AN INTELLIGENT ALARM CLOCK SYSTEM BASED ON BIG DATA AND ARTIFICIAL INTELLIGENCE

Leon He[1] and Ang Li[2]

[1]La Canada High School, 4463 Oak Grove Drive, Irvine, CA 92604
[2]California State University,
Long Beach, 1250 Bellflower Blvd, Long Beach, CA 90840

ABSTRACT

Sleep is a crucial part of a person's daily routine [1]. However, oversleeping is often a hindrance to many people's daily life. This paper develops an application to prevent people from oversleeping or falling back to sleep after snoozing the alarm. We applied our application to fellow students and conducted a qualitative evaluation of the approach. The results show that the application improves the chances of waking up to a significant degree.

KEYWORDS

Machine Learning, Recommendation System, Data Mining.

1. INTRODUCTION

In the current society, high expectations and workloads are held towards youngsters [2]. Students and others with a cluttered regular schedule suffer from pressure and deadlines [3][4]. Luckily, sleep can relieve such stress, improve people's mood, reduce the risk of diseases, plus other benefits like improving memory and productivity. After an exhausting day, a good night's sleep would be perfect for recovery. But sometimes, when people resort to sleeping, they snooze for a bit too long [5]. This phenomenon occurs in almost every single person's life, whether they forgot to set an alarm before falling asleep, if they are simply too tired after a long day of work and remain in deep sleep instead of waking up, or if they are just reluctant to wake-up even if the alarm rings off [6][7]. They oversleep, Oversleeping is also likely to cause depression and nausea.

Some existing methods are physical alarm clocks or applications on phones. Other methods include lighting or even smell. These methods rely on sound to stimulate people's senses. The main issue is the lack of sophisticated interactions that engages users as a process of waking up. Sure most of the time a distorted sound or emitted fragrance could wake people up but it is inevitable that someone just falls back asleep immediately right after responding to the device. Sometimes, people even forget to set up alarms before they decide to take a snooze.

When the alarm goes off, we are able to subconsciously hit the snooze button without even being fully conscious. To solve these issues, Interactive Alarm implements a system to require users to complete customisable tasks before being able to turn off the alarm while mimicking a calendar format. While setting up the alarm, users are able to randomize or customize trivial questions that require a certain degree of thinking to answer. These questions will appear after the user hits the snooze button. But if they are not answered, meaning the user has gone back to sleep, the alarm will go off shortly after the questions are not answered correctly or left blank. This process will repeat until all the questions are answered correctly. To ensure that the user is fully awake after answering the questions, they have the choice to adjust the difficulty and the quantity of the questions. It could range from basic questions like “what times is it?” or thought provoking ones like “what am I waking up for?” Interactive Alarm also features a calendar-like interface where an alarm could be set for multiple days of the week [8]. For example if the user needs to wake-up early on all the weekdays. All alarms can be seen in each section underneath the column for the day of the week. This way, users don’t have to remember to turn on the alarm beforehand for regular scheduled alarms.

To prove that the program is stable, we relied on the application as an alarm for 3 weeks. Every single day, the alarm went off at the correct time and we altered between answering the questions and not. For the occasions that we answered the questions, the alarm successfully turned off. For those of which we did not answer the questions, the alarm was able to sound again after a short time. We also recommended this application to several classmates for them to try out. Most of them reported that it was easier for them to wake-up using Interactive Alarm compared to the preexisting alarm applications on their phones.

The rest of the paper is organized as follows: Section 2 gives the details on the challenges thatwe met during the experiment and designing the sample; Section 3 focuses on the details of our solutions corresponding to the challenges that we mentioned in Section 2; Section 4 presents the relevant details about the experiment we did, following by presenting the related work in Section 5. Finally, Section 6 gives the conclusion remarks, as well as pointing out the future work of this project.

2. CHALLENGES

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

2.1. New programming language

To develop this application, we learned a new programming language [9]. In order to use the flutter app we learned about Dart, It has its similarity with a few of the languages we already know such as C++, Java, and other languages used for object oriented programming [10]. But before this, we rarely had any knowledge about graphic user interfaces. This includes learning various emulators, and many new syntax for designing the pages of the application.

