ejs
      package-lock.json
      package.json
      server.js

public > JS script.js > connectToNewUser
35
36 const socket = io('/')
37 const videoGrid = document.getElementById('video-grid')
38 const myPeer = new Peer(undefined, {
39   host: '/',
40   port: '3001'
41 })
42 const myVideo = document.createElement('video')
43 myVideo.muted = true
44 const peers = {}
45 navigator.mediaDevices.getUserMedia({
46   video: true,
47   audio: true
48 }).then(stream => {
49   addVideoStream(myVideo, stream)
50
51   myPeer.on('call', call => {
52     call.answer(stream)
53     const video = document.createElement('video')
54     call.on('stream', userVideoStream => {
55       addVideoStream(video, userVideoStream)
56     })
57   })
58
59   socket.on('user-connected', userId => {
60     connectToNewUser(userId, stream)
61   })
62 })
63
64 socket.on('user-disconnected', userId => {
65   if (peers[userId]) peers[userId].close()
66 })
```

Script.js – This code sets up a video conferencing application using WebRTC technology. It establishes a connection between clients using socket.io and Peer.js libraries.



```
EXPLORER  ...  JS server.js  X  <> room.ejs  JS script.js  {} package.json

VIDEO_APPLICATION
├── node_modules
├── public
├── JS script.js
├── views
├── {} package-lock.json
├── {} package.json
└── JS server.js

JS server.js > ...
29
30   const express = require('express')
31   const app = express()
32   const server = require('http').Server(app)
33   const io = require('socket.io')(server)
34   const { v4: uuidV4 } = require('uuid')
35
36   app.set('view engine', 'ejs')
37   app.use(express.static('public'))
38
39   app.get('/', (req, res) => {
40     res.redirect(`/${uuidV4()}`)
41   })
42
43   app.get('/:room', (req, res) => {
44     res.render('room', { roomId: req.params.room })
45   })
46
47   io.on('connection', socket => {
48     socket.on('join-room', (roomId, userId) => {
49       socket.join(roomId)
50       socket.to(roomId).broadcast.emit('user-connected', userId)
51
52       socket.on('disconnect', () => {
53         socket.to(roomId).broadcast.emit('user-disconnected', userId)
54       })
55     })
56   })
57
58   server.listen(3000)
59
```

Activate Window
Go to Settings to act

Server.js - This server-side code sets up a signaling server using Node.js, Express, and Socket.IO to facilitate communication between clients in the video application.

- Technologies and libraries

