A SMART MEDICINE MOBILE PLATFORM FOR INJURY DIAGNOSIS AND MENTAL STRESS MANAGEMENT USING ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Zelin Jason Hu[1], Garret Washburn[2]

[1]Westminster school, 995 Hopmeadow St. Simsbury, Connecticut, 06070
[2]Computer Science Department, California State Polytechnic University, Pomona, CA 91768

ABSTRACT

The need for immediate medical or psychological attention is something that humanity has always needed as an essential service [1]. However, it is apparent nowadays that most individuals would say that the medical attention they receive is hardly immediate and doesn't provide a diagnosis or advice in a quick fashion. To solve this issue, we propose the Physiomed mobile application available on both the Apple and Google Play Stores. The Physiomed mobile app provides immediate professional AI generated diagnosis and advice to individuals who have encountered physical or emotional symptoms in their day-to-day life [2]. The Physiomed application utilizes ChatGPT AI prompt generation on a hosted back-end server that receives the symptoms from different users and can provide accurate diagnosis based on real life medical data. The biggest challenge during development was creation of the prompt to receive insightful and accurate diagnosis, as well as the communication between the back-end server and the different user's mobile devices.

KEYWORDS

AI-powered diagnosis, Mobile health app, ChatGPT integration, Real-time medical advice

1. INTRODUCTION

The issue this paper seeks to address and provide a solution too is the absence of access to immediate and correct medical and emotional advice for any individual. The issue surrounding not having immediate medical advice is less about not knowing what to do, but more so what individuals tend to do without it [3]. Unfortunately, it is not an uncommon story for people to not know that a minor pain or injury is actually a sign of a much deeper issue, and due to this unknowing ignorance, do something to continue or worsen their condition. Additionally, the same can be said about one's mental well-being, it is also a very common occurrence for an individual to ignore their mental health and let their mental well-being passively worsen. Inherently, this issue is most common with sports athletes or those who are active, but this issue can also impact those who believe themselves to be ordinary people. In a study in which the elite youth athletes (age ranged 14-18 years old) were observed to see how frequently injuries occurred, "injury rates ranged from 1.4 to 4.6 injuries/1,000 hours, while competition rates were higher, ranging from 10.5 to 22.4 injuries per 1,000 hours". Additionally, "Injury rates in badminton were 2.8 for practice and 5.9 for games per 1000 hours," meaning, in a 10-hour

David C. Wyld et al. (Eds): CCSIT, NLPCL, AISC, ITE, NCWMC, DaKM, BIGML, SIPP, SOEN, PDCTA – 2024
pp. 259-269, 2024. - CS & IT - CSCP 2024 DOI: 10.5121/csit.2024.141722

badminton competition with 20 people, there is a %5 that a player will get injured. As such, the likelihood of injury is increasingly high, and the solution proposed in this paper seeks to remedy the aftermath.

Within the essay, three methodologies were reviewed in an attempt to observe what the industry standard is for medical advice applications. The three methodologies reviewed are Medic Chat, the WebMD Symptom Checker, and MediFind [5]. These three applications are essentially the standard for medical advice services online. Medic Chat is a website used to connect users with doctors in a timely fashion, however, that timely fashion on average takes around 2 hours and is not immediate [4]. The WebMD Symptom Checker aims to be an all-encompassing healthcare mobile application, and provides potential diagnosis based on a set few inputs [6]. The WebMD Symptom Checker does not have any custom inputs for specific user information and only gives diagnosis based on recorded issues. MediFind is a website that is quite similar to the WebMD Symptom Checker feature, in that it asks a series of questions and provides diagnosis in a similar fashion. However, neither of the two previously mentioned methodologies have the option to customize the symptom input and provide more background information to the diagnosis process. The Physiomed Mobile Application seeks to improve the design of all of these methods by allowing the user to completely customize the prompt input to include whatever information necessary to receive an insightful diagnosis [7].

