Volume: 11 Issue: V Month of publication: May 2023
DOI: https://doi.org/10.22214/ijraset.2023.53505

Mental Health Web App

Prof. Rosemary Varghese[1] (Guide), Mridul Krishna P.M[2], Sanath Savio Nelson[3], Sidharth Rajesh[4]
Dept of Computer Science, ASIET, Kalady Ernakulam, Kerala

Abstract: The mental health app is designed to provide users with a holistic approach to managing their mental health. The app is built using React, and features a chatbot, a meditation timer, and a task manager. The chatbot provides users with a safe space to express their feelings and emotions, and to receive advice and support from the app's virtual assistant. The meditation timer helps users to focus on their breathing and relax, while the task manager helps users to organize and prioritize their daily tasks. With these features, the mental health app provides users with the tools they need to manage their mental health in a more efficient and effective way.
Keywords: mental health, depression, wellness, React, chatbot, meditation timer, task manager, holistic approach, user-friendly interface, data collection, data preprocessing, model training ,model evaluation

I. INTRODUCTION

Mental health is an important aspect of our overall well- being, and taking care of it can greatly improve our quality of life. With the increasing use of technology in our daily lives, mental health apps have become a popular way to support individuals in managing their mental health. This app aims to provide a comprehensive solution for mental health management, using React as the development framework. The app features a chatbot, meditation timer, and task manager to help users with their mental health needs. The chatbot provides a safe and confidential space for users to talk about their feelings, receive support, and access resources for mental health. The meditation timer allows users to set a timer for their meditation sessions, helping them to reduce stress and anxiety. The task manager helps users to organize their daily tasks, set reminders, and stay on top of their priorities.

With the combination of these features, users can manage their mental health more effectively, stay organized, and improve their overall well-being. The app's user-friendly interface and intuitive design make it easy for users to navigate and access the features they need. By using this app, users can take control of their mental health and develop healthy habits to support their overall well-being.

II. LITERATURE SURVEY

The article "Mental Health Assessment using Smartphone Sensors and Machine Learning: A Review" published in the IEEE Sensors Journal in 2019, presents an overview of the potential of using smartphone sensors and machine learning for mental health assessment. The authors, Bhattacharya and Basu, explore the various ways in which sensors embedded in smartphones can be used to capture physiological, behavioral and environmental data, which can be used for the assessment of mental health. They also discuss the application of machine learning techniques for analyzing the data and deriving insights that can aid in the early detection, diagnosis and treatment of mental health disorders. The article highlights the promising role of smartphones in the future of mental health care and suggests avenues for future research in this field.The article "Investigating persuasive technologies for mental health and wellbeing" published in IEEE Pervasive Computing explores the potential of persuasive technologies for promoting mental health and wellbeing. The authors discuss various persuasive technologies, such as mobile apps, wearable devices, and virtual reality, and how they can be utilized for mental health interventions. The article also highlights the importance of user-centered design in developing persuasive technologies, as well as the need for rigorous evaluation to determine their effectiveness. The authors conclude that while persuasive technologies havethe potential to be effective in improving mental health and wellbeing, there is still a need for more research and development in this area to ensure that these technologies are designed and evaluated in a responsible and ethical manner. A Systematic Literature Review on Chatbots and Conversational Agents for Mental Health ,This paper presents a systematic literature review on the use of chatbots and conversational agents for mental health. It explores the conceptualization and development of these technologies, highlighting design considerations and their integration with existing mental health services. The review encompasses various applications, including depression screening, psychoeducation, therapy delivery, and crisis intervention. It analyzes methodologies, data collection processes, and evaluation techniques used in the selected studies. Ethical considerations, privacy concerns, and user acceptance factors are also discussed. The review concludes by emphasizing the potential of chatbots and conversational agents in enhancing mental health interventions and suggests future research Mental Health Web App directions.

Overall, it provides a comprehensive overview, synthesizing existing knowledge and offering insights for researchers, practitioners, and developers in the mental health technology field.

