Amity Journal of Computational Sciences
ISSN 2456-6616 (Online)
Volume 6 Issue 2, 2022

AI Therapist Impelementation

Kinjalk Tripathi
Batchelor of Technoogy,CSE
Amity University Uttar Pradesh , India

Akash Kumar
Batchelor of Technoogy,CSE
Amity University Uttar Pradesh, India

Abhishek Singh
Batchelor of Technoogy, CSE
Amity University Uttar Pradesh, India

***Abstract*—The following research paper concludes that there's a gap left by traditional therapy which can be complimented by an AI Therapist. To create such an app, we use the following tech stack-- Nuxt.js for frontend, Tauri for backend, Git for source version control & FastAPI for testing purposes. Via the use of above technology, we uncover how to create an AI Therapist and deploy the application.**

Keywords—Artificial Intelligence, Virtual Therapist, AI Therapist

I. INTRODUCTION

In today's age, communicating with people far away is quite convenient. However, as the world gets more on-board with internet and its newfound culture, a knockoff-effect is widely observed where most people still feel isolated and left out. The need for mental health support has never been as prevalent as it is now. There are a ton of social issues that can be discussed but they all ultimately lead to victims needing support and guidance. As we know the world is not perfect hence the proper support and guidance is not readily available to many people who need it. Therapy is a way most people can take their emotional load-off and reflect on life more profoundly so that they could increase quality of life however there are still challenges with therapy that limits its popularity and keeps people away. These challenges are -- Accessibility, Anonymity, Affordability, Immediate Support & Customization. A Virtual AI Therapist seeks to eliminate these challenges however it is important to note that Virtual AI Therapist should not replace human therapists entirely. They can complement traditional therapy but may not be able to replicate the same level of human connection, empathy, and intuition that a human therapist can provide.

II. FRONTEND OF APPLICATION

A. *Vue.js*

Vue.js is a JavaScript framework that combines the features of other popular frameworks (such as React & Angular) and presents them with an ease of syntactic sugar. Its composition API will be the main language of our frontend.

B. *Nuxt.js: A vue.js framework*

Nuxt.js is a micro-framework for Vue.js which is a framework for JavaScript, meaning Nuxt is built on top of Vue.js to ease development further. Beyond that, Nuxt.js offer a whole lot of things to the table including its own backend server however for this research, we're only going to use Nuxt.js for frontend.

Nuxt 3 has been incorporated into the research project for serving the frontend. It's very convenient to add custom fonts, icons and animation via it. The frontend (Client-side) also stores the log of conversation and passes it onto the backend for further operations. The storing of conversation log is deliberate so that once the session ends, the data is flushed; hence keeping the anonymity of the user intact. Additionally, there is also no fear of data leakage of stuff simply because it's not really being stored on server or database.

The frontend must be minimalistic which is why to sort that out, we'll first write all the basic necessary features needed. The list of necessary items to be included are: -

- Light/Dark Mode
- Copyright information
- Close app button

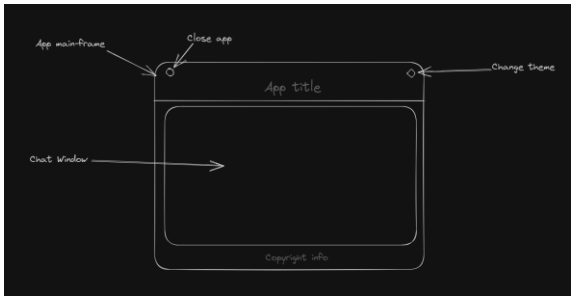

Fig. 1. Protype of the frontend for our application

III. OPEN AI

Open AI is a research and deployment company for Artificial Intelligence [1]. Through research, it creates AI models that are trained to generate text, images, and audios as per prompts they are passed on. Through deployment, they rent out their AI models based on *tokens* system. Their deployment method is to create API (Application Programming Interface) hosted on a server endpoint for each model and the type of interaction user desires.

For this research paper, the images and audios are irrelevant. Text however can be generated for two purposes – Completion & Chat. Completion refers to single-turn conversation. For example, a FAQ chatbot or search-engine chatbot. Chat, on the other hand, refers to multi-turn conversation where the context is preserved. For example, a virtual therapist or personal assistant.