The solution to the problem proposed in this paper is Physiomed, a mobile application designed to generate professional medical advice utilizing an AI model that responds to the user based on their input prompt containing their symptoms [8]. Physiomed solves the issue of the lack of immediate and ready medical and emotional advice by being present right on the users phone, allowing quick access to professional advice, potentially preventing the worsening of their symptoms. Additionally, as compared to another method of quick and easy diagnosis, using Google to diagnose one's symptoms is not as efficient as the Physiomed methodology. This is due to the nature of how the applications work, Google is merely a search query platform, meaning it's purpose is only to match search queries with what exists on the internet currently. However, the proposed Physiomed methodology provides a specialized AI model for creating diagnosis with the given symptoms. Additionally, the Physiomed mobile app can potentially save emergency room visits, as for relatively smaller incidents or injuries, the AI model can recommend potential solutions that can minimize risk and increase remediation. All of which would have been re-verbed at the emergency room had the user gone. On top of all of the physical injury prevention and remediation that can be done with the Physiomed application, Physiomed also offers a psychological health AI model that seeks to help the user diagnose their emotional symptoms and provide them with an idea of what could be mentally afflicting them. The combination of these two resources, Physiomed, seeks to be a one stop shop for quick and easy diagnosis before seeking help from a licensed professional.

After the development process, to ensure the mobile application was reliable and worked as intended, we went through the process of creating and conducting two major experiments to gather data on the efficacy of the app. The first experiment we conducted had to do with the AI model and how it generated responses based on user input. We figured that because the AI model directly uses user input in the prompt submission that the application could be vulnerable to prompt injection. To perform this experiment, we utilized a few different prompt injection techniques and recorded their successfulness. We found that asking the model to change persona's was an effective way to create a not intended response from the model, which led us into our next experiment. We figured that because we could change the prompt, that we could also ask for an unnecessary amount of output from the model and try to waste the allowed tokens from the model. To test this, we conducted an experiment asking the AI model to write a few short essays and recorded the amount of output we received from the model. We found that the

AI model never surpassed the token limit, as that was hard set in the prompt submission in the back-end server.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Use of Function

The gptAssessmentWithPersona function gives the AI model the personality of whatever we set it to; therefore, we can give it the character of a doctor which allows the model to respond to prompts with professionality and the knowledge of a doctor. One of the major problems that we encountered during the development of the ChatGPT implementation in this function was the deprecated functionalities in updates from the ChatGPT SDK library for Flutter [9]. Additionally, after some testing, we also found that users in other countries were unable to functionally use the app due to ChatGPT request restrictions. To remedy both of these issues, we deployed the use of a backend server in order to simplify the ChatGPT request code using Python and also enable users in other countries to be able to use the app.

2.2. How to Properly Organize the Diagnosis

Two other major components that make up the significant pieces of the Physiomed application are the saveAssessment and getAssessments backend functions. The purpose of these functions is to save the ChatGPT generated diagnosis in local storage on the user's device, so that the user has a saved log of all of their diagnoses and can view them. The saveAssessment function, as it's name suggests, handles the saving of the diagnosis in local storage, and the getAssessments functions handles the retrieving of items from local storage. The biggest challenge that we faced during development was how to properly organize the diagnosis, which we resolved by organizing them by the timestamp of the current time.

2.3. Ensuring Both Pages Worked Properly

The last major components of the Physiomed mobile application are the Physical Analysis and Psychological Analysis pages. The purpose of these pages are to receive the text input of the users symptoms, whether physical or emotional, and handle the sending of the prompt to the backend server. The biggest challenge that we faced during development was ensuring both pages worked properly and looked good. Additionally, the creation of the performAndSaveAssessment helper function was employed to handle the calling of the gptAssessmentWithPersona function and saving of the response with the saveAssessment function. Overall, the creation of this page was simple, however, the fine tuning was quite extensive.

