Expert API for Early Detection of TB Disease with Forward Chaining and Certainty Factor Algorithms

Nicholas Dwiarto Wirasbawa, Christian Teguh Prasetya Widjaja, Christian Imanuel Wenji and Seng Hansun*
E-mail: seng.hansun@lecturer.umn.ac.id
Informatics Department, Universitas Multimedia Nusantara, Tangerang, Indonesia

Keywords: certainty factor, expert system, forward chaining, REST API, tuberculosis

Received: January 27, 2022

Despite being a curable disease, Tuberculosis has become the leading cause of death of infectious disease prior to COVID-19. It has asymptomatic infections that are hard to detect for weeks or years. Although there have been many studies on Tuberculosis disease detection and prevention, very few of them discuss the creation of an expert system based on API. Hence, in this study we propose an Expert API that implements Forward Chaining and Certainty Factor algorithms for the task of Tuberculosis early detection. The evaluation of the proposed system was carried out using several testing methods and in-depth interviews with medical experts. We got a satisfactory result for this study.

Povzetek: Razvita je izvirna metoda za zgodnjo detekcijo tuberkuloze.

1 Introduction

Tuberculosis (TB) is a disease that can be cured with proper treatment and medicines, but it has become the leading cause of death of infectious disease (1). Indonesia is the second-highest contributor to Tuberculosis cases in 2020 (2), and this is evident from Tuberculosis sufferers. If they do not have proper care and treatment, it can cause death in 45% of patients who do not have HIV and almost 100% in patients who do have HIV. The World Health Organization (WHO) aims to eliminate Tuberculosis from the face of the Earth through its End TB strategy (3).

Tuberculosis is a disease that has asymptomatic infections that are hard to detect for weeks or even decades, which led many people that are infected did not realize it and made the number of cases increase (4). Therefore, it is often overlooked by medical personnel and can be difficult to diagnose and treat. Nonetheless, this disease is very likely to be treated. It is proven from 2000-2019 that around 63 million Tuberculosis sufferers have recovered. Tuberculosis itself is a disease that is better treated when it is in its early stages, but early detection tools for Tuberculosis have their own challenges in the form of low accuracy and economic conditions (5). Anyone can be infected with Tuberculosis, and the disease itself is very dangerous, especially for people infected with HIV.

Indonesia has been one of the highest contributors to Tuberculosis patients in the world, therefore early detection of tuberculosis is needed in Indonesia to treat patients before it is too late. The existence of early detection for Tuberculosis sufferers can help the treatment process and shorten the medication time period. Early detection of the disease will also help to realize the ideals of Indonesia and the world to eradicate the Tuberculosis disease.

There has not yet been any research about creating an expert system based on an Application Programming Interface (API). Many previous research that was explored beforehand only created an expert system that is tightly coupled with the web front-end (6,7). Creating an API would allow for the expert system to be more portable, as APIs are technology agnostic and can be consumed by any interfaces, such as web-based technologies, mobile-based technologies, Internet of Things devices, and even command line interfaces.

API is also used to achieve more separation of concerns and is easier to perform scalability on, as API is completely separate from the front-end and is able to communicate with anything as long as it conforms to the interface (8). The API architecture used in this research project is Representational State Transfer or REST architecture (9). Previous research has shown that REST API can make a system to be more scalable and less coupled (10). The API will return the response in JavaScript Object Notation (JSON) format.

The expert system created in this research project will use Forward Chaining and Certainty Factor algorithms to determine the level of diagnosis accuracy. The main technologies used in the creation of this system are Go and TypeScript. The internal evaluation of the system is carried out using the white-box testing method with the use of unit testing and integration testing, which are to test each and every function in the system and determine whether they pass the test cases or not. External evaluation of this system is done by the help of medical experts who are experienced in dealing with Tuberculosis disease. The

* Corresponding author

application is complete with the API, and will also implement internationalization to provide the application in multiple languages (English and Bahasa Indonesia) that conforms to the previous research (11).

It is hoped that this research can be the foundation for people who want to develop an expert system of their own, and we hope that this research can help to diagnose Tuberculosis faster, thus helping the mission of eradicating Tuberculosis disease from the face of the Earth.