2.2. Connecting the user data to online database

Another challenge is connecting the users data to the online database and grouping the user’s information under the username. We wanted to incorporate an account system into the application so that users could access all of their alarms on other devices. We had to learn how to Firebase to store user information like the properties of the alarms, the username and password; additionally, figuring out how to access and update pre-existing data when users want to change or delete alarms or if they log in on another device.

2.3. Sitting for hours to debug

The hardest challenge was having to sit down for hours debugging the program for tons of errors.

While syntax errors were fairly obvious, logical errors required additional conditional testing to locate and fix. The biggest issue was the inconsistency in the errors. For example, the application would occasionally not switch to the screen containing the questions after pressing the snooze button. It all came down to patience.

3. SOLUTION

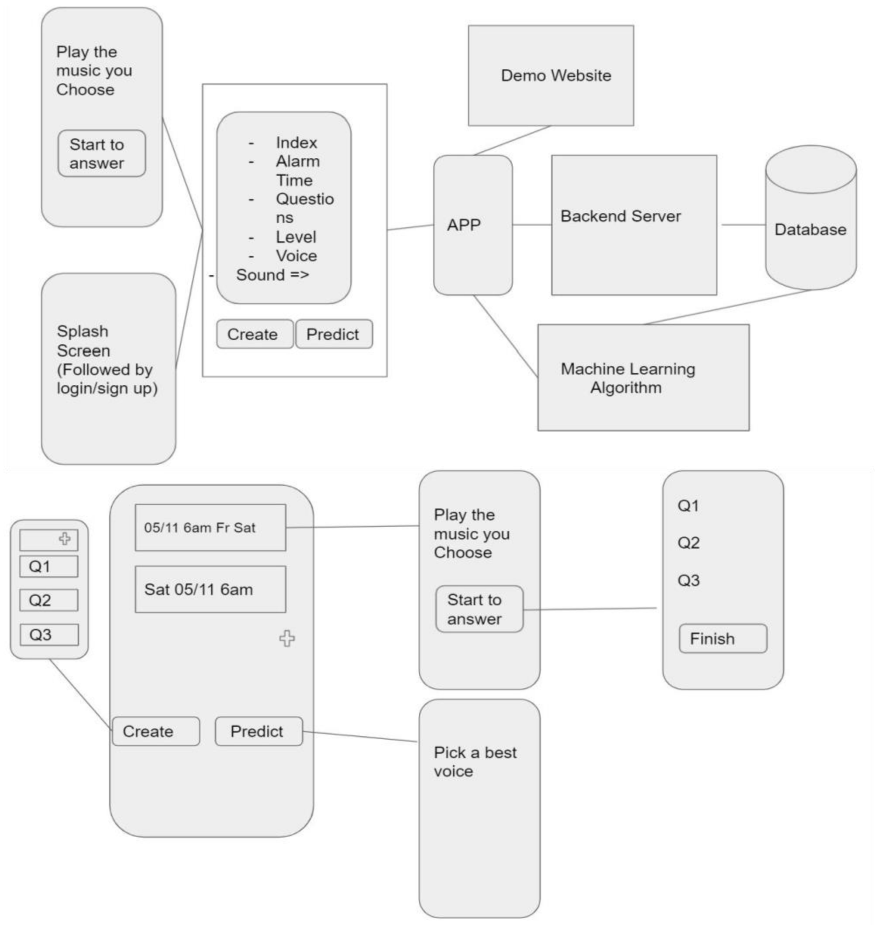
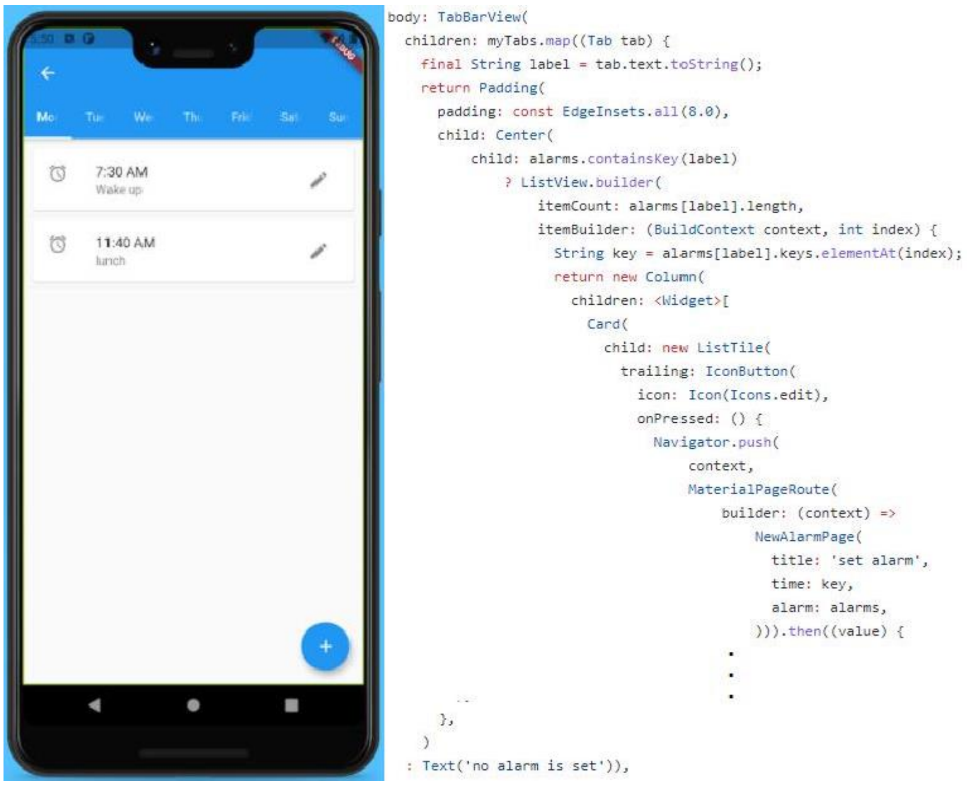