3. SOLUTION

The Physiomed application consists of three major components: the ChatGPT prompt and server, the local storage saving and retrieval functionalities, and the frontend design of the Physical and Psychological analysis pages. The main structure of our program connects a main 'home' page to an option of two pages: psychological and physical. The two pages are where the input for prompt of symptoms is contained which when completed, takes the user to a history page where the data of the user is stored and can be accessed. Behind the scenes, when the user submits their symptoms, a request is made to the backend server, which subsequently sends the prompt to

ChatGPT and finally returns the result to the user. Our main development environment throughout the project was Android Studio, and we utilized Flutter, a mobile application development framework, to create and structure the Physiomed application [10]. The deployment of a backend server was necessary to avoid any regional roadblocks from online language learning models, as we were potentially implementing more processing logic in the future to give the user an even better diagnosis. For the backend server, we used Render.com as it provides a nice and easy way to set up a backend server for any application. The main technology that is used to provide the user with a diagnosis is ChatGPT, in which we give a specific persona, either a 'Physician' or a 'Psychologist,' and ask for a diagnosis given the user's symptoms. Finally, after each diagnosis generation is created, it is lastly stored in the user's local storage on their device and stored by the timestamp that the generation was created.

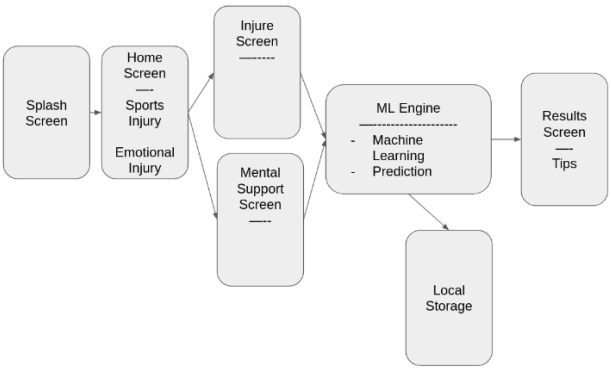

Figure 1. Overview of the solution

The first major component, the gptAssessmentWithPersona functionality, has the intended purpose of creating a diagnosis based on the user's given symptoms in a manner comparable to either physician or psychologist. This component relies on both the ChatGPT python api as well as the Render.com backend server, as both of these technologies come together to create the backend service that the gptAssessmentWithPersona function calls.

Figure 2. Screenshot of the assessment

```dart
Future<String> gptAssessmentWithPersona(String persona, String description) async {
  String apiUrl = "https://physiomed-server.onrender.com/gpt-assessment";

  final response = await http.post(
    Uri.parse(apiUrl),
    headers: {'Content-Type': 'application/json'},
    body: jsonEncode({'persona': persona, 'description': description})
  );

  if (response.statusCode == 200) {
    Map<String, dynamic> responseData = jsonDecode(response.body);
    return responseData['response'];
  } else {
    return 'Error occurred: ${response.body}';
  }
}
```

```python
# Initialize the Flask app
app = Flask(__name__)

# Set OpenAI API key
os.environ["OPENAI_API_KEY"] = "sk-proj-i9Khl3Cq7QEdBf88lv63T3BlbkFJWjOw6xiU9PiaS764KLnu"

def generic_prompt(persona, description):
    setup_prompt = f"Act as a {persona}."
    assessment_prompt = f"Provide an assessment of my symptoms based on the following descript

    prompt = PromptTemplate(
        template="{setup_prompt}\n\n{assessment_prompt}",
        input_variables=["setup_prompt", "assessment_prompt"]
    )

    llm = ChatOpenAI(model_name="gpt-3.5-turbo", temperature=0)

    # Chain
    rag_chain = prompt | llm | StrOutputParser()

    # Run
    generation = rag_chain.invoke({
        "setup_prompt": setup_prompt,
        "assessment_prompt": assessment_prompt
    })

    return generation

@app.route('/gpt-assessment', methods=['POST'])
def gpt_assessment():
    data = request.json
    persona = data.get('persona', '')
    description = data.get('description', '')

    try:
        response = generic_prompt(persona, description)
        return jsonify({'response': response})
    except Exception as e:
        return jsonify({'error': str(e)}), 500

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```

Figure 3. Screenshot of code 1

The first chunk of code above is the gptAssessmentWithPersona function that is present in the mobile application and is called when a user requests a diagnosis and has given their symptoms. The second chunk of code is the entire backend server that is running on a Render.com server and is called by the gptAssesssmentWithPersona function when the user submits a diagnosis. The gptAssessmentWithPersona function, upon receiving the persona and symptoms, forwards these values to the backend server by fitting the person and symptoms into a http request and sending to the backend server. In the backend server, after configuring the flask server and the ChatGPT api key as an environment variable, the '/gpt-assessment' route is initialized and the subsequent gpt_assmessment function that is run when visiting it is defined. In the gpt_assessment function, the persona and description keys are pulled from the received request and are then passed to the generic prompt function. The generic prompt function receives the persona and symptoms inputs and makes the request to ChatGPT after fitting the persona and symptoms variables into the customized prompt. Once the generation is received, it is passed back to the user's device for displaying by returning the generation as seen in the '/gpt_assessment' route.

The second major component, the local storage functionalities that save and get the AI generated assessments, is employed to provide a convenient way for users to reflect on their diagnosis and have them in a place they can refer to them later. The major library used for storing and maintaining the locally stored data is SharedPreferences, which is a common library used to store data locally in mobile applications.

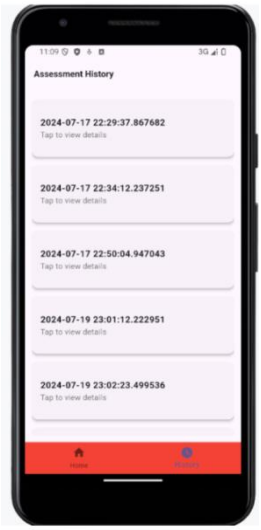

Figure 4. Screenshot of the assessment history