A. *Basic Idea*

We'll send a POST request to the OpenAI's endpoint which serves the model we wish to interact with. The POST request will contain a payload; the structure of which is important to study. Below we've written about both Request

Body and Response Body. That will explain how we are going to send and collect information.

B. Request Body

- **model**: For conveying which model to use for generating text. We use 'gpt-3.5-turbo' for this research project as it is meant to carry on chats.
- **messages**: This is an array of dictionary which has primarily two properties – Role & Content. Role can be system, user, or assistant. System role is used for defining what personality the AI model must adopt. User role is used for conveying user message and assistant role is used for preserving context.
- **temperature**: What sampling temperature to use, between 0 and 2. Higher values like 0.8 will make the output more random, while lower values like 0.2 will make it more focused and deterministic.
- **stream**: To receive a data stream for visual effect.
- **max_tokens**: maximum number of tokens to generate in chat completion [2].

C. Response Body

As show in Fig. 2. The response body returns a structure which is different from request. It's, again, important to understand this. The main content we wish to extract is found in '*Choices*' array. Choices is an array of objects where each object has a property called '*message*'. That is where user's query is stored. The reason the object is present in an array is because a response can have multiple answers to user's query. Besides *choices*, we also have the ability to gather more information about the exchange such as how many tokens were used or what is the time of response. In view of further development, these properties can be quite handy.


```
{
  "id": "chatcmpl-123",
  "object": "chat.completion",
  "created": 1677652288,
  "choices": [{
    "index": 0,
    "message": {
      "role": "assistant",
      "content": "\n\nHello there, how may I assist
    },
    "finish_reason": "stop"
  }],
  "usage": {
    "prompt_tokens": 9,
    "completion_tokens": 12,
    "total_tokens": 21
  }
}
```


Fig. 2. Response body from OpenAI's endpoint

D. Some More Insights

- All data are exchanged in JSON format.
- Json is a schema format and is applied to the sourcing information (aka *request*) and the *response.*
- The JSON is standard for implement a schema or model. These schemas are imposed so that the information can be read and processed in an effective way. Similarly, the response is also a JSON so that we can always expect the structure as it only serves to ease the developer experience.
- OpenAI offers $5 of free tier which is more than enough for testing and tweaking the AI to our liking.
- In request body, in the very first prompt; we specify the role we want the AI playing. In this case, it's that of Mental Therapist.

IV. Backend

Since we're constructing a Tauri application so we can only use *Rust* as our backend language. This is why even though Nuxt.js offers its own Node.js server, we cannot use it to call any APIs. Nuxt.js will try to render the frontend with its own backend server by default. To prevent this behaviour, we'll configure the *nuxt.config.ts* file and set '***ssr***' to **false**.

A. Rust: Programming language

Rust has ranked to be the most loved programming language on Stack Overflow survey consecutively for 4 years since 2018. Rust is blazingly fast and memory-efficient: with no runtime or garbage collector, it can power performance-critical services, run on embedded devices, and easily integrate with other languages. Rust's rich type system and ownership model guarantee memory-safety and thread-safety — enabling you to eliminate many classes of bugs at compile-time. Hundreds of companies around the world are using Rust in production today for fast, low-resource, cross-platform solutions. Software we know, like Firefox, Dropbox, and Cloudflare, use Rust [3]. Rust is excellent for writing performant webapps and network related services henceforth we use Rust for backend. To use Rust as backend, we add following packages to its package manager file, which is named as *Cargo.toml* :-

a) **Hyper** - A http library because as we learned, OpenAI uses rest endpoint to expose their AI models.

b) **Serde** - For ¬serialization and deserialization of Json as both request and response have Json structure.

c) **Tokio** – Provides async function support. We need it because response will be returned as a promise by the server.

Now, we add a series of code presented in the Fig. 3, Fig. 4, Fig. 5, Fig. 6 & Fig. 7.

```rust
/* RESPONSE */
#[derive(Deserialize, Debug)]
#[allow(dead_code)]
2 implementations
struct OIChoices { //this is a subset of openai-response.
    text: String,
    index: u8,
    longprobs: Option<u8>,
    finish_reason: String,
}

#[allow(dead_code)]
#[derive(Deserialize, Debug)]
2 implementations
struct OIResponse {
    id: Option<String>,
    object: Option<String>,
    created: Option<u64>,
    model: Option<String>,
    choices: Vec<OIChoices>,
}

/* REQUEST */
#[derive(Serialize, Debug)]
2 implementations
struct OIRequest {
    prompt: String,
    max_tokens: u32,
}
```