III. PROPOSED METHOD

A. Data Collection

In the data collection phase, the aim is to gather a large dataset of user queries and their corresponding intents. This dataset is crucial for training the chatbot to understand and respond appropriately to user inputs. To collect the dataset, various methods can be employed. Surveys can be conducted where users are asked to provide queries related to specific topics or scenarios. These surveys can be administered online or in-person, depending on the target user group. Another approach is to directly engage with users through chat interfaces or customer support interactions, where their queries and intents can be recorded. Additionally, publicly available data sources can be scraped to gather relevant queries and intents. These sources can include online forums, social media platforms, or question- and-answer websites. By extracting data from these sources, a diverse range of user queries can be obtained, covering different topics and scenarios. The goal of data collection is to ensure that the dataset is representative of the target user group. It should encompass the various types of queries and intents that the chatbot is expected to handle. By collecting a comprehensive dataset, the chatbot can be trained on a wide range of inputs, enabling it to understand and respond accurately to user queries in real- world scenarios.

B. Data Preprocessing

Data preprocessing is an important step in preparing the collected dataset for training the chatbot model. Let's dive into the details of each preprocessing step:

Tokenization: Tokenization involves breaking down the text data (user queries) into individual words or tokens. This is done using the nltk.word_tokenize() function from the Natural Language Toolkit (NLTK) library. The function splits the text into a list of tokens based on spaces and punctuation marks.

Lemmatization and Lowercasing: After tokenization, the words in the queries are lemmatized and converted to lowercase. Lemmatization reduces words to their base or dictionary form, enabling better generalization by treating different forms of the same word as a single entity. For example, "running" and "runs" would both be lemmatized to "run". Lowercasing all the words ensures consistency and avoids treating words with different cases as separate entities.

Removing Duplicates and Sorting: To create a unique and organized set of words, duplicates are removed from the list of words obtained after tokenization and lemmatization. This step ensures that each word appears only once in the vocabulary. Additionally, sorting the vocabulary alphabetically provides a consistent ordering for the words, which can be helpful in various stages of the chatbot development process.

Bag-of-Words Matrix: Once the vocabulary is obtained, a bag-of-words matrix is constructed. The matrix represents the presence or absence of each word in each query. Each row in the matrix corresponds to a query, and each column represents a word in the vocabulary. The values in the matrix indicate whether a particular word is present in a specific query or not. This binary representation of the words allows the model to learn patterns based on the presence or absence of specific words in the queries.

By performing these preprocessing steps, the collected dataset is transformed into a suitable format for training the chatbot model. Tokenization breaks down the text into individual words, lemmatization reduces words to their base forms, removing duplicates and sorting create a unique vocabulary, and constructing the bag-of-words matrix represents the presence or absence of words in each query. These steps help in improving the effectiveness of the subsequent model training and enable the chatbot to better understand and respond to user queries.

TABLE 1

Dataset (Before Preproce ssing)	Tokeniza tion	Lemmati zation & Lowerca sing	Duplicat es Remove d	Sorted Vocabul ary	Bag-of- Words Matrix
How are you?	[How, are, you, ?]	[how, be, you, ?]	[how, be, you]	[be, how, you]	[1, 1, 1]
What is the weather today?	[What, is, the, weather, today, ?]	[what, be, the, weather, today, ?]	[what, be, the, weather, today]	[be, the, today, weather, what]	[1, 1, 1, 1, 0]
Can you help me?	[Can, you, help, me, ?]	[can, you, help, me, ?]	[can, you, help, me]	[can, help, me, you]	[1, 1, 1, 1]

C. Model training

Model training is a crucial step in building a chatbot, and it involves training a neural network model using the preprocessed dataset. Here's a detailed explanation of the model training process:

Model Architecture: The neural network model is typically built using the Keras Sequential class, which allows for stacking layers on top of each other. The model architecture consists of an input layer, one or more hidden layers, and an output layer.

The input layer receives the preprocessed data, which is the bag-of-words matrix representing the presence or absence of words in each query. The hidden layers are responsible for learning patterns and extracting meaningful representations from the input data. The number of hidden layers and the number of neurons in each layer depend on the complexity of the problem and the size of the dataset. Experimentation and optimization techniques such as cross-validation can help determine the optimal architecture.

The output layer has a number of neurons equal to the number of intent classes. Each neuron in the output layer represents a specific intent, and the output values indicate the model's confidence or probability for each intent.

Model Compilation: Before training the model, it needs to be compiled with specific settings. One important aspect is the choice of optimizer, which determines how the model learns and adjusts its weights during training. In this case, the SGD (Stochastic Gradient Descent) optimizer is used with the Nesterov accelerated gradient variant. Nesterov accelerated gradient helps the model converge faster and find better solutions.