```
Future<void> saveAssessment(String assessment) async {
  Map<String, dynamic> assessmentMap = {};

  var currentTime = DateTime.now();

  final SharedPreferences preferences = await SharedPreferences.getInstance();

  if (preferences.containsKey("assessments")) {
    assessmentMap = jsonDecode(preferences.getString('assessments')!);
  }
  assessmentMap[currentTime.toString()] = assessment;

  String newAssessmentMap = jsonEncode(assessmentMap);
  await preferences.setString('assessments', newAssessmentMap);
  print(assessment);
}

Future getAssessments() async {
  final SharedPreferences preferences = await SharedPreferences.getInstance();

  if (preferences.containsKey("assessments")) {
    return jsonDecode(preferences.getString("assessments")!);
  } else {
    return null;
  }
}
```

Figure 5. Screenshot of code 2

The two methods you can see in the image above are the saveAssessment and the getAssessments functionalities that are used in the mobile application. The saveAssessment function is called right as an assessment has been received from the backend server, as it is necessary to save the assessment directly into local storage. The getAssessments function is called whenever the user visits the History page, as the getAssessments function returns all of the saved assessments in json format, which can be decoded and displayed neatly. In the saveAssessment method, after initializing the assessmentMap to be pulled from local storage, the current timestamp is collected so that the assessment passed as a parameter may be stored in local storage by this timestamp. Then, the instance of SharePreferences is initialized and it checks to see if SharedPreferences already has a key for ‘assessments.’ If the key is already found, the assessmentMap then is assigned to the json decoded assessments from storage. After that, the new assessment is added to the assessmentMap, with its key as the timestamp and the newAssessmentMap is then stored back into local storage. For the saveAssessments function, all that is done is SharedPreferences is initialized and the ‘assessments’ is pulled from local storage and returned.

The third and final major component in the Physiomed mobile application are the Physical and Psychological analysis pages. The overall purpose of these pages is to act as a medium to receive the user’s symptoms and give the user the option to submit to analysis by the AI model. These are essentially frontend pages, the main technology used to create them was the Flutter framework.

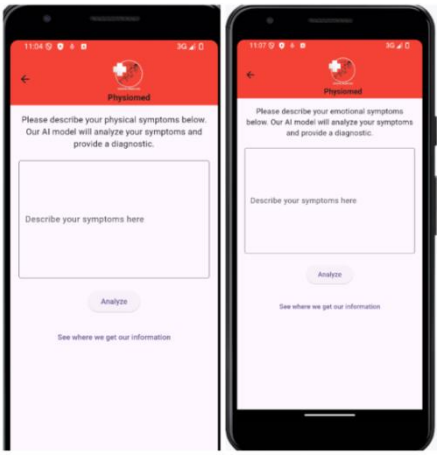

Figure 6. Screenshot of the APP