2 Research methods

This section will explain the research methodology that is used in this research, the algorithms being used, and also explain the system design that is chosen to fulfill the research objectives.

2.1 Expert system

An expert system is a system that has the same knowledge as a human expert and is able to make decisions based on that knowledge (12). The knowledge is processed by the system to help the user that does not have such expert knowledge. The expert system will help to process input from the users and give an accurate output based on the knowledge. The system will also give an unbiased outcome that a human usually has.

2.2 Forward chaining

Forward-chaining algorithm is one of two main methods that are used by inference engines for its reasoning; the algorithm can be described as the application of repetition of modus ponens which consist of one set inference rule and a valid argument (13). This algorithm is useful for decision-making on the knowledge base that is given by a domain expert. Forward chaining is able to process the knowledge base as a tool to process input and give a desirable outcome based on the knowledge stored within the system.

2.3 Certainty factor

The Certainty Factor is usually used as a parameter to measure the trust level of owed information. This algorithm has been used in a lot of expert system applications. The Certainty Factor was created by Shortliffe Buchanan to make MYCIN - one of the earliest expert systems, and used to measure its trust level. The algorithm uses Eq. (1) to determine the trust level (i.e., the confidence level) (14,15).

$$CF(H,E) = MB(H|E) - MD(H|E) \quad (1)$$

where

$$MB(H|E) = \begin{cases} 1, & P(H) = 1 \\ \dfrac{\max[P(H|E),P(H)] - P(H)}{\max[1,0] - P(H)}, & otherwise \end{cases}$$

$$MD(H|E) = \begin{cases} 1, & P(H) = 1 \\ \dfrac{\min[P(H|E),P(H)] - P(H)}{\min[1,0] - P(H)}, & otherwise \end{cases}$$

and H is hypothesis, E is evidence, $P(H)$ is the probability of H, $P(H|E)$ is the conditional probability of H given E, $MB(H|E)$ is the measure of belief of H given E, and $MD(H|E)$ is the measure of disbelief of H given E.

2.4 Proposed system design

Figure 1 describes the process of the system. First, a user will diagnose him/herself with the symptoms that are available in the system. Then, the user will send their symptoms to the REST API to be calculated and processed further with two algorithms, namely Forward Chaining and Certainty Factor. If the user's input is somehow not valid, then the user will receive an error message and they have to input their symptoms again. If the input is valid, the user will receive a JSON API response whose content is the result of the diagnosis and also the information about prediction on Tuberculosis disease.

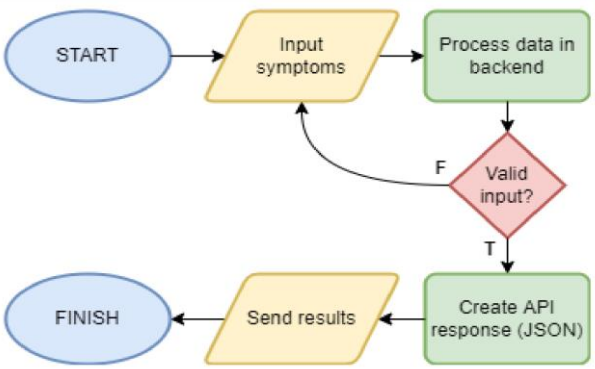

Figure 1: Flowchart of the system.

Figure 2 shows the use-case diagram of the system. In the system, the target users are two kinds of users, i.e., ordinary users and medical experts. A user is able to self-diagnose him/herself and is also able to check information about Tuberculosis in the system. An expert is able to do all of the previous use-cases, but s/he has an additional use-case and that is to evaluate the system. An expert who uses this system should always evaluate the system as time progresses in order to ensure that the symptoms, diseases, and information are always up to date and as accurate as possible.