Fig. 3. We create Structs for response and Request

```rust
#[tokio::main]
async fn completion(
    token: &str,
    chat_history: Vec<Convo>,
) -> Result<std::string::String, std::string::String> {
    let https: HttpsConnector<HttpConnector> = HttpsConnector::new();
    let client: Client<HttpsConnector<HttpConnector>> = Client::builder().build(https);
    let uri: &str = "https://api.openai.com/v1/chat/completions";
```

Fig. 4. Creating HTTP client for communicating with APIs

```rust
/* auth token */
let oai_token: String =
    env::var("OAI_TOKEN").expect("OAI_TOKEN not found in environment variables");
```

Fig. 5. Adding API-key securely via enviornment variables

```rust
/* creating prompt */
let oi_request: OIRequest = OIRequest {
    prompt: format!("{} {}", preamble, input),
    max_tokens: 64,
};

/* building request */
let body: Body = Body::from(serde_json::to_vec(&oi_request)?);
let req: Request<Body> = Request::post(uri).Builder
    .header(key: header::CONTENT_TYPE, value: "application/json").Builder
    .header(key: "Authorization", value: &auth_header_val).Builder
    .body(body).Result<Request<Body>, Error>
    .expect(msg: "server error");

/* now we're gonna use our hyper client to send the request to the endpoint and wait for a response */
let res: Response<Body> = client.request(req).await?;
let body: impl Buf = hyper::body::aggregate(body: res).await?;
```

Fig. 6. We create a request and hit the desired endpoint

```rust
    /* deserialize response */
    let json: OIResponse = serde_json::from_reader(body.reader()).expect("invalid json");

    println!("response: {:?}", json);

    let answer: String = json.choices[0].message.content.to_string();

    Ok(answer)
} fn completion

#[tauri::command]
fn get_response(chats: Vec<Convo>, token: &str) -> String {
    /* extracting chats to sendable prompts */
    let response: Result<String, String> = completion(token, chats);
    response.unwrap()
}
```

Fig. 7. Lastly, we deserialize the response and return it.

This will allow us to fetch AI generated responses.

B. Tauri: A Rust Toolkit

a) Our Rust webapp will also need a GUI (Graphical User Interface). There are many crates (library for Rust) in Rust to achieve that however Tauri stands out as the most promising framework. Tauri helps developers make applications for the major desktop platforms - using virtually any frontend framework in existence. The core is built with Rust, and the CLI leverages Node.js making Tauri a genuinely polyglot approach to creating and maintaining great apps [4]. This is why Tauri has been chosen for this research project.

b) Via 'yarn create <appname>', we scaffold a project. After that we add our Nuxt 3 framework in it. Lastly, via 'yarn add @tauri-apps/api', we add a JavaScript library that provides access to the core functionality of windows, the filesystem, and more through convenient JavaScript abstractions [5].

c) In, 'src-tauri', our Rust backend will live. In, 'src', our Nuxt frontend will live.

d) Now, to communicate between the backend and frontend, we'll add a '#[tauri::command]' before our function in the backend. This can be viewed in the previous section.

e) For the frontend, we import 'invoke' function from the respective library. After that we can establish communication like this:

```rust
/* rust */
const response = await invoke("get_response", { chats: conversation.value, token: config.public.
token_id});
if (response) {
    waiting.value = false;
    conversation.value.push({
        role: "assistant",
        content: response
    })
} else {
    waiting.value = false;
    conversation.value.push({
        role: "assistant",
        content: "Sorry, there seems to be an issue with the server. Please try again later."
    })
}
```

Fig. 8. Communicating between Backend and Frontend

f) In this communication, the frontend collects the message user send, stores it into an object and sends that object to the backend. Backend extracts the message and parses it into the request to await the response. The response is then transferred to frontend via function's return call. The frontend fetches the AI response and stores it into the same object where user message is kept. By iterating over the array of objects, the frontend displays all the messages in the chat window. The cycle repeats for every message user sents.

C. Testing with FastAPI

The OpenAI offers a free tier of usage for API calls. However, during testing, we cannot afford to call an API every time we make some changes and wish to test them out. Since we also cannot use Nuxt's Node server because Tauri only supports Rust as backend, so we'll have to create our own API endpoint to test out the frontend.

