AN INTELLIGENT SOCIAL-BASED ASSISTANT APPLICATION FOR STUDY TIME MANAGEMENT USING ARTIFICIAL INTELLIGENCE AND NATURAL LANGUAGE PROCESSING

Haoyu Li[1], Ryan Yan[2] and Ang Li[3]

[1]Whittier Christian High School, 501 N Beach Blvd, La Habra, CA 90631
[2]Cal Poly Pomona, 3801 West Temple Avenue, Pomona, California 91768
[3]California State University, Long Beach,
1250 Bellflower Blvd, Long Beach, CA 90840

ABSTRACT

The question that we aim to solve is "How will elderly people be able to increase their productivity and remember their tasks?" There are many ways to go about answering this question, but we have devised a simple solution to this question that can directly and quickly have a positive impact on these individuals. Our method to solving this is creating a to-do list in Flutter, which will allow elderly people to have easy access to a list of tasks that they have to complete [5].

Some predicted results of this to-do list is that it can raise productivity for its users. Our to-do list features a ChatBot, which talks to the user through a message-like system in order to prompt user input for specific details such as the time, date, and main description of the task [6]. Then, the ChatBot will take in all this information to produce a clean and concise final task description that takes keywords from the user-inputted description. This provides users of our to-do list with an alternative method of adding tasks, which may be greatly appreciated by those who are less able-bodied or struggle to type. By offering the elderly a way of adding tasks that can take less typing, these individuals may rely on this to-do list as a great convenience to their lives.

KEYWORDS

Flutter, To-Do List, Tasks, Productivity.

1. INTRODUCTION

Our topic is creating a to-do list with Flutter, and this to-do list can create tasks to remind people of what to do each day. This to-do list can be helpful to anyone of all ages, as long as they have a smartphone or a similar portable electronic device that is compatible with Flutter. However, this to-do list is mainly directed towards those who are not as able-bodied, such as senior citizens. Benefits of the Flutter to-do list include providing people with an intuitive and easy-to-use application for storing tasks to be completed later. Some to-do lists that currently exist may be harder to use and might make people frustrated when trying to create and handle their to-do list. Therefore, making a to-do list that is more user-oriented will allow people to live their day-to-day

David C. Wyld et al. (Eds): DMML, SEAS, ADCO, NLPI, SP, BDBS, CMCA, CSITEC - 2022
pp. 161-171, 2022. CS & IT - CSCP 2022 DOI: 10.5121/csit.2022.120714

lives more conveniently. There is little to no negative consequence for having this to-do list, besides taking extra memory on the device.

Our topic is significant because it allows elderly people to live more productive lives and brings them a convenient way to remember their tasks [7]. Some elderly people may have poor memory, and a to-do list would provide them with a method of remembering what they need to accomplish. A setback that elderly people may come across is that they may have less capable bodies, or they may not be great at typing. This to-do list seeks to improve the lives of these individuals by using keywords in order to retrieve the main ideas of these tasks and delivering them to the users.

There are some existing to-do lists on mobile device application stores that have aimed to achieve the same general goal as we have, which is to improve productivity among our application's users. These to-do lists, for the most part, have the same general format. This format includes a checklist with functioning check boxes, which include the list of tasks that still need completing [8]. The users will press a button that prompts them to type in text that describes what task they will need to complete in the future, and they will check the boxes next to the corresponding task once they have completed it to cross the tasks out.

Although these existing to-do lists may not necessarily have glaringly large issues in the eyes of the general public, those who are less-able bodied may struggle to use these applications. This is because the vast majority of these to-do lists involve having to type out the entire task in order to save it onto the to-do list. However, for those who have poor typing skills or are otherwise unable to easily type, they will likely have a difficult time using these to-do lists. This concern mainly applies with the elderly, whose joints may not work as well as before due to arthritis or other medical conditions. Therefore, an alternative way to add tasks that requires less typing or physical ability is a necessity for some of these individuals. Another issue that lies with some to-do lists is the inability to save tasks after closing the application. The tasks that are typed in are only saved for that specific instance of the application. Most people want their tasks to be saved for the next time they reopen the application, considering that the tasks they typed in were tasks that still needed completing. Saving tasks from the previous session every time the application is opened is a crucial part of a functioning to-do list.