```
Future performAndSaveAssessment() async {
  setState(() {
    isLoading = true;
  });

  String result = await gptAssessmentWithPersona(Persona, description);
  await saveAssessment(result);

  setState(() {
    isLoading = false;
    Navigator.push(
        context,
        MaterialPageRoute(builder: (context) => HistoryPage())
    );
  });
}
```

```
body: SingleChildScrollView(
  child: Center(
    child: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Text(
            "Please describe your emotional symptoms below. Our AI model will analyze your symptoms and provide a diagnostic.",
            style: TextStyle(fontSize: 16),
            textAlign: TextAlign.center,
          ), // Text
          SizedBox(height: 20),
          TextField(
            maxLines: 8,
            decoration: InputDecoration(
              labelText: "Describe your symptoms here",
              border: OutlineInputBorder(),
            ), // InputDecoration
            onChanged: (text) {
              description = text;
            },
          ), // TextField
          SizedBox(height: 20),
          ElevatedButton(
            onPressed: () {
              performAndSaveAssessment();
            },
            child: Text("Analyze"),
          ), // ElevatedButton
          SizedBox(height: 20),
```

Figure 7. Screenshot of code 3

Both the Physical and Psychological pages are built to have essentially the same structure; however, they differ in some frontend text to make it obvious to the user that there is a difference as well as some background functionality. The encapsulating widget for the entire body of the Physical and Psychological pages is a SingleChildScrollView, which allows the user to scroll through the items inside of it. On each page there are three major components, the description at the top, the TextField for symptom input, and the ElevatedButton to submit the symptoms to the AI model by calling the performAndSaveAssessment function. The performAndSaveAssessment function calls the gptAssessmentWithPersona function given the symptoms and persona of the page, “Physician” or “Psychiatrist,” saves the response in local storage using saveAssessment, and then pushes the user to the History page. There are not any extra technologies that are used with this component of the project, besides the function calling of the saveAssessment and gptAssessmentWithPersona functions.

4. EXPERIMENT

4.1. Experiment 1

Due to the nature of the Physiomed Application relying heavily on the variable input of the user's symptoms, it could be possible that input prompt to the AI model is susceptible to prompt injection, that is, the user could submit a malicious prompt that is intended to get a response from the AI model separate from its intended use.

For this experiment, we intend to test to see if the prompt is susceptible to prompt injection by trying a few different strategies for prompt injection and seeing if we are able to get a response that is separate from the prompt's intended output. We will try a few strategies, and mark to see if they are successful in getting an unintended response from the AI model. One of the strategies will be to ask the AI model to disregard all previous instructions and do something unique, others will be more indirect to see if we can get incorrect information.

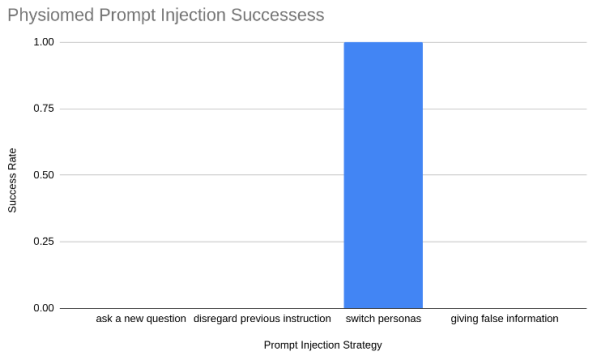

Figure 8. Figure of experiment 1

After conducting the experiment, the first thing we noticed and were quite surprised by was how the AI model did not directly ignore the previous instruction when asked in the prompt. The AI model did really good job of staying in character and only giving responses of that of a "Physician" or "Psychologist," and we were only really able to have it give us a response that was unintended by asking the model to stop pretending to be a Physician or Psychologist and ask it to pretend to be something else.

4.2. Experiment 2

After reflecting on the previously conducted experiment, we posed the question of what limits the user could reach when asking the AI model to generate a lot of text with the malicious purpose of wasting the AI model's tokens.