FastAPI is THE choice for creating APIs. Its performance is on par with Node.js and also creating asynchronous functions has never been easier.

a) We'll firstly setup a virtual environment by the name of '*server*' by running the following command: **py -3 -m venv server**.

b) Then we'll install FastAPI through pip package manager inside the virtual environment. Next, we'll create a main.py and write server configuration in it.

c) It has to be noted that FastAPI has 'auto-import' feature built-in; meaning that when you're using any module, it will automatically get imported to your file without you having to worry about it.

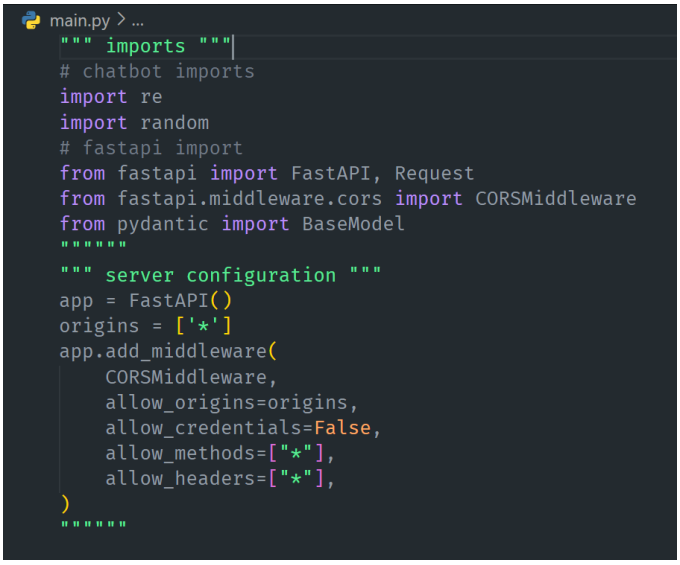

Fig. 9. Imports and Server Configuration in FastAPI

d) Now we move on to creating a chatbot that has some predefined responses so to test out our frontend. For this, we'll create a function that will be called once our endpoint gets hit. First off, we create a dictionary called responses which has user-input as its key and predefined-response as its value. Once that is set up, we'll create a function called **match_response**.

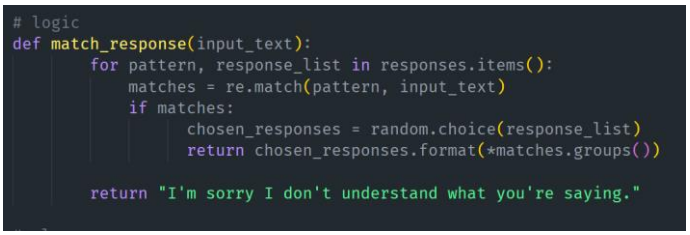

Fig. 10. Creating Server function in FastAPI

e) Now, we don't want user hitting our endpoint without any proper request model so we'll define a class that is basically a Pydantic model used in FastAPI to ensure correct request is accepted & rest is rejected. So, we'll create just that and with it; our endpoint as well.

Fig. 11. Exposing our chatbot function to endpoint

f) Now we have successfully built our FastAPI endpoint. We will start the server by using the command: **uvicorn main:app –reload**. This will get our FastAPI server up and running.

g) We will now configure our client side for testing, basically making Nuxt communicate with FastAPI. In order to do that, we'll write some JavaScript code in our frontend.

Fig. 12. Communicating JavaScript in Frontend Nuxt with FastAPI backend.

V. Building the App

The thing with Nuxt is that it doesn't generate a **dist** folder with all the static files in it; unlike how other frameworks do. It instead creates a .**output** folder where it holds the static assets & build files. The reason simply being that Nuxt supports server-side rendering so rather than just generating static files, by default it creates an entry point that launches a "ready-to-run Node server". The server is then used to render the webpage therefore static files aren't needed but static assets are. Now, this creates a conflict because Tauri needs the static files for production so during the development, your app will run just fine. However, when you'll run the build command for your app, you'll likely get an error at the runtime. To solve this we'll configure two files, namely – ***nuxt.config.ts*** & ***tauri.conf.json***.