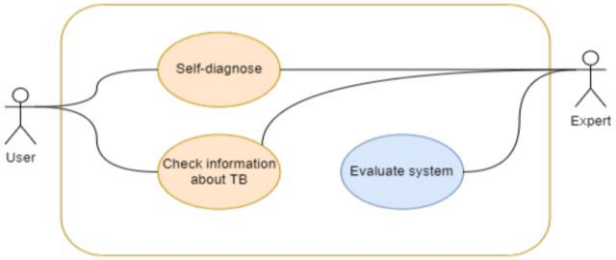

Figure 2: Use-case diagram of the system.

Figure 3 illustrates the architecture design of the system. A user is able to send a request to the web front-end, and said front-end will forward the request to the REST API back-end. The back-end will process the request according to the algorithms specified, and it will return the results (as a JSON response) to the front-end. Finally, the front-end will send back the response to the user as a Hyper Text Markup Language (HTML) document.

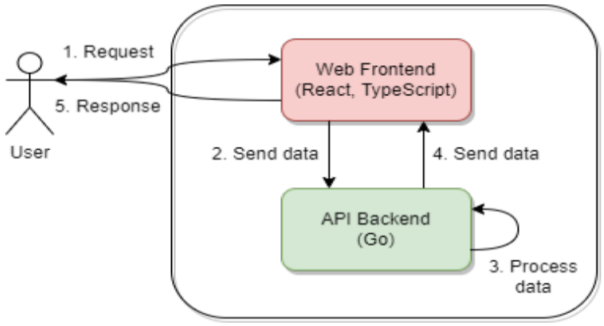

Figure 3: System architecture plan.

3 Implementation results

We have successfully implemented the REST API together with a web application as explained in the following subsections.

3.1 Data processing

Before delving further into the implementation, it is necessary to find and gather datasets in order to be used as the base estimations of the Expert System before consulting with the pertaining medical experts. In this research, the dataset used is from Victor Caelina, which is stored in Kaggle (18). The dataset provides a number of symptoms and a 'flag' to know if the person did experience the symptom(s) or not. In order to know the quantitative weight to be used in Forward Chaining and Certainty Factor algorithms, we calculated them with the following conditions.

For the Forward Chaining algorithm, we calculated the weight by calculating the average of each patient's total number of symptoms. From this, we can know the usual number of symptoms that a person usually has when they are infected with Tuberculosis.

For the Certainty Factor algorithm, we calculated the weight by calculating the average number of each symptom. From this, we can know the 'weight' of every symptom in the dataset (the 'percentage' of each).

From the data analysis process in a separate spreadsheet file, it was discovered that people usually experience seven symptoms if they are positively infected with Tuberculosis. It is also discovered the symptoms in the dataset have the following quantitative weights:

- Fever for two weeks has a weight of 0.513.
- Coughing blood has a weight of 0.475.
- Sputum mixed with blood has a weight of 0.519.
- Night sweats have a weight of 0.514.
- Chest pain has a weight of 0.494.
- Back pain has a weight of 0.511.
- Shortness of breath has a weight of 0.487.
- Weight loss has a weight of 0.521.
- Feeling tired has a weight of 0.496.
- Lumps appearing have a weight of 0.484.
- Continuous coughs have a weight of 0.493.
- Swollen lymph nodes have a weight of 0.478.
- Loss of appetite has a weight of 0.488.

Certainty Factor requires a user to also input their own 'certainty weight' (the quantitative weight of a user's certainty level to having experienced the symptom) to also be used as a metric for the calculation. We assume the certainty weights to be as follows:

- 'I do not/never feel that symptom' has a weight of 0.
- 'I sometimes feel that symptom' has a weight of 0.25.
- 'I often feel that symptom' has a weight of 0.75.
- 'I strongly feel that symptom' has a weight of 1.

From here on, it is now possible to start implementing the full system (web application and the REST API).

3.2 Technologies

The following technologies are used in order to construct the system.

- React.js with TypeScript as the front-end engine of the system;
- Go as the back-end engine of the system (as the REST API and the inference engine);
- Git and GitHub for the code version control;
- Docker as the containerization technology to produce deterministic builds and to ease up deployment;
- GitHub Actions to perform CI/CD in our system;
- Yarn and Go Modules as the package managers;
- Heroku as the native cloud deployment solution.