To perform this experiment, we used the previously tested persona switch prompt injection technique to have the AI model change its persona to being a famous writer. After creating this switch, we asked the AI model, as a famous writer, to write a 5-paragraph essay about a specific topic. The topics, as seen in the graph below, were specifically chosen in an attempt to draw out as many wasted tokens as possible. It's important to note that the AI model still gives a diagnosis for the symptoms inputted by the user, and only switches personas afterwards to the famous writer.

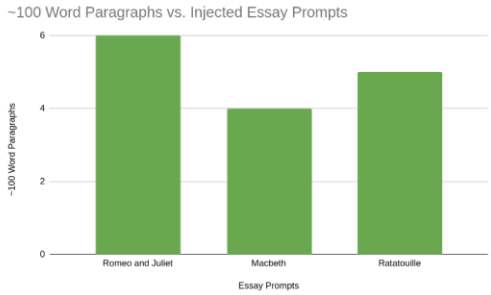

Figure 9. Figure of experiment 2

After conducting the experiment, and asking the model to write the essays, we found that on average the AI model produced around 5 paragraphs of around 100 words each. It is important to note that in all of these responses, that one of the paragraphs is of course the diagnosis to the symptoms, which in all of these cases we simply said "I have a headache." Initially, we were quite surprised with how the AI model was able to generate such a well thought out essay. However, after experimenting with the generation, and our prompts got more and more in depth, we became surprised with how the AI model wasn't generated what we had asked for. After considering how long our generations were becoming and how as we became more and more specific the AI model became less and less accurate, we figured that the AI model was becoming less accurate the closer it got to the token limit. The AI model takes into consideration the token limit, the amount of words the AI model is capable of producing, and will ensure it's response stays within it even if it means giving an inaccurate response.

5. RELATED WORK

Medic Chat, an online service where you can schedule a quick chat with a healthcare provider, serves as a tool for those who don't feel like they should go to the emergency room as they can simply and quickly talk to a doctor in the comfort of their own home [11]. As the user gets to talk to a physician or psychologist doctor, the solution is quite effective. However, the biggest identifiable issue we found during our investigation was the wait time to see a doctor. Seemingly all doctors on the site had around a 2 hour wait time to be seen. As an example, if a patient had a headache and their symptoms were bad enough they needed to go to the emergency room, but they were unaware, the Physiomed application could tell them right away. However, Medic Chat would take at least 2 hours to schedule the user with a doctor in order to get their advice.

The WebMD Symptom Checker is an all-in-one healthcare app that includes not only a symptom checker, but also contains the features of a medication lookup, a medication reminder system, a doctor appointment scheduler, and a system tracker to name a few [12]. While the WebMD Symptom Checker app is undoubtedly effective, its design is to do a lot of things sufficiently, and not one thing excellently. The WebMD Symptom Checker's diagnosis functionality consists of a symptom lookup, and only has a set amount of possible diagnosis. However, the Physiomed application has a lot more flexibility with the input, and the AI model is intended to give feedback for each unique case for each patient.

Another methodology of providing a patient with a quick possible diagnosis is MediFind [13]. MediFind is an online website that takes the user through a series of questions about themselves and their symptoms and then provides a series of possible condition diagnosis. The first noticeable issue with MediFind is that it doesn't aim to give any practical advice to treat minor symptoms, the service is entirely focused on providing diagnoses. Additionally, MediFind is limited in the options the user can fill out whereas the Physiomed application provides a medium

of communication with the AI model that is very flexible so the user can include any information they feel relevant.

6. CONCLUSIONS