Figure 1. Overview solution of Interactive alarm app

Interactive Alarm is an alarm app with an account system that stores the alarms of users in an online database. After logging in or creating an account, users can create change or delete alarms. Properties of an alarm include ringtone, index/name, time, corresponding question(s), preset volume, and days of the week. Each alarm could be set to go off on selected days of the week. The home page offers tabs for the days of the week so that the user can get a view of all the alarms on a certain day in chronological order. There will also be a demo website for interested people. We also plan on implementing machine learning to predict the ideal volume/questions difficulty needed to wake the user up. Those predictions based on pass data will also be conveniently stored in the online database under each user's information.

The interactive alarm app contains diverse screens. A home screen that contains the list of alarms, an set alarm screen to set up the alarms and the awake/answer screen to answer mathematical or customized questions.

As shown in Figure 2, the home screen contains a tab menu for the day of the week. Each tab contains the list of alarms that is set up for the day of the week. This list shows the time and the description/title of the alarm and the option to edit the alarm. To develop this feature, we use the TabBarView widget that is provided by the material flutter package, an instance of a list for the week day and the alarm information that is saved in the firebase database if it exists. Since there is some day of the week that does not have any alarm set up, we implemented an if-else condition to show the list of the alarms if it exits in the database otherwise the tab shows the message that the “no alarm is set”.



```
body: TabBarView(
  children: myTabs.map((Tab tab) {
    final String label = tab.text.toString();
    return Padding(
      padding: const EdgeInsets.all(8.0),
      child: Center(
          child: alarms.containsKey(label)
              ? ListView.builder(
                  itemCount: alarms[label].length,
                  itemBuilder: (BuildContext context, int index) {
                    String key = alarms[label].keys.elementAt(index);
                    return new Column(
                      children: <Widget>[
                        Card(
                          child: new ListTile(
                            trailing: IconButton(
                              icon: Icon(Icons.edit),
                              onPressed: () {
                                Navigator.push(
                                    context,
                                    MaterialPageRoute(
                                        builder: (context) =>
                                            NewAlarmPage(
                                              title: 'set alarm',
                                              time: key,
                                              alarm: alarms,
                                            ))).then((value) {
                                         .
                                         .
                                         .
        ..
      },
    )
  : Text('no alarm is set')),
```

Figure 2. Home Screen User Interface and implementation

In the Set alarm screen, users add/select the title/description of the alarm, time, a single or multiple day of week and the ringtone. (see Figure 3) Also, they can use the mathematical questions that are provided by the app or use their own questions. The user can select between 1 to 5 questions, so if a user uses an alarm to wake-up and always oversleep then the user can select more than one question to help him/her to wake-up.