A. Configuring Nuxt

While working with Tauri, since we're already using Rust for the backend therefore we'll first make sure to disable the server-side rendering for Nuxt. To do so, we need to specify in nuxt.config.ts that we don't wish to use server-side rendering for this research project.

```
export default defineNuxtConfig({
    ssr: false
})
```

Fig. 13. Configuring nuxt.config.ts: ssr false

Then, we simply tell Nuxt to prerender our site (meaning, create static files instead of relying on the server to run and process them during runtime). We do this by adding the following code to our nuxt.config.ts:

```
routeRules: {
    '/': { prerender: true }
},
```

Fig. 14. Configuring nuxt.config.ts: prerender true

Alternatively, we can also manually generate them by adding:

```
generate: {
    routes: [
        '/',
    ]
},
```

Fig. 15. Configuring nuxt.config.ts: generating route

B. *Configuring Tauri*

Lastly, we need to specify the path to our static files folder. In *tauri.conf.json*, point to the public folder located in the .**output** folder.

```
"build": {
    "distDir": "../.output/public"
  },
```

Fig. 16. Configuring tauri.conf.json

Text heads organize the topics on a relational, hierarchical basis. For example, the paper title is the primary text head because all subsequent material relates and elaborates on this one topic. If there are two or more sub-topics, the next level head (uppercase Roman numerals) should be used and, conversely, if there are not at least two sub-topics, then no subheads should be introduced. Styles named “Heading 1”, “Heading 2”, “Heading 3”, and “Heading 4” are prescribed.

VI. Conclusion

Finally, we’re done with our application, so we build our files and create a repository on GitHub. The repository can be found at https://github.com/kinxyo/CooperAI. Although this research was a success yet there are still many limitations that keep it from being perfect. To address them:

a) **Time Constraint**: 28 days were simply not enough. Although they were enough to create an application, it is still quite basic overall. Integrating and adding more features would require more time.

b) **No offline mode**: The application is not offline due to its dependence on Rest endpoint for OpenAI models. The original intent was to create my own AI but that would obviously require more than 1 month hence the idea had to scraped and a compromise was made.

c) **Dependance on OpenAI**: Involvement of third-party for a data-secure application can be tricky. On top of that, a subscription needs to be purchased for AI services which could be avoided if I create my own AI model specialized for treating mental illness.

References

[1] P. K. Kushwaha and M. Kumaresan, "Machine learning algorithm in healthcare system: A Review," 2021 International Conference on Technological Advancements and Innovations (ICTAI), Tashkent, Uzbekistan, 2021, pp. 478-481, doi: 10.1109/ICTAI53825.2021.9673220.

[2] P. K. Kushwaha, V. Bibhu, B. P. Lohani and D. Singh, "Review on information security, laws and ethical issues with online financial system," 2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH), Greater Noida, India, 2016, pp. 49-53, doi: 10.1109/ICICCS.2016.7542350.

[3] G. Gulati, B. P. Lohani and P. K. Kushwaha, “A Novel Application Of IoT In Empowering Women Safety Using GPS Tracking Module,” 2020 Research, Innovation, Knowledge Management and Technology Application for Business Sustainability (INBUSH), Greater Noida, India, 2020, pp. 131-137, doi: 10.1109/INBUSH46973.2020.9392193.

[4] D. Pareta, I. N. Verma, B. P. Lohani, P. K. Kushwaha and V. Bibhu, "IoT Enabled Smart and Efficient Musical Water Fountain," 2022 2nd International Conference on Innovative Practices in Technology and Management (ICIPTM), Gautam Buddha Nagar, India, 2022, pp. 369-373, doi: 10.1109/ICIPTM54933.2022.9754129.

[5] B. P. Lohani, M. Trivedi, R. J. Singh, V. Bibhu, S. Ranjan and P. K. Kushwaha, "Machine Learning Based Model for Prediction of Loan Approval," 2022 3rd International Conference on Intelligent Engineering and Management (ICIEM), London, United Kingdom, 2022, pp. 465-470, doi: 10.1109/ICIEM54221.2022.9853160.

[6] V. Bibhu, A. Kumar, B. P. Lohani and P. K. Kushwaha, "Robust Secured Framework for Online Business Transactions over Public Network," 2021 2nd International Conference on Intelligent Engineering and Management (ICIEM), London, United Kingdom, 2021, pp. 555-560, doi: 10.1109/ICIEM51511.2021.9445380.