The system is built with Go in order to maximize concurrency processing. From empirical observations, APIs built with Go are much faster than most interpreted and compiled languages due to its native support for concurrency and coroutines, with speed that is second only to Rust. React is used to build high-performance web applications with a simple and intuitive UI. Docker, GitHub Actions, and Heroku are used to help with the deployment process.

3.3 Results

The system's core functionalities are described in this section. When a user enters the web application, they will be shown the main page of the application. All of the pages are developed with UI/UX in mind, and conforms to Shneiderman's Eight Golden Rules (19). This is to ensure that the system is properly designed and is easy to use by the majority of the users. The main page of the system is as shown in Figure 4.

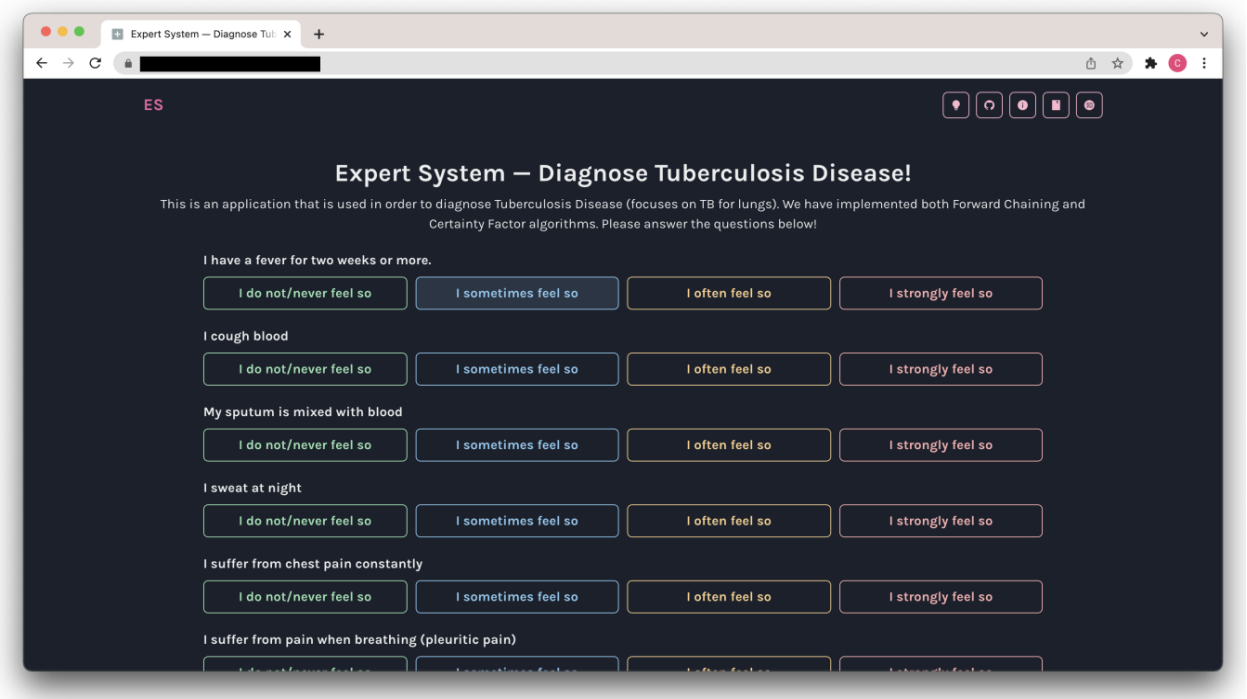

Figure 4: System's homepage.