Our to-do list admittedly shares many similarities with other to-do lists that currently exist in mobile application stores. However, our to-do list differs due to having a login and register system. Our to-do list uses Firebase, which is a database that has the capabilities of saving usernames and passwords [9]. The user can use our application by first creating an account, then logging in with this account. Most existing to-do lists do not have a system like this, and the tasks are only saved to the corresponding device. However, with the login and register system of our file, the tasks that are saved to a specific account can be transferred over to another device, as long as the user is using the same account to log in with. This capability provides our to-do list with more flexibility than most others.

Another feature of our to-do list that most existing to-do lists do not have is a ChatBot [10]. This ChatBot is in a second tab of our application, which prompts the user in a message-like system for input on their task. First, it will ask for the user-inputted description of the task. Then, the ChatBot will ask for the date and the time of when the task needs to be completed. Finally, the ChatBot will take keywords from the user-inputted task description and combine it with the date and time, then ask for confirmation from the user if they want to add ChatBot's task to their to-do list. After the user agrees and responds with "yes", the task is successfully added. This provides a more intuitive way for users who may not be as adept in using technology, such as elderly individuals.

We proved our results about the effectiveness of the to-do list by performing two types of experiments. One of these experiments involved testing out the IBM Watson Assistant and recording how accurate it was when extracting the main keywords from a user-inputted description and including it into the final description. In order to have the measurement of accuracy tested more fairly, we had 10 participants download the application and experiment with using the ChatBot at least 20 times each, and we urged them to use as much variety in the test descriptions as possible. Each person would record how many times they had tested this feature in total and how many times the IBM Watson Assistant would properly extract the necessary keywords out of these test descriptions.

The second experiment involved testing their overall productivity boost from using the app. Each of the aforementioned participants would give themselves a rating from 1 to 10 that measured how productive they felt during that week. Then, they would use the to-do application for all of their needs during the next week and record their rating from 1 to 10 of how productive they felt after one week of using the application. Furthermore, each participant would state whether they used the ChatBot feature at all in their day-to-day lives and whether they preferred using the ChatBot to add tasks over the traditional method of typing the task description on their own. This experiment would attempt to measure how effective this application was at improving productivity in its users.

The rest of the paper is organized in sections from 2 to 5. Section 2 dives into the details of what challenges we faced during the process of creating our to-do list application. Section 3 covers a general overview of the solution we proposed to boost the productivity of individuals, as well as each detail and step that was taken to reach the final product. Section 4 describes the experiments performed to prove the effectiveness of the to-do list application. Section 5 goes over a brief summary of three related works and how they compare to this paper. Lastly, Section 6 provides concluding remarks to summarize our paper and consider what could be done in the future regarding this project.

2. CHALLENGES

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

2.1. Lack of knowledge in Flutter

Our first challenge was a lack of knowledge in Flutter. It was the first real experience that we had with using Flutter and Dart, so being able to create a full-fledged application with not only basic to-do list compatibility but also adding a login and register system was a massive accomplishment [11]. At first, we struggled greatly when trying to wrap our head around the Flutter framework, and writing code seemed to involve a lot of unnecessary indenting and formatting here and there that made it difficult to understand what our Dart code was trying to accomplish at its core. However, as we persevered and studied Dart code closer, we were able to figure out the purpose of each line of code and were able to write our own code that could successfully perform to-do list functions and even include our own features to the application. we will be able to use the knowledge and experience from overcoming this challenge in future projects.

2.2. Figuring out how to approach the topic

Our second challenge was figuring out how we could approach the topic of to-do lists in a unique way. When we looked at other papers to figure out how we should write our own, all we saw

were good ideas that were all being taken. Other papers focused on creating a group to-do list that could be accessed and modified by multiple people or comparing the efficiency of electronic to-do lists and paper to-do lists. In order to come up with our own emphasis on to-do lists, we had to dig deeper and think about what we personally valued the most in a to-do list. The answer we came up with was that we wanted a to-do list that could be as convenient and helpful as possible for the user. We are very pleased that this was the concept we chose to focus on, as it was something that we felt passionate about.