After reviewing other methodologies and taking a look back at the development process, the most prominent area in which the Physiomed application lacks is the amount of features the application has. Right now, Physiomed only has a AI model prompt submission feature to get diagnosis, however, in the future we would like to add a few new features [14]. The list of new features starts with a map that displays the nearest medical facilities such as an emergency room, medical hospitals, or Psychologist offices. Additionally, a feature which allows the user to set up appointments with a medical professional from the hospitals would be amazing, that way if the AI model determines the user's symptoms to need proper medical attention it can recommend a doctor as well as a meeting time. Additionally, integrating the appointment scheduling and paperwork handling as a feature within the application would be amazing too, as that would enable users to easily maintain and deliver necessary paperwork for prescriptions or any necessary information. To add these features to the application, our first steps would be to add a pain checker page in the line of Physical Symptom analysis pages to ask where the user finds their pain [15]. Then, implementing the new features as their own unique pages able to be navigated to within the route widget at the bottom would likely be the way we implement them.

To conclude, throughout the development and review of the Physiomed application, it is our firm belief that the Physiomed application can serve as a valuable tool in anyone's pocket for getting quick and easy symptom diagnosis. Given more time, we are sure the Physiomed app could become even better as its purpose is to make a positive impact upon the user base that utilize the app.

REFERENCES

[1] Garg, Nitin, and Jonathan I. Silverberg. "Association between childhood allergic disease, psychological comorbidity, and injury requiring medical attention." Annals of Allergy, Asthma & Immunology 112.6 (2014): 525-532.

[2] Abdel-Khalek, Sayed, et al. "Leveraging AI-Generated Content for Synthetic Electronic Health Record Generation With Deep Learning-Based Diagnosis Model." IEEE Transactions on Consumer Electronics (2024).

[3] Das, Jishnu, Jeffrey Hammer, and Kenneth Leonard. "The quality of medical advice in low-income countries." Journal of Economic perspectives 22.2 (2008): 93-114.

[4] Patil, Sharvari, et al. "Medic: Smart Healthcare AI Assistant." Proceedings of the International Conference on Smart Data Intelligence (ICSMDI 2021). 2021.

[5] Le Bao, Khanh, et al. "MediFind-A medicine detection framework for Vietnamese medical prescription." 2023 International Conference on Advanced Computing and Analytics (ACOMPA). IEEE, 2023.

[6] Farmer, S. E. J., Matteo Bernardotto, and V. Singh. "How good is Internet self-diagnosis of ENT symptoms using Boots WebMD symptom checker?." Clinical Otolaryngology 36.5 (2011).

[7] Hoehle, Hartmut, and Viswanath Venkatesh. "Mobile application usability." MIS quarterly 39.2 (2015): 435-472.

[8] Reddy, Sandeep, et al. "A governance model for the application of AI in health care." Journal of the American Medical Informatics Association 27.3 (2020): 491-497.

[9] Lappalainen, Yrjo, and Nikesh Narayanan. "Aisha: A custom AI library chatbot using the ChatGPT API." Journal of Web Librarianship 17.3 (2023): 37-58.

[10] Wolfson, Mike, and Donn Felker. Android developer tools essentials: Android Studio to Zipalign. " O'Reilly Media, Inc.", 2013.

[11] Garrud, Paul, et al. "'"I'ma Medic"–a web-based, social capital approach to health careers." MedEdPublish 7 (2018).

[12] Fraser, Hamish, et al. "Comparison of diagnostic and triage accuracy of Ada health and WebMD symptom checkers, ChatGPT, and physicians for patients in an emergency department: clinical data analysis study." JMIR mHealth and uHealth 11.1 (2023): e49995.

[13] Trivedi, Devharsh, Vaishnavi Gopalakrishnan, and Dharati Dholariya. "MediSearch: Advanced Medical Web Search Engine." 2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC). IEEE, 2023.

[14] Mao, Bomin, et al. "AI models for green communications towards 6G." IEEE Communications Surveys & Tutorials 24.1 (2021): 210-247.

[15] Van Wijk, Cécile MT Gijsbers, and Annemarie M. Kolk. "Sex differences in physical symptoms: the contribution of symptom perception theory." Social science & medicine 45.2 (1997): 231-246.