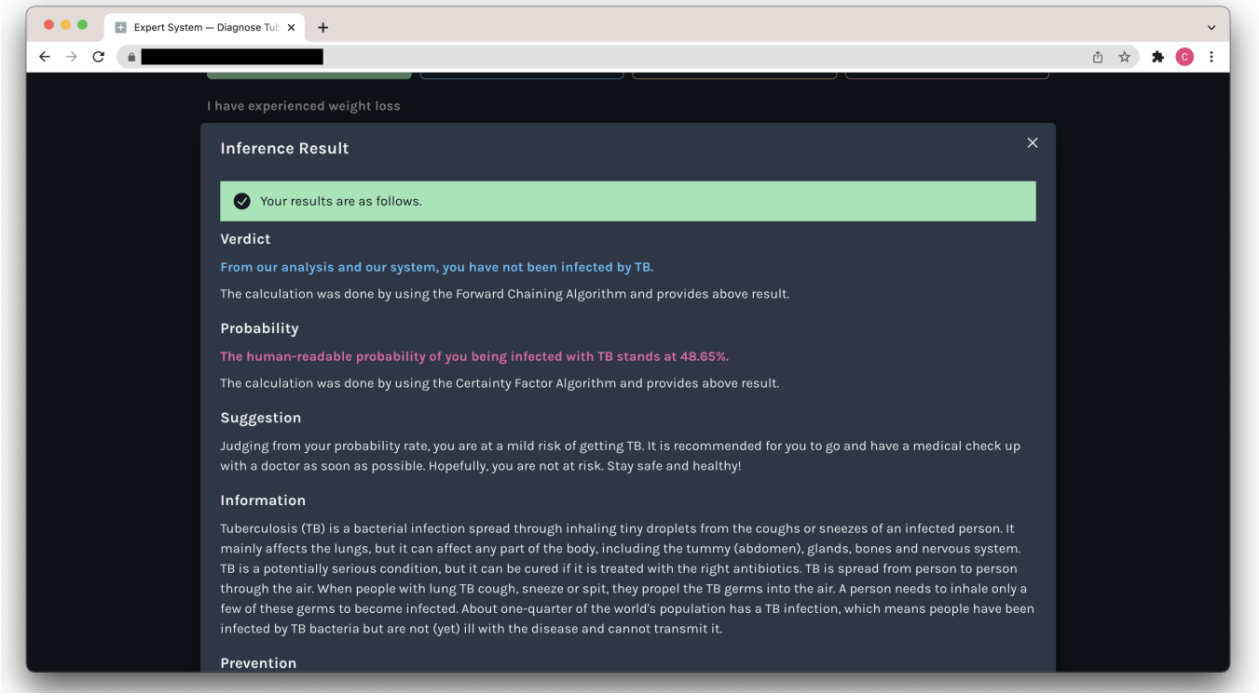

Figure 5: Results modal.

The right part of the system's header shows several icons that can be clicked, such as a light bulb button (to switch the system's theme), a GitHub icon (to access the GitHub repository), an 'i' icon (to show a modal displaying information about the system), a book icon (to show a modal displaying disclaimer and terms), and a Kanji icon (to change the system's language). A user can input their own diagnosis by choosing one of the available choices that suits their own condition the most. After the user has finished answering all of the questions, the user can click on the 'Results' button and they will be shown the results modal page, as shown in Figure 5.

The evaluation results will show the probability of a user's being infected with Tuberculosis (calculation is from the Certainty Factor algorithm), and a verdict of the user's condition, whether they are infected with Tuberculosis or not (using Forward Chaining algorithm). The result from this diagnosis is not recommended to be taken as it is, and the user will be recommended by the system to visit a medical expert to further diagnose their condition.

In addition, the evaluation results will also show the raw JSON data (from the API response, as this research is about the implementation of an API) for people who are interested in seeing the raw data. The raw JSON data will return an object consisting of the status of the request, the status code, a message, and another object (denoted as 'data') consisting of the verdict, probability, and the disease data. The disease data will show the identification, name, description, treatment, prevention, sources of information, and the symptoms, complete with the weights and its name.

The output of this research project could be accessed from several links, such as the source code at a GitHub repository in https://github.com/lauslim12/expert-systems, and the live version of the system at the link inside the repository. For further information and contribution to this research project, interested readers are welcomed to contact the first author at nicholas.dwiarto@student.umn.ac.id or the corresponding author.

4 Evaluation and Discussion

For the evaluation part of the built system, we used both internal and external testing.

4.1 Internal Testing

The internal testing process uses white-box testing method and implements unit testing and integration testing in order to make sure that the system works properly. The unit testing process is satisfactory, and we have received 100% code coverage in our whole API, as shown in Figure 6.