[7] V. Bibhu, P. K. Kushwaha and B. P. Lohani, "A review of security of the cloud computing over business with implementation," 2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH), Greater Noida, India, 2016, pp. 192-198, doi: 10.1109/ICICCS.2016.7542342.

[8] Amardeep Gupta and Ranjeet Kumar Rout, “ROTEE: Remora Optimization and Tunicate swarm algorithm-based Energy-Efficient cluster-based routing for EHenabled heterogeneous WSNs,” International Journal of Communication System (Q1), vol. 33, no. 6, pp. 1-23, 2022. DOI:https://doi.org/10.1002/dac.5372. (IF: 2.0)

[9] A. D. Gupta and R. K. Rout, "An Effective Optimization Method for Energy Efficient Clustering in EH Wireless Sensor Networks," 2021

International Conference on Technological Advancements and Innovations (ICTAI), Tashkent, Uzbekistan, 2021, pp. 699-702, doi: 10.1109/ICTAI53825.2021.9673312.

[10] S. S. N. Challapalli, P. Kaushik, S. Suman, B. D. Shivahare, V. Bibhu and A. D. Gupta, "Web Development and performance comparison of Web Development Technologies in Node.js and Python," 2021 International Conference on Technological Advancements and Innovations (ICTAI), Tashkent, Uzbekistan, 2021, pp. 303-307, doi: 10.1109/ICTAI53825.2021.9673464.

[11] A. D. Gupta, S. Suman, S. S. N. Challapalli, P. Kaushik and V. Bibhu, "Survey Paper: Comparative Study of Machine Learning Techniques and its Recent Applications," 2022 2nd International Conference on Innovative Practices in Technology and Management (ICIPTM), Gautam Buddha Nagar, India, 2022, pp. 449-454, doi: 10.1109/ICIPTM54933.2022.9754206.

[12] A. Rana, A. Reddy, A. Shrivastava, D. Verma, M. S. Ansari and D. Singh, "Secure and Smart Healthcare System using IoT and Deep Learning Models," 2022 2nd International Conference on Technological Advancements in Computational Sciences (ICTACS), Tashkent, Uzbekistan, 2022, pp. 915-922, doi: 10.1109/ICTACS56270.2022.9988676.

[13] Rana A, Goyal N, Ahlawat A, Jamwal S, Reddy B, Sharma S. Mechanisms involved in attenuated cardio-protective role of ischemic preconditioning in metabolic disorders. Perfusion. 2015;30(2):94-105. doi:10.1177/0267659114536760

[14] Ahlawat, A., Rana, A., Goyal, N. et al. Potential role of nitric oxide synthase isoforms in pathophysiology of neuropathic pain. Inflammopharmacol 22, 269–278 (2014). https://doi.org/10.1007/s10787-014-0213-0

[15] A. R. Yeruva, P. Choudhari, A. Shrivastava, D. Verma, S. Shaw and A. Rana, "Covid-19 Disease Detection using Chest X-Ray Images by Means of CNN," 2022 2nd International Conference on Technological Advancements in Computational Sciences (ICTACS), Tashkent, Uzbekistan, 2022, pp. 625-631, doi: 10.1109/ICTACS56270.2022.9988148.

[16] Ghosh, S., Rana, A., & Kansal, V. (2020). A benchmarking framework using nonlinear manifold detection techniques for software defect prediction. International Journal of Computational Science and Engineering, 21(4), 593-614.

[17] Raghavendra, M. S., Chawla, P., & Rana, A. (2020, June). A survey of optimization algorithms for fog computing service placement. In 2020 8th international conference on reliability, infocom technologies and optimization (trends and future directions)(ICRITO) (pp. 259-262). IEEE.

[18] Gupta, S., Rana, A., & Kansal, V. (2020). Optimization in wireless sensor network using soft computing. In Proceedings of the Third International Conference on Computational Intelligence and Informatics: ICCII 2018 (pp. 801-810). Springer Singapore.

[19] Kunwar, V., Agarwal, N., Rana, A., & Pandey, J. P. (2018). Load balancing in cloud—a systematic review. Big Data Analytics: Proceedings of CSI 2015, 583-593.

[20] Chawla, P., Chana, I., & Rana, A. (2015). A novel strategy for automatic test data generation using soft computing technique. Frontiers of Computer Science, 9, 346-363.

[21] Walia, H., Rana, A., & Kansal, V. (2017, September). A Naïve Bayes Approach for working on Gurmukhi Word Sense Disambiguation. In 2017 6th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO) (pp. 432-435). IEEE.