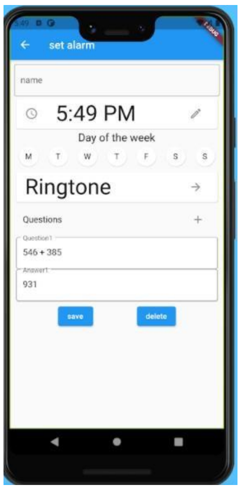
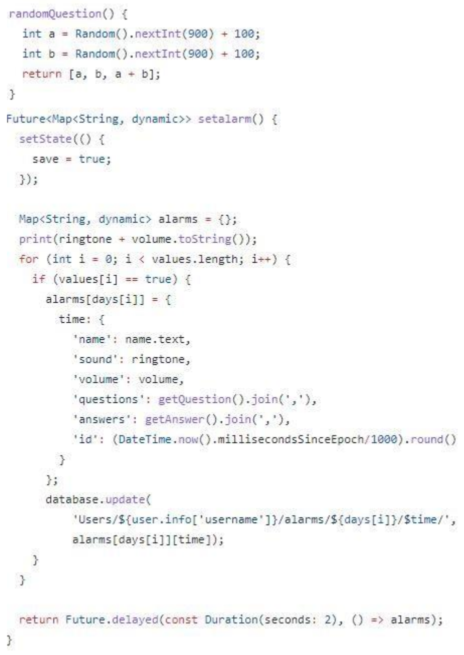


Figure 3. Set up alarm Screen

To automatically generate mathematical questions, we use the random function to select 2 numbers between 1-1000, and add them together to create the addition equation.(see Figure 4) After users select and add the alarm information, the information is sent to the Firebase. We use the Firebase Realtime Database to store the information as a Map structure. (see Figure 5) To develop this feature, we created a Map that contains the name, volume, ringtone, list of questions and answers and time as keys with their respective values. Then the Map is sent to the database and stored in the corresponding user.



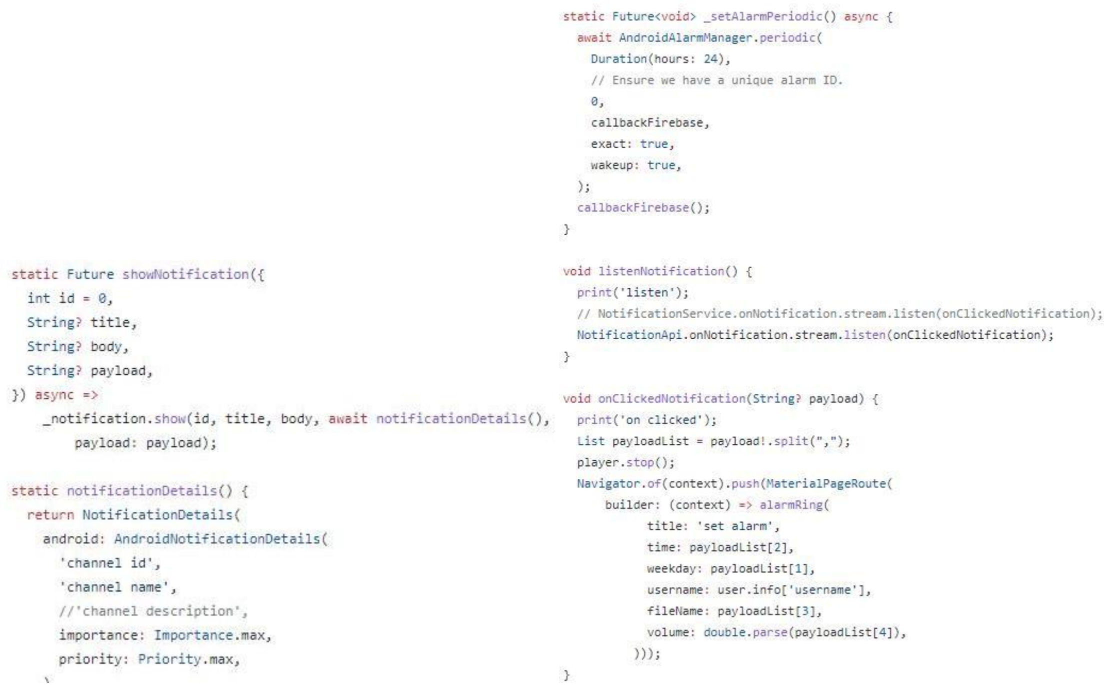
```
randomQuestion() {
  int a = Random().nextInt(900) + 100;
  int b = Random().nextInt(900) + 100;
  return [a, b, a + b];
}
Future<Map<String, dynamic>> setalarm() {
  setState(() {
    save = true;
  });

  Map<String, dynamic> alarms = {};
  print(ringtone + volume.toString());
  for (int i = 0; i < values.length; i++) {
    if (values[i] == true) {
      alarms[days[i]] = {
        time: {
          'name': name.text,
          'sound': ringtone,
          'volume': volume,
          'questions': getQuestion().join(','),
          'answers': getAnswer().join(','),
          'id': (DateTime.now().millisecondsSinceEpoch/1000).round()
        }
      };
      database.update(
          'Users/${user.info['username']}/alarms/${days[i]}/$time/',
          alarms[days[i]][time]);
    }
  }

  return Future.delayed(const Duration(seconds: 2), () => alarms);
}
```