2.3. Finding ways to send a signal from a watch to the server

Our third challenge was finding a way to integrate Python code into our Flutter project [12]. Our goal with this to-do list application was to add a ChatBot as an additional option for adding tasks. This ChatBot involved the IBM Watson Assistant, which would help in extracting the necessary keywords from user-inputted descriptions. However, the only way we knew to use this ChatBot was through Python. Therefore, we needed a method to fuse these two different programming languages into one coherent and functional application. After searching online for a while, we discovered that this was possible through integrating the Python code. We progressed steadily with our project, using Flutter primarily as our front-end code and using a few small Python files to assist with back-end code. We would not have learned about such a useful Flutter skill if we had not thought to include this feature into our to-do list application.

3. SOLUTION

The whole system of our program works by running a Flutter application [13]. Most of the application's code is made up of Dart files, which control the formatting and functionalities of the application. The application uses multiple files that control both the application's front-end and back-end. Firebase is used as the database to store the users' usernames and passwords as well as the tasks in their to-do lists. With Firebase, a login system is used to store a user's to-do list and tasks. Furthermore, the Flutter application has Python code integrated into it, allowing even more functionality. This Python integration was specifically added to include the feature of a ChatBot. With the help of the IBM Watson Assistant, the ChatBot gives the user a second way of adding a task to the to-do list with a series of questions that require short answers. First, the user taps on the ChatBot tab to enter a tab that has a user interface that is similar to a messaging app. Then, after a second of staying on the ChatBot tab, the ChatBot will send a message to the user, prompting the user to send a description of the task. The ChatBot will prompt the user for a date once the user sends the description as a message, then the ChatBot will prompt the user for a time once the user sends the date as a message. If the user had added any information for the next prompt in a previous prompt, such as including the time along with the date when prompted for just the date, the ChatBot will take this into account by taking in both the date and time and skipping the prompt asking for the time. Finally, the ChatBot will ask for confirmation from the user. If the user accepts, the task is added to the to-do list successfully.

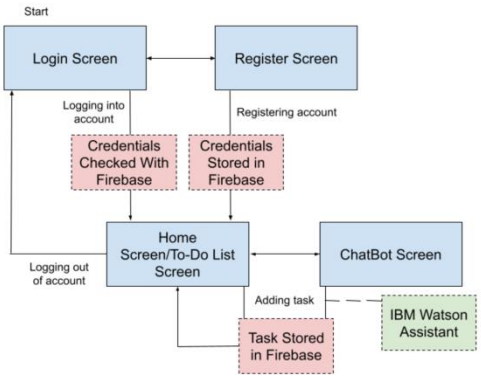

Figure 1. Overview of the system

The database was a significant part of the application, as it was what stored both the login credentials and the tasks in the to-do list that corresponded to each account. We decided to use Firebase as the database due to its compatibility with Flutter and its cost (free to use below a certain threshold of data stored). The database was implemented by first importing the Firebase libraries into the relevant Dart files. In the main Dart file, the application would first initialize Firebase before running. Firebase would handle the login, register, and logout processes for the application as well as adding tasks to the to-do list that corresponded with the user's ID. The logging in, registering, and logging out was handled in a Dart file that created an instance of the Firebase authentication service and called functions from the instance of this service to perform these actions. In the case of logging in and registering, the email and password entered into the text field were used as parameters for the corresponding functions. Registering saves the email and password combination to Firebase.

```dart
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();

  runApp(MyApp());

}
```

```dart
import 'package:firebase_auth/firebase_auth.dart';

class AuthService {
  final FirebaseAuth _firebaseAuth = FirebaseAuth.instance;
  Myuser? _userFromFirebase(User? user) {
    if (user == null) {
      return null;
    }
    return Myuser(user.uid, user.email!);
  }

  Stream<Myuser?> get user {
    return _firebaseAuth.authStateChanges().map(_userFromFirebase);
  }

  Future<Myuser?> signinWithEmailandPassword(
      String email, String password) async {
    print(email);
    final credential = await _firebaseAuth.signInWithEmailAndPassword(
        email: email, password: password);

    return _userFromFirebase(credential.user);
  }

  Future<Myuser?> createWithEmailandPassword(
      String email, String password) async {
    print(email + password);
    final credential = await _firebaseAuth.createUserWithEmailAndPassword(
        email: email, password: password);

    return _userFromFirebase(credential.user);
  }

  Future<void> signOut() async {
    return await _firebaseAuth.signOut();
  }
```