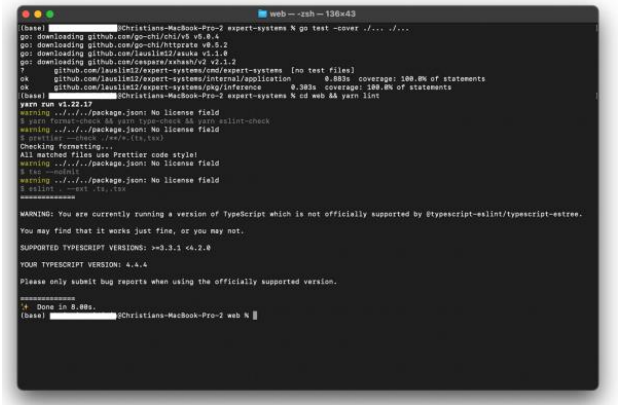

Figure 6: Unit testing coverage.

We have also performed integration tests. Integration tests are tests that are performed when the application is running in order to ensure that the application works as expected in production scenarios. Figure 7 shows the results of our integration tests, which are successfully done according to our expectations. It also shows the JSON response of the system.

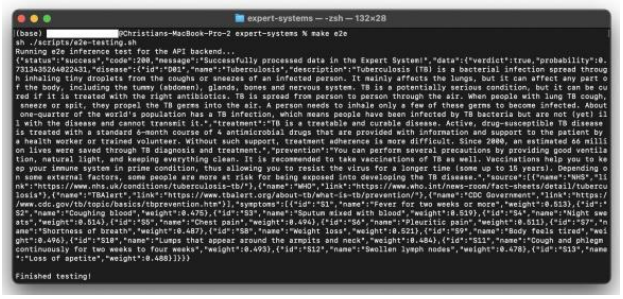

Figure 7: Integration tests.

4.2 External Testing

The external testing process uses black-box testing method and we relied on medical experts on Tuberculosis disease to perform a deep evaluation of the system. In addition, external testing is also performed by conducting a quick interview to twenty people who may have to utilize this application to analyze their own health condition (and the potential of being infected with Tuberculosis).

Halodoc application is used in order to find medical experts to consult and evaluate our system with. Halodoc is a medical consulting application that is catered for people from Indonesia (20). We thoroughly interviewed two experts and an undergraduate medical student:

Dr. Maria from Halodoc evaluated that our system is easy to use, fully responsive on many devices, and accurate. Dr. Maria suggested that the system should focus more on TB for lungs at the moment, and she also suggested removing non-relevant TB for lung symptoms.

Dr. Devi from Halodoc evaluated that our system is easy to use, accurate, and responsive. She suggested fixing several wordings about TB symptoms in the system, and also suggested adding a single additional symptom.

Daniel is an undergraduate student in Medical Science. He was interviewed and stated that the system is easy to use, informative, helpful, and has the potential of being used by hospitals if we add more inference engines for additional diseases.

Non-medical experts whom we have interviewed all claimed that the system possesses a good user interface, easy to use, useful, and very informative. They all consider using the system to diagnose themselves in the future.

4.3 Comparisons

Table 1 describes the relevant works of this research, and showcases their differences. Metrics used as the comparisons are evaluation methods, algorithms used, and the media used to deliver the results of the research.

When compared with previous research as shown in Table 1, it is depicted that the current research contains two algorithms, is tested multiple times, and processes the whole input and output via an API which is connected to a website. This is different from the previous papers, which usually only use one algorithm, only tested by one evaluation scenario, and use a coupled media.

Table 1: Related Works.