```
1  {
2    "name": "video_application",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "devStart": "nodemon server.js"
8    },
9    "keywords": [],
10   "author": "",
11   "license": "ISC",
12   "dependencies": {
13     "ejs": "^3.1.3",
14     "express": "^4.17.1",
15     "socket.io": "^2.3.0",
16     "uuid": "^8.1.0"
17   },
18   "devDependencies": {
19     "nodemon": "^2.0.4"
20   }
21 }
```

Node.js: A JavaScript runtime environment used to execute server-side JavaScript code.

Express: A web application framework for Node.js that simplifies the development of web applications and APIs.

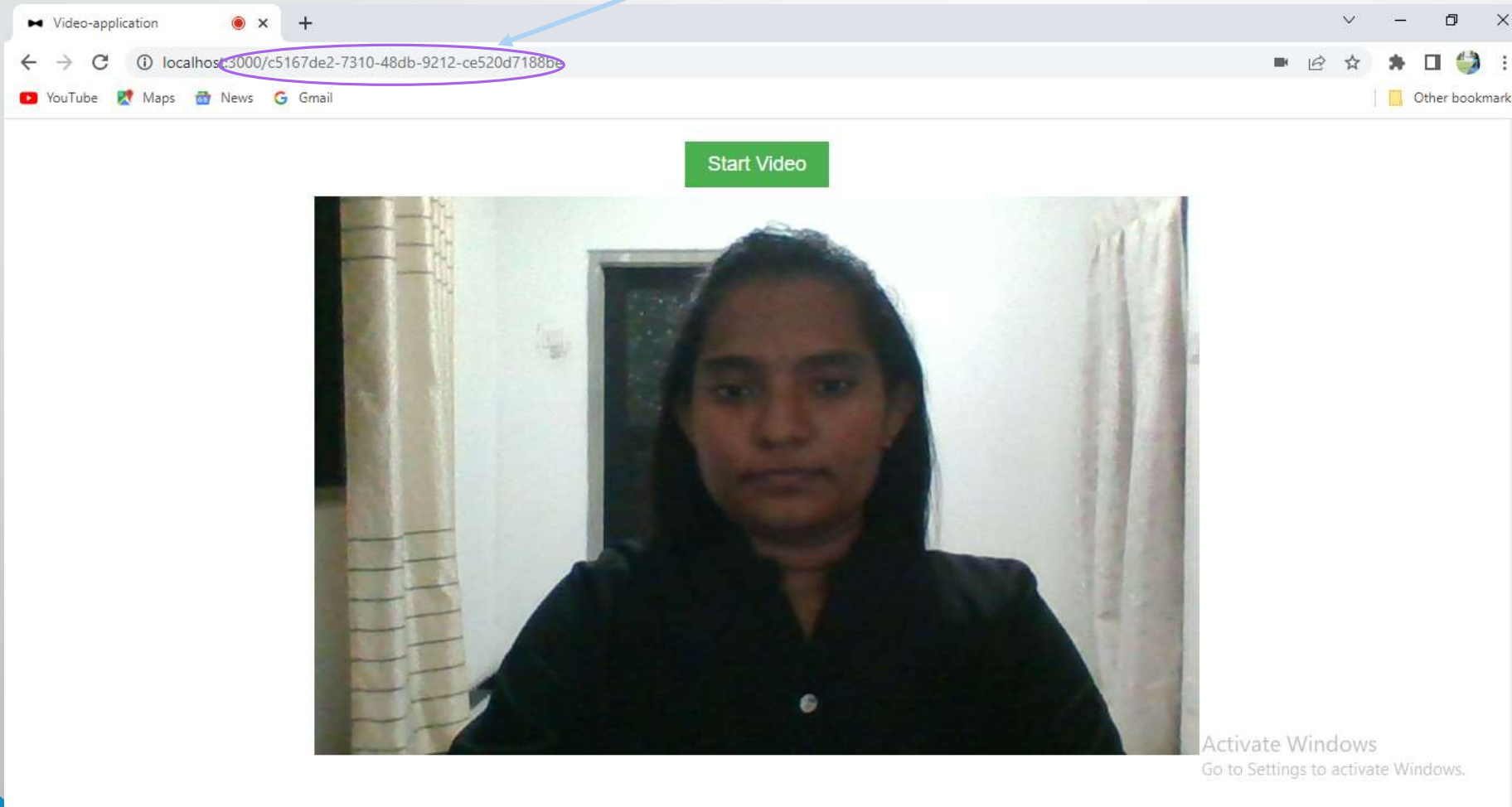
Socket.IO: A JavaScript library that enables real-time, bidirectional communication between web clients and servers using WebSockets.

EJS: Embedded JavaScript, a templating engine that generates HTML markup with embedded JavaScript code.

UUID

Output of the implementation

Generating video with the id for user



Supportive Information


- Commercialization
- Budget

Commercialization

- The proposed system aims to streamline the registration process for newborns and their parents by utilizing decentralized technology like blockchain. By doing so, the system offers increased security, transparency, and accessibility.
- To meet the functional requirements, the system would need to enable hospitals and healthcare providers to register newborns and parents' details, provide a user-friendly interface for parents to register their information, and ensure that the data is stored accurately on the blockchain.
- Nonfunctional requirements would include performance, scalability, and security, such as ensuring that the system can handle a large volume of registrations and protect data from unauthorized access.
- To commercialize the system, it could be marketed to hospitals and healthcare providers as well as parents who want a secure and accessible way to register their child's details.
- The system could be monetized through subscription or usage fees and could also be integrated with other healthcare systems or applications to provide additional value to users.

Budget

<u>Task</u>	<u>Cost (Rs.)</u>
• Domain Name	5000.00
• Hosting	9000.00
• Backups	5000.00
• Website Strategy	2000.00
• Testing	2000.00
• Other	3000.00
• Maintenance	2000.00
• Marketing	10000.00
Total Cost	38000.00



Thank You!

Team TMP 23-103