[22] Dash, Y., Dubey, S. K., & Rana, A. (2012). Maintainability prediction of object oriented software system by using artificial neural network approach. International Journal of Soft Computing and Engineering (IJSCE), 2(2), 420-423.

[23] Dubey, S. K., & Rana, A. (2010). A comprehensive assessment of object-oriented software systems using metrics approach. International Journal on Computer Science and Engineering, 2(8), 2726-2730.

[24] S. Gupta, A. Rana and V. Kansal, "Comparison of Heuristic techniques:A case of TSP," 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 2020, pp. 172-177, doi: 10.1109/Confluence47617.2020.9058211.

[25] Ghosh, S., Rana, A., & Kansal, V. (2018). A nonlinear manifold detection based model for software defect prediction. Procedia computer science, 132, 581-594.

[26] Chawla, P., Chana, I., & Rana, A. (2016). Cloud-based automatic test data generation framework. Journal of Computer and System Sciences, 82(5), 712-738.

[27] Bhardwaj, M., & Rana, A. (2016). Key Software Metrics and its Impact on each other for Software Development Projects. International Journal of Electrical & Computer Engineering (2088-8708), 6(1).

[28] Rana, A., & Sharma, S. (2016). Mechanism of sphingosine-1-phosphate induced cardioprotection against I/R injury in diabetic rat heart: Possible involvement of glycogen synthase kinase 3β and mitochondrial permeability transition pore. Clinical and Experimental Pharmacology and Physiology, 43(2), 166-173.

[29] G. Dubey, A. Rana and N. K. Shukla, "User reviews data analysis using opinion mining on web," 2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE), Greater Noida, India, 2015, pp. 603-612, doi: 10.1109/ABLAZE.2015.7154934.

[30] Ghosh, S., Rana, A., Kansal, V. (2017). Predicting Defect of Software System. In: Satapathy, S., Bhateja, V., Udgata, S., Pattnaik, P. (eds) Proceedings of the 5th International Conference on Frontiers in Intelligent Computing: Theory and Applications . Advances in Intelligent Systems and Computing, vol 516. Springer, Singapore. https://doi.org/10.1007/978-981-10-3156-4_6

[31] Sanjay Kumar Dubey, Ajay Rana, and Yajnaseni Dash. 2012. Maintainability prediction of object-oriented software system by multilayer perceptron model. SIGSOFT Softw. Eng. Notes 37, 5 (September 2012), 1–4. https://doi.org/10.1145/2347696.2347703

[32] S. Chawla, G. Dubey and A. Rana, "Product opinion mining using sentiment analysis on smartphone reviews," 2017 6th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, 2017, pp. 377-383, doi: 10.1109/ICRITO.2017.8342455.

[33] Dubey, S. K., Rana, A., & Sharma, A. (2012). Usability evaluation of object oriented software system using fuzzy logic approach. International Journal of Computer Applications, 43(19), 1-6.

[34] Saini, Rimmi, Sanjay Kumar Dubey, and Ajay Rana. "Analytical study of maintainability models for quality evaluation." Indian Journal of Computer Science and Engineering 2.3 (2011): 449-454.

[35] Ghosh, Soumi, Ajay Rana, and Vineet Kansal. "A statistical comparison for evaluating the effectiveness of linear and nonlinear manifold detection techniques for software defect prediction." International Journal of Advanced Intelligence Paradigms 12.3-4 (2019): 370-391.

[36] A. Singh, M. Chaudhary, A. Rana and G. Dubey, "Online Mining of data to generate association rule mining in large databases," 2011 International Conference on Recent Trends in Information Systems, Kolkata, India, 2011, pp. 126-131, doi: 10.1109/ReTIS.2011.6146853.

[37] N. Tyagi, A. Rana and V. Kansal, "Creating Elasticity with Enhanced Weighted Optimization Load Balancing Algorithm in Cloud Computing," 2019 Amity International Conference on Artificial Intelligence (AICAI), Dubai, United Arab Emirates, 2019, pp. 600-604, doi: 10.1109/AICAI.2019.8701375.

[38] Dubey, Sanjay Kumar, and Ajay Rana. "A fuzzy approach for evaluation of maintainability of object oriented software system." International Journal of Computer Applications 49.21 (2012).