Authors	Evaluation	Algorithm	Media
Kusnadi, A. (16)	Literature review	Forward Chaining	Desktop application
Didik W, et al. (17)	Interview with experts	Forward Chaining	Mobile application
Leonardo J, et al. (6)	Interview with experts	Analytical Hierarchy Process	Web-based application
Hossain MS, et al. (7)	Interview with experts	Fuzzy	Web-based application
This research	Unit tests, integration tests, user acceptance testing, and interview with experts	Forward Chaining and Certainty Factor	Web-based application that is connected to an API

5 Conclusion

In this research, we have successfully built an API, and in extension of a web application, to self-diagnose Tuberculosis disease with Forward Chaining and Certainty Factor algorithms that is accurate and easy to use according to the reviewers. The system is built by using mainly Go, React with TypeScript, and Docker, with Test Driven Development as the programming paradigm. Evaluation was done with white-box testing (unit tests and integration tests), and black-box testing (interviews with general users and medical experts to give feedback to improve the system even further).

The internal testing processes have been successfully run and results in 100% code coverage, which means that the system has passed all test cases and therefore no bugs are found in the application. All calculations are done according to the Forward Chaining and Certainty Factor algorithms and are accurate.

The external testing process has also been successfully done. Medical experts and general users whom we have interviewed claim that the application is accurate, provides a good user experience, responsive, easy to use, and provides complete information about Tuberculosis, from its symptoms, general information, prevention, and its potential treatments.

For future development, we would like to add more diseases into the system, as we have prepared and added the data structures, API responses, and data types that allow for horizontal scalability with minor to no problems at all. We would also like to create an open-source library about the expert system, so people who want to integrate their own applications with our system could just implement our library in their codebase, resulting in an integrated and a good developer experience. We would also like to try and implement different algorithms to try to diagnose the disease, for example by incorporating various Machine Learning and Deep Learning approaches in detecting TB disease (21, 22).

Acknowledgements

The authors would like to thank Universitas Multimedia Nusantara for the support and facilities given for this research project. We would also like to thank the following resources, people, and open-source libraries who have assisted in creating this web application: 1) Icons8.com for the hospital favorite icon, 2) Segun Adebayo for ChakraUI, 3) Peter Kieltyka for Chi web framework, 4) Kamijin Fanta for React Icons, 5) i18next for the internationalization library, 6) All testers of the system, who have spent their time evaluating, giving feedback, and helping us to improve this system, and 7) The anonymous reviewer(s) of this paper, who have provided valuable comments and corrections.

References

[1] Hansun S. TB CNR Prediction Using H-WEMA: A Five Years Reflection. Int J Adv Soft Comput its Appl, 12(3):1–10, 2020. http://www.i-csrs.org/Volumes/ijasca/2020.3.1.pdf

[2] Chakaya J, Khan M, Ntoumi F, Aklillu E, Fatima R, Mwaba P, et al. Global Tuberculosis Report 2020 – Reflections on the Global TB burden, treatment and prevention efforts. Int J Infect Dis, 113:S7–12, 2021. https://doi.org/10.1016/j.ijid.2021.02.107

[3] World Health Organization. The End TB Strategy [Internet]. WHO. 2022. Available from: https://www.who.int/teams/global-tuberculosis-programme/the-end-tb-strategy

[4] Rangaka MX, Cavalcante SC, Marais BJ, Thim S, Martinson NA, Swaminathan S, et al. Controlling the Seedbeds of Tuberculosis: Diagnosis and Treatment of Tuberculosis Infection. Lancet, 386(10010):2344–53, 2015. https://doi.org/10.1016/S0140-6736(15)00323-2

[5] Nurmaya ER. Tes Deteksi Dini TBC Belum Direkomendasikan, WHO Sebut Tingkat Akurasi Belum Bagus. suaramerdeka.com [Internet]. 2021 Jun 23; Available from: https://www.suaramerdeka.com/gaya-hidup/pr-04422025/tes-deteksidini-tbc-belum-direkomendasikan-who-sebut-tingkat-akurasi-belumbagus

[6] Leonardo J, Young JC, Hansun S. Early Detection of Pulmonary Tuberculosis Disease with Fuzzy AHP Expert System. Compusoft, 8(10):3444–7, 2019. https://www.ijact.in/index.php/ijact/article/view/1041/699

[7] Hossain MS, Ahmed F, Fatema-Tuj-Johora, Andersson K. A Belief Rule Based Expert System to Assess Tuberculosis under Uncertainty. J Med Syst, 30;41(3):43, 2017. https://doi.org/10.1007/s10916-017-0685-8

[8] Subramanian H, Raj P. Hands-On RESTful API Design Patterns and Best Practices: Design, develop,

and deploy highly adaptable, scalable, and secure RESTful web APIs. Packt Publishing; 2019.