Another crucial component is the loss function. For multi- class classification tasks like intent classification, the categorical cross-entropy loss function is commonly used. This loss function measures the difference between the predicted intent probabilities and the actual intent labels. The goal of the model is to minimize this difference during training.

Model Training: The model is trained using the model.fit() function in Keras. This function takes the preprocessed dataset as input and performs the training process.

Training involves iteratively presenting the training data to the model, computing the predicted intents, comparing them with the actual intents, and updating the model's weights to minimize the loss function. The number of trainingiterations is determined by the number of epochs. Each epoch represents a complete pass through the entire training dataset. Training for more epochs allows the model to learn more from the data, but too many epochs can lead to overfitting. The batch size determines the number of samples processed before the model's weights are updated. Training with mini- batches instead of individual samples can help improve training efficiency and generalization.

During the training process, the model learns to recognize patterns in the preprocessed data and make accurate predictions of the intents for user queries. By adjusting the weights based on the optimization algorithm and loss function, the model optimizes its performance over time. The training process continues until the specified number of epochs is reached.

It's important to note that model training can be an iterative process that involves experimentation, hyperparameter tuning, and monitoring the model's performance on validation data to achieve the best results.

D. Model Evaluation

Evaluation Dataset: An evaluation dataset is prepared, which consists of a separate set of user queries and their corresponding intents. This dataset should be distinct from the training dataset to ensure a fair evaluation of the model's generalization capability.

The evaluation dataset should contain a representative sample of user queries, covering a range of topics and scenarios. It helps evaluate how well the model performs on unseen data.

Predicting Intents: The evaluation dataset is fed into the trained model to predict the intents for each query.

The model takes the preprocessed representation of the query as input and produces a probability distribution over the intent classes. The intent with the highest probability is considered as the predicted intent.

Performance Metrics: Several metrics can be used to evaluate the performance of the model. Commonly used metrics include accuracy, precision, recall, and F1-score.

Accuracy: It measures the overall correctness of the model's predictions. It is calculated as the ratio of the number of correctly predicted intents to the total number of queries in the evaluation dataset.

Precision: It measures the proportion of correctly predicted positive intents out of all intents predicted as positive. Precision is calculated as the ratio of true positive predictions to the sum of true positive and false positive predictions. It indicates the model's ability to avoid false positives.

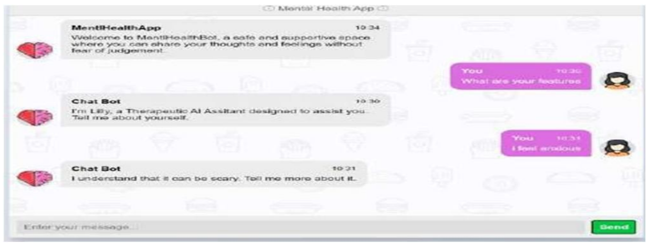
Recall (Sensitivity): It quantifies the model's ability to correctly identify positive intents. Recall is calculated as the ratio of true positive predictions to the sum of true positive and false negative predictions. It reflects the model's ability to avoid false negatives.

F1-score: It combines precision and recall into a single metric that provides a balanced measure of the model's performance. F1-score

is the harmonic mean of precision and recall and is calculated as 2 * (precision * recall) / (precision + recall). F1- score is useful when both precision and recall are important.
These metrics provide insights into different aspects of the model's performance, such as its accuracy, ability to avoid false positives and false negatives, and overall balance between precision and recall
Interpreting Results: The evaluation metrics are analyzed to assess the model's performance. A high accuracy, precision, recall, and F1-score indicate good performance, while lower scores may suggest areas for improvement.
It's important to consider the specific requirements of the chatbot application and the importance of each metric. For example, if false positives are more critical, precision may be of higher importance. Alternatively, if false negatives are more problematic, recall may be prioritized.
By evaluating the model's performance on an independent dataset, you can gain insights into its effectiveness in predicting intents and make informed decisions about its readiness for deployment or the need for further improvements.