Figure 2. Screenshot of code 1

Another major component of the application is the ChatBot. Unlike the rest of the code in the application, ChatBot uses Python as its programming language. In order to integrate the Python code, we imported an API called Flask in one of the Python files. In this Python file, we made

two functions, one of them for creating a ChatBot session and the other one for sending messages in the session. For each of these functions, we used "@app.route", which allowed this code to be included in the functionality of the application. The ChatBot worked by importing the IBM Watson Assistant in a Python file and setting up the service for this assistant through an API key. The ChatBot would start a session when the user tapped on the ChatBot tab to enter the ChatBot screen, and the ChatBot would send messages by asking about the task description, date, and time. After going through the task adding process with the user and having the user confirm the task description, the ChatBot would send a final message as a visual notice to the user that the task was successfully added. As each message sent by the ChatBot is checked for whether it includes the string "I would schedule", it would then notice that the final message has been sent and call a function from Firebase that adds a new task with the full task description that is shown in the second half of the ChatBot's final message.

```
from flask import Flask
import ibm_chatbot as bot
import json

app = Flask(__name__)

# Create session.
@app.route("/create_session/<user_id>")
def create_session(user_id):
    response = {}
    response ['session_id'] = bot.create_session()
    bot_init_message = bot.get_response(message='',session_id=response ['session_id'],user_id=user_id)
    response.update(bot_init_message)
    return response



@app.route("/send_message/<message>/<session_id>/<user_id>")
def send_message(message,user_id,session_id):
    response = bot.get_response(message=message,session_id=session_id,user_id=user_id)
    return json.dumps(response)

app.run(host='0.0.0.0')
```

```
print(response)
if response['output']['generic']:
    for generic in response['output']['generic']:
        text = generic['text']
        res['response'] += text + '\n\n'
        if 'I would schedule' in text:
            start_index =  len('I would schedule ')
            firebase.set_context(text [start_index:], user_id)
            res['end_session'] = True
return res
```

Figure 3. Screenshot of code 2

There are four main screens of the application: the login screen, the register screen, the home screen, and the ChatBot screen. The home screen includes the to-do list with tasks. The screens were implemented by creating Dart files for each screen, in which the Dart files had code that both formatted the layout of these screens and implemented functional buttons that either brought the user to another screen or performed some sort of action. When the user opens the application for the first time, they will start at the login screen. From here, the user could enter the credentials of an existing account and tap the Login button to enter the home screen. The user could also tap the Register button to switch to the register screen, create the new account by filling in the credentials, and tap Read Content and then Back to Login to enter the home screen. These screens will also check for any errors in the user input. For example, if the two passwords are not matching in the register screen, the Dart file that handles the register screen will check for this and return the message "password not equal" when it determines that the two password text fields

hold different String values. After logging in or registering, the user can swap between the home screen with the to-do list and the ChatBot screen through a bottom navigation bar in a Dart file for routing that has separate dart files imported for both the home screen and the ChatBot screen. The check boxes in the home screen function by creating a strikethrough effect on the text, filling the check box in blue with a white check mark, and having a boolean variable switch from false to true when an unchecked checkbox is tapped. When a checked checkbox is tapped again, the text and checkbox revert, and the boolean variable becomes false again.