[9] Fielding RT. Architectural Styles and the Design of Network-based Software Architectures. University of California; 2000. https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation_2up.pdf

[10] Wirasbawa ND, Wibawanto MDR, Kosasi A, Hansun S. Scalable Building Management System for Offices and Co-Working Spaces. Indian J Econ Bus, 20(2):451–61, 2021. http://www.ashwinanokha.com/resources/ijeb v20-2-30. Ijeb.pdf

[11] Lako C. Studia Universitatis Petru Maior. Philologia. Stud Univ Petru Maior Philol, (19):151–60, 2015. https://www.researchgate.net/publication/293653270_ON_INTERNATIONALIZATION_I18N

[12] Salman FM, Abu-Naser SS. Expert System for COVID-19 Diagnosis. Int J Acad Inf Syst Res, 4(3):1–13, 2020. http://dstore.alazhar.edu.ps/xmlui/bitstream/handle/123456789/588/IJAISR200301.pdf?sequence=1&isAllowed=y

[13] Akil I. Analisa Efektifitas Metode Forward Chaining dan Backward Chaining pada Sistem Pakar. J Pilar Nusa Mandiri, 13(1):35–42, 2017. https://ejournal.nusamandiri.ac.id/index.php/pilar/article/view/12/8

[14] Sembiring AS, Sulindawaty, Manahan O, Napitupulu MH, Hasugian PS, Riandari F, et al. Implementation of Certainty Factor Method for Expert System. J Phys Conf Ser, 1;1255(1):012065, 2019. https://doi.org/10.1088/1742-6596/1255/1/012065

[15] Zhang L, Zhang B. Information Synthesis in Multi-Granular Computing. In: Quotient Space Based Problem Solving, 105–27, 2014. https://doi.org/10.1016/B978-0-12-410387-0.00003-2

[16] Kusnadi A. Perancangan Aplikasi Sistem Pakar untuk Mendiagnosa Penyakit pada Manusia. Ultimatics: Jurnal Ilmu Teknik Informatika, 5(1);1-8, 2013. https://doi.org/10.31937/ti.v5i1.307

[17] Widianto ED, Zaituun YW, Windasari IP. Aplikasi Sistem Pakar Pendeteksi Penyakit tuberkulosis Berbasis Android. Khazanah Informatika: Jurnal Ilmu Komputer dan Informatika, 4(1), 2018. https://doi.org/10.23917/khif.v4i1.5496

[18] Caelina V. Tuberculosis Symptoms [Internet]. kaggle.com. 2021. Available from: https://www.kaggle.com/victorcaelina/tuberculosis-symptoms

[19] Aottiwerch N, Kokaew U. Design Computer-assisted Learning in an Online Augmented Reality Environment based on Shneiderman's Eight Golden Rules. In: 2017 14th International Joint Conference on Computer Science and Software Engineering (JCSSE), 1–5, 2017. https://doi.org/10.1109/JCSSE.2017.8025926

[20] Kushendriawan MA, Santoso HB, Putra POH, Schrepp M. Evaluating User Experience of a Mobile Health Application 'Halodoc' using User Experience Questionnaire and Usability Testing. J Sist Inf, 29; 17(1):58–71, 2021. https://doi.org/10.21609/jsi.v17i1.1063

[21] Gaafar AS, Dahr JM, Hamoud AK. Comparative Analysis of Performance of Deep Learning Classification Approach based on LSTM-RNN for Textual and Image Datasets. Informatica, 46:21–28, 2022. https://doi.org/10.31449/inf.v46i5.3872

[22] Gadri S. Developing an Efficient Predictive Model Based on ML and DL Approaches to Detect Diabetes. Informatica, 25:433-440, 2021. https://doi.org/10.31449/inf.v45i3.3041