E. Chatbot Implementation

Flask Web Framework: Flask is a Python web framework that allows you to create web applications. It provides tools and libraries to handle routing, request handling, and response generation. The chatbot implementation utilizes Flask to create a web application that serves as the front-end for users to interact with the chatbot. Flask allows you to define routes, which are URLs that users can access to interact with different parts of the chatbot application. Loading Trained Model and Intents: The pre-trained model, which was trained using the methodology explained earlier, is loaded into the chatbot application. Additionally, the intents and training data are loaded from a JSON file. This file contains information about different intents, including patterns that users might input and their corresponding responses. Loading the model and intents allows the chatbot to understand user input, predict the intent of the input, and generate appropriate responses.
Preprocessing User Input: When a user interacts with the chatbot, their input needs to be preprocessed before it can be passed to the trained model for prediction.
Preprocessing involves tokenizing the user's input by breaking it down into individual words or tokens, similar to the preprocessing done during the training phase.
The input is then lemmatized and converted to lowercase to ensure consistency in word representation.
Predicting Intent: The preprocessed user input is passed through the trained model to predict the intent.
The model takes the preprocessed representation of the user's input as input and produces a probability distribution over the intent classes.
The intent with the highest probability is considered the predicted intent. Matching Intent with Responses: Once the intent is predicted, it needs to be matched with the appropriate response. The intents file, loaded earlier from the JSON file, contains patterns and corresponding responses for each intent. The pattern associated with the predicted intent is used to identify the appropriate response from the intents file.
Generating Response: Based on the matched response, the chatbot generates a suitable response to be sent back to the user. This response can be a simple text message or a more complex output, depending on the specific requirements of the chatbot application.
User Interaction via Web Application: With the chatbot implementation in place, users can interact with the chatbot through the web application created using the Flask framework.The web application typically consists of HTML templates that define the user interface for interacting with the chatbot. Users can enter their queries or messages in a text input field, and the application sends the input to the chatbot for processing. The generated response from the chatbot is then displayed back to the user through the web application.
By implementing the chatbot using the Flask web framework, Keras neural network library, and NLTK library, you create a user-friendly interface that allows users to interact with the chatbot and receive appropriate responses based on their input.



MentlHealthApp 10:34
Welcome to MentlHealthBot, a safe and supportive space where you can share your thoughts and feelings without fear of judgement.

You 10:30
What are your features

Chat Bot 10:30
I'm Lilly, a Therapeutic AI Assitant designed to assist you. Tell me about yourself.

You 10:31
I feel anxious

Chat Bot 10:31
I understand that it can be scary. Tell me more about it.

Fig 1 - user interface

F. User Testing

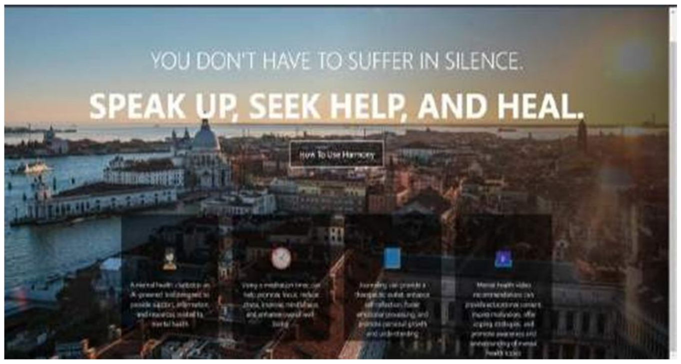

YOU DON'T HAVE TO SUFFER IN SILENCE.
SPEAK UP, SEEK HELP, AND HEAL.
How To Use Harmony

fig 2-front page

Selecting a Group of Users: A diverse group of users is selected to participate in the user testing phase. The users should represent the target user group for whom the chatbot is designed.
It's important to include users with different backgrounds, knowledge levels, and preferences to gather a comprehensive range of feedback.
Interacting with the Chatbot: Users are provided with an opportunity to interact with the chatbot and use it as they would in real-world scenarios. They can input queries, ask questions, or engage in conversations with the chatbot. The chatbot responds to user inputs by predicting intents and generating appropriate responses based on the implemented logic.
Collecting User Feedback: During or after the interaction, users are encouraged to provide feedback on their experience with the chatbot.
Feedback can be collected through various means, such as surveys, interviews, or feedback forms. Users are asked to evaluate the chatbot's accuracy in understanding their queries and providing relevant responses. They may also be asked to rate the overall user experience, interface design, and ease of use.
Analyzing User Feedback: The collected user feedback is analyzed to identify patterns, common issues, and areas for improvement. Feedback is categorized based on different aspects of the chatbot's performance and user experience, such as accuracy, relevancy of responses, and usability. By analyzing the feedback, specific strengths and weaknesses of the chatbot can be identified.
Making Necessary Changes and Enhancements: Based on the analyzed feedback, necessary changes and enhancements are made to improve the chatbot. Common issues and areas for improvement are addressed, such as refining the training data, improving the model's performance, or enhancing the user interface. Changes may involve updating the training dataset, retraining the model, adjusting the response generation logic, or making modifications to the user interface.
Iterative Testing and Improvement: User testing is an iterative process, and multiple rounds of testing and improvement may be conducted. The updated version of the chatbot is tested again with a new group of users to assess the impact of the changes and gather additional feedback.
This iterative approach allows for continuous refinement of the chatbot's performance and user experience. By conducting user testing and incorporating user feedback, the chatbot can be optimized to meet user expectations, provide accurate responses, and deliver a satisfactory user experience.