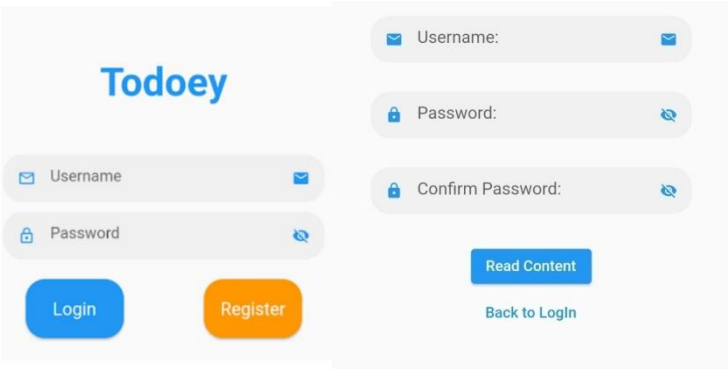

Figure 4. Screenshot of login page

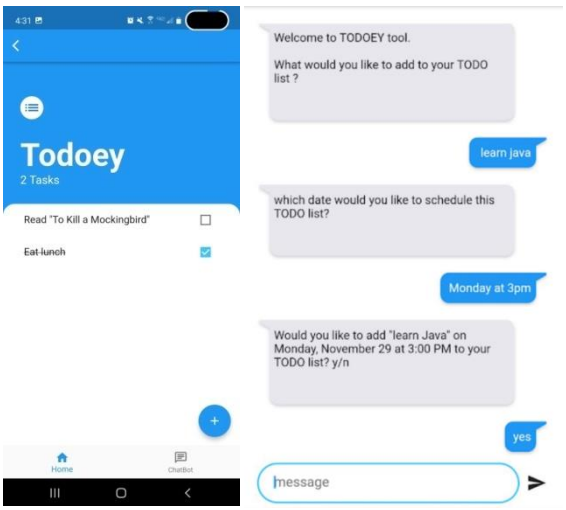

Figure 5. Screenshot of task page

4. EXPERIMENT

4.1. Experiment 1

The problem we aim to solve is creating a to-do list for users who may be slow or unable to type well, such as those who may be elderly and have issues with their joints. The solution we have devised to tackle this problem is creating a ChatBot that can potentially reduce the amount of typing and break the task description creation process down into simple steps. The experiment design involves 10 participants testing the accuracy of the IBM Watson Assistant in the ChatBot regarding the extraction of important keywords from the user-inputted descriptions at least 20 times each, which gives a decent sample size to account for variability. For each time that the IBM Watson Assistant failed to successfully extract the keywords, either through having an

awkward-sounding description or losing crucial information, the participant must provide what exactly they inputted and what exactly the ChatBot outputted.

Participant #	Number of times tested	Number of successful tests	Percentage of successful tests
1	20	20	100%
2	20	20	100%
3	22	21	95.45%
4	20	20	100%
5	21	21	100%
6	20	18	90%
7	25	24	96%
8	21	21	100%
9	20	19	95%
10	20	20	100%

Table 1. Result of Experiment 1

According to the results, the ChatBot was fairly accurate for the majority of the user inputs. However, there are still cases where some of the keywords are cut out of the final description that make the task description sound awkward. An example is when one of the participants entered the following task description: “do math”. Instead of the ChatBot including the full user-inputted description into the final task description, the ChatBot simply decided that the word “math” would be included. The participant who tested this description labeled this as an “ awkward description ”. Since the description still gets its point across without losing any crucial information, this seems to just be a minor issue. All minor issues were also labeled as “ inaccurate ” for the sake of this experiment. Throughout all the tests, there were no major issues involving the loss of any crucial information in any of the ChatBot results. Therefore, the ChatBot seems to be a very accurate and reliable alternative to adding tasks.

4.2. Experiment 2

The problem that our solution aims to solve is that people in their everyday lives, especially the elderly, may forget their tasks and become less productive than they could potentially be as a result. The solution we have designed, a to-do list application that can be accessed on smart devices, would be able to achieve this by giving people a quick and convenient way to remind themselves of their tasks. This second experiment is a survey that the participants will take after using the to-do list application for all of their needs for 7 days. Since the experiment includes 10 participants, there is enough to account for any variability. The participants are also asked at the end of the survey whether they regularly opted to use the ChatBot feature when adding their tasks.

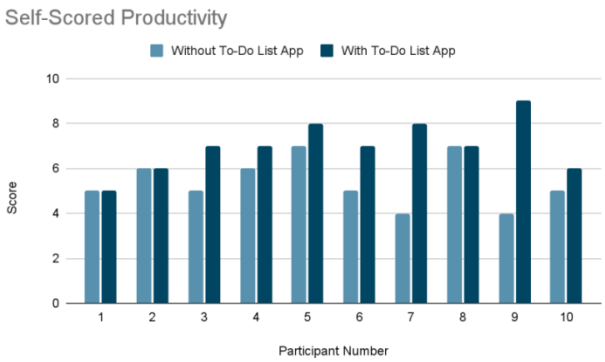

Figure 6. Self-Scored Productivity

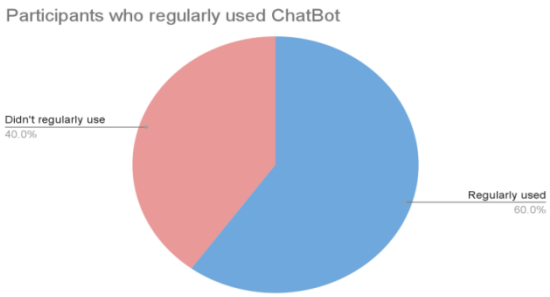

Figure 7. Participants who regularly used ChatBot

The experiment's main measurements were a self-given score of how productive they felt the week before using the to-do list application and the week after using the to-do list application. From the results that were collected from this experiment, it appears that all participants felt that they were either more productive or at the same level of productivity after a week of using this to-do list. Most of the improvements in productivity seemed to be relatively small, but there were two participants who felt that they had their productivity significantly boosted in the week after, with a significant productivity boost being considered as an improvement of 4 or greater from the previous week. Therefore, this to-do list application has been shown to boost productivity among its users. In the feedback, six out of the ten participants used the ChatBot feature regularly in their to-do list application use, indicating that the feature was a significant part of their experience with using the application.

For the first experiment, the results proved that the ChatBot is capable of being a convenient alternative to the traditional method of typing in full task descriptions. Due to being accurate at extracting keywords with the majority of user-inputted descriptions tested, the ChatBot is a fairly reliable tool when setting reminders for tasks. In the second experiment, the results showed that slightly over half of all our to-do list application users will regularly use the ChatBot application. A significant portion of the users, therefore, will treat this feature as a necessary part of the experience when using this application. Furthermore, the second experiment also proves that regular use of the to-do list application can provide, on average, a small boost in productivity to its users. None of the participants that tested the application experienced a reduction in productivity. Based on the experiment results, the to-do list application appears to increase the productivity of its users and provide the ChatBot as an alternative method to add tasks that the majority of its users will take advantage of.

5. RELATED WORK

The related work involved a study that compared the productivity of a smartphone to-do list application and paper-based lists within an intensive care unit. The study found that although the to-do list application received positive feedback from the medical staff, the application could not be proven to be more effective at increasing productivity than paper to-do lists [1]. This related work had its main emphasis on experimenting with the effectiveness of a smartphone application . However, our work focused more on the development of the application itself. Our strengths were detailing our application creation process, while the strengths of the related work were studying how the workflow within an intensive care unit was affected through the use of a to-do list application.

The main topic of the related work is using the Analytic Hierarchy Process, or AHP for short, to prioritize use cases in mobile application development [2]. AHP is a method that utilizes the main problem that is meant to be solved, every possible solution to the aforementioned problem, and the criteria that will be used to measure how preferable each solution is. Through these three factors, AHP can find the best solution and analyze complex decisions through math and psychology. [3]. The related work appears to cover how AHP is used with mobile applications in general, with to-do list applications as a small focus in part of the paper. On the other hand, our work involves exclusively to-do lists and how to make the individual user's experience of a to-do list application as convenient as possible. Overall, the related work covers a broader topic with more information, while our work has a more narrow focus.

In this related work, the to-do list that they have created was meant to appeal to families and those in group projects who were in need of a to-do list that could be edited by multiple people rather than a single person [4]. While our work focused on developing a ChatBot to create tasks for the users, the author of this work had a great emphasis on having a group to-do list as well as a monetary management system built in. This work's strength is improving to-do lists to fit a group setting and allow for collaboration. On the other hand, our work had our greatest strengths in maximizing the convenience of an individual.