IV. ARCHITECTURE

1) *Input Layer:* The input layer is the starting point of the model and receives the preprocessed input data. The number of neurons in the input layer is determined by the size of the vocabulary, representing the number of unique words in the dataset. Each neuron in the input layer corresponds to a word in the vocabulary, and its value represents the presence or absence of that word in the input query.
2) *Hidden Layers:* The hidden layers are responsible for learning patterns and representations from the input data. The number of hidden layers and neurons in each layer depend on the complexity of the problem and the size of the dataset. Commonly used activation functions in the hidden layers include Rectified Linear Unit (ReLU) and hyperbolic tangent (tanh), which introduce non-linearity and enable the model to capture complex relationships between words and intents.

3) *Output Layer:* The output layer produces the final predictions of the model, indicating the intent of the user query. The number of neurons in the output layer is equal to the number of intent classes or categories. Each neuron represents a specific intent, and the output values from these neurons are transformed into probabilities using a suitable activation function, such as softmax. The intent with the highest probability is considered the predicted intent of the input query.
4) *Model Compilation:* Before training the model, it needs to be compiled with specific settings.The optimizer, such as Stochastic Gradient Descent (SGD) with Nesterov accelerated gradient, determines how the model learns and adjusts its weights to minimize the loss function.The loss function, typically categorical cross-entropy, measures the difference between the predicted intents and the true intents in the training data.Additional metrics, such as accuracy, can be specified to monitor the model's performance during training.
5) *Model Training:* The compiled model is trained using the preprocessed dataset.Training involves feeding the input queries and their corresponding intents to the model and adjusting the weights iteratively based on the calculated loss.The number of epochs specifies the number of iterations over the entire dataset during training.The batch size determines the number of samples processed before updating the weights, and it affects the speed and stability of the training process.By designing the model architecture with appropriate layers, activation functions, and optimization settings, the chatbot model can effectively learn the underlying patterns in the input queries and make accurate predictions about user intents. The training process enables the model to adjust its weights based on the provided dataset, improving its ability to understand and respond to user queries.

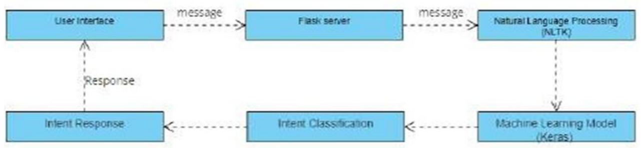


Fig 2- architecture

V. FUTURE SCOPE

Mobile app: A mobile app could provide greater convenience and accessibility to users, making it easier for volunteers and organizations to connect. The app could include features like push notifications to remind volunteers of upcoming events or tasks, a GPS system to help volunteers locate nearby opportunities, and an easy-to-use interface for both volunteers and organizations to communicate and manage tasks. Additionally, a mobile app could be designed to be compatible with different operating systems and devices, allowing a wider range of users to participate.

Social media integration: By integrating social media features, such as sharing, commenting, and liking, the project could be promoted and shared more widely on social media platforms. This could increase the visibility of the project and attract more volunteers and organizations to participate. Social media integration could also allow for more efficient communication between volunteers and organizations, making it easier to share updates and progress reports.

Gamification: Gamification elements like leaderboards, badges, and rewards can make volunteering more engaging and fun, and incentivize volunteers to participate more frequently and complete more tasks. Gamification could also foster a sense of competition and community among volunteers, as they strive to achieve the highest rankings and earn the most rewards.

Machine learning: Machine learning algorithms could help to improve the accuracy and efficiency of the matching process between volunteers and organizations. By analyzing the skills, interests, and preferences of volunteers, machine learning algorithms could suggest the best matches for each volunteer, increasing the likelihood of a successful partnership. Machine learning could also be used to predict which tasks are most likely to be completed successfully and which volunteers are most likely to complete those tasks, optimizing the matching process even further.

Multi-language support: Adding support for multiple languages could make the project more accessible to non- English speakers and allow it to reach a more diverse audience. This would involve creating language-specific interfaces for both volunteers and organizations, as well as providing support for different character sets and text directions. Providing multi-language support could also help to create a more inclusive and welcoming environment for volunteers and organizations from different cultural backgrounds.

Analytics and reporting: Incorporating analytics and reporting features could help organizations track the impact of the project and identify areas for improvement. This could include tracking the number of volunteers, the types of tasks completed, and the overall impact of the project. By analyzing this data, organizations could identify which tasks are most in demand, which areas need more attention, and which volunteers are most successful, allowing them to refine and improve the project over time.

VI. CONCLUSION

In conclusion, mental health trackers can be a useful tool for individuals who want to monitor and improve their mental health. They can provide valuable insights into mood patterns, triggers, and behaviors, allowing users to identify areas for improvement and develop coping strategies. Additionally, mental health trackers can facilitate communication with mental health professionals, providing them with valuable information that can inform treatment decisions. However, it's important to remember that mental health trackers should not be used as a substitute for professional medical advice or treatment. They can be a useful addition to a mental health treatment plan, but should be used in conjunction with therapy and other forms of support. Ultimately, the effectiveness of a mental health tracker will depend on how consistently and accurately it is used, and whether it is integrated into a comprehensive mental health plan.

VII. ACKNOWLEDGMENT

We are very thankful to the Department of Computer Science and Engineering of Adi Shankara Institute of Engineering and Technology for permitting us to work on the topic “Mental health web app”. We express our gratitude to prof. Rosemary Vargeehese., our project guide who gave us all the guidance and motivation and at any time available to us for doubts solving, grateful for giving us the required help, point by point recommendations and furthermore support to do the venture. We are excited and happy to offer our thanks to the Head of the Information Technology Department Prof. Manesh T I., for his endorsement of this undertaking. We are amazingly appreciative to all staff and the administration of the school for giving every one of us theoffices and assets required.

REFERENCES

[1] S. Taylor, The psychology of pandemics, First Edit. UK: Cambridge Scholar Publishing, 2019.
[2] J.jin Zhang et al., “Clinical characteristics of 140 patients infected with SARS-CoV-2 in Wuhan, China,” Allergy Eur. J. Allergy Clin. Immunol., vol. 75, no. 7, pp. 1730–1741, 2020.
[3] I. I. Haider, F. Tiwana, and S. M. Tahir, “Impact of the COVID- 19 pandemic on adult mental health,” Pakistan J. Med. Sci., vol. 36, no. COVID19-S4, pp. S90–S94, 2020.
[4] J. Shigemura, R. J. Ursano, J. C. Morganstein, M. Kurosawa, and D. M. Benedek, “Public responses to the novel 2019 coronavirus (2019- nCoV) in Japan: Mental health consequences and target populations,”Psychiatry Clin. Neurosci., vol. 74, no. 4, pp. 281–282, 2020.
[5] C. S. Ho, C. Y. Chee, and R. C. Ho, “Mental Health Strategies toCombat the Psychological Impact of COVID-19 Beyond Paranoia andPanic,” Ann. Acad. Med. Singapore, vol. 49, no. 1, pp. 1–3, 2020.
[6] W. Kawohl and C. Nordt, “COVID-19, unemployment, and suicide,”The Lancet Psychiatry, vol. 7, no. 5, pp. 389–390, 2020.
[7] K. S. Khan, M. A. Mamun, M. D. Griffiths, and I. Ullah, “The Mental Health Impact of the COVID-19 Pandemic Across Different Cohorts,” Int. J. Ment. Health Addict., 2020.
[8] L. Tang, B. Bie, S. E. Park, and D. Zhi, “Social media and outbreaks of emerging infectious diseases: A systematic review of literature,” Am. J. Infect. Control, vol. 46, no. 9, pp. 962–972, 2018.
[9] T. M. Huang, “Design and implementation of APP system for legal consulting based on Java technology,” Procedia Comput. Sci., vol. 166, pp. 99–103, 2020.
[10] M. J. Dutta, S. Kaur-gill, N. Tan, and C. Lam, “mHealth , Health , and Mobility : A Culture-Centered Interrogation Á Community Á Neoliberalism Á Mobile health,” mHealth Innov. Asia, pp. 91–107, 2017.