6. CONCLUSIONS

The application we propose to help others against forgetfulness and to improve productivity is our to-do list [14]. We believe that having a list of tasks that can be conveniently accessed from a portable electronic device will aid users in their everyday lives. This to-do list application includes the basic features of adding tasks, checking off tasks, and viewing the whole task list [15]. Our application has two ways to add tasks, with the second way being a ChatBot that the user can communicate with through messages to potentially require less typing. Typing less will be incredibly helpful towards the elderly and others who may have poor typing skills. This ChatBot comes with the added benefit of making concise tasks that are easy to understand. Some users may type long, complicated messages to add to their to-do list, then come back and get confused about what they were trying to remind themselves about before. On the other hand, some users may type up task descriptions that are much too simple, and they may not remember important details such as the date and time that a task needed to be completed. The ChatBot aims to solve both of these issues. When the user inputs a description into the ChatBot, the IBM Watson Assistant will extract the most necessary keywords from the description to use later for the final task description. The ChatBot will also explicitly ask the user for a date and a time to ensure that the user will have a clear idea of when a task needs to be completed.

One limitation that our to-do list has is the ability to check off multiple tasks at once. At the moment, our to-do list is capable of only checking off one task at a time, and trying to check off another task will uncheck the previously checked-off task. A second limitation that our to-do list has is the ability to remove tasks. Tasks can only be added, and only one task can be checked off at a time, but there is currently no way in our to-do list to have a task completely removed. With extended use, the to-do list can begin to feel cluttered with many unneeded tasks.

In the future, we plan to solve these limitations by adding a button next to each task that removes the task rather than just checking them off. Furthermore, to solve the issue of only being able to check off one task at a time, we will search for a different method of implementing the checking off of tasks.

REFERENCES

[1] Esposito, M., Rocq, P.-L., Novy, E., Remen, T., Losser, M.-R., & Guerci, P. (2020, January 21). Smartphone to-do list application to improve workflow in an intensive care unit: A superiority quasi-experimental study. International Journal of Medical Informatics. Retrieved February 4, 2022, from https://www.sciencedirect.com/science/article/pii/S1386505619309487

[2] Yildirim, Onur, and Serhat Peker. “Prioritizing Use Cases for Development of Mobile Apps Using AHP: A Case Study in To-Do List Apps.” SpringerLink, 26 July 2019, link.springer.com/chapter/10.1007%2F978-3-030-27192-3_24.

[3] “What Is the Analytic Hierarchy Process (AHP)?” Passage Technology, www.passagetechnology.com/what-is-the-analytic-hierarchy-process.

[4] Ringgau, Diana anak, et al. “Development of to-Do List and Monetary Management System.” International ABEC, abecindonesia.org/iabec/index.php/iabec/article/view/4.

[5] Payne, Rap. "Developing in Flutter." Beginning App Development with Flutter. Apress, Berkeley, CA, 2019. 9-27.

[6] Dahiya, Menal. "A tool of conversation: Chatbot." International Journal of Computer Sciences and Engineering 5.5 (2017): 158-161.

[7] Vergados, Dimitrios, et al. "Intelligent services for assisting independent living of elderly people at home." Proceedings of the 1st international conference on PErvasive Technologies Related to Assistive Environments. 2008.

[8] Gordon, Suzanne, Patrick Mendenhall, and Bonnie Blair O'toole. Beyond the checklist. Cornell University Press, 2012.

[9] Moroney, Laurence. "The firebase realtime database." The Definitive Guide to Firebase. Apress, Berkeley, CA, 2017. 51-71.

[10] Adamopoulou, Eleni, and Lefteris Moussiades. "An overview of chatbot technology." IFIP International Conference on Artificial Intelligence Applications and Innovations. Springer, Cham, 2020.

[11] Kuzmin, Nikita, Konstantin Ignatiev, and Denis Grafov. "Experience of developing a mobile application using flutter." Information Science and Applications. Springer, Singapore, 2020. 571-575.

[12] Sanner, Michel F. "Python: a programming language for software integration and development." J Mol Graph Model 17.1 (1999): 57-61.

[13] Joorabchi, Mona Erfani, Ali Mesbah, and Philippe Kruchten. "Real challenges in mobile app development." 2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. IEEE, 2013.

[14] Kral, Vojtech Adalbert. "Senescent forgetfulness: benign and malignant." Canadian Medical Association Journal 86.6 (1962): 257.

[15] Benacerraf, Paul. "Tasks, super-tasks, and the modern eleatics." The Journal of Philosophy 59.24 (1962): 765-784.