Figure 4. Set alarm screen Code

To play the ringtone in the background and schedule alarms, we use flutter_local_notifications and android_alarm_manager packages respectively. As shown in Figure 5, to set up the notification, we set the priority and importance fields to be higher, so the notification shows immediately at the time that is set up. The notification shows the time and the title/description of the alarm, and sound setting as “payload”. When users click the alarm notification, the mobile screen is redirected to the screen to answer the questions with the corresponding alarm information. To set up and schedule alarms, we use the android_alarm_manager package to fetch the data every 24 hours from the Firebase and schedule the alarm for the corresponding day of week.



```
static Future showNotification({
  int id = 0,
  String? title,
  String? body,
  String? payload,
}) async =>
    _notification.show(id, title, body, await notificationDetails(),
        payload: payload);

static notificationDetails() {
  return NotificationDetails(
    android: AndroidNotificationDetails(
      'channel id',
      'channel name',
      //'channel description',
      importance: Importance.max,
      priority: Priority.max,
    \
```

```
static Future<void> _setAlarmPeriodic() async {
  await AndroidAlarmManager.periodic(
    Duration(hours: 24),
    // Ensure we have a unique alarm ID.
    0,
    callbackFirebase,
    exact: true,
    wakeup: true,
  );
  callbackFirebase();
}

void listenNotification() {
  print('listen');
  // NotificationService.onNotification.stream.listen(onClickedNotification);
  NotificationApi.onNotification.stream.listen(onClickedNotification);
}

void onClickedNotification(String? payload) {
  print('on clicked');
  List payloadList = payload!.split(",");
  player.stop();
  Navigator.of(context).push(MaterialPageRoute(
      builder: (context) => alarmRing(
            title: 'set alarm',
            time: payloadList[2],
            weekday: payloadList[1],
            username: user.info['username'],
            fileName: payloadList[3],
            volume: double.parse(payloadList[4]),
          )));
}
```

Figure 5. Set notification code

4. EXPERIMENT

In order to verify that our solution can effectively solve problems at different levels and have good user feedback, we decided to select multiple experimental groups and comparison groups for several experiments. For the first experiment, we want to prove that our solution works stable and continuously, so we choose a group size of 40 different trials in 4 different math problems. The 4 different types of problems are add operations, sub operations, multiple operations, divide operations. The goal of the first experiment is to verify if the Calculation System works good for different types of operations of math problems. Through sampling 4 groups of operations of math problems ask the same person to finish all these tasks with the schedule of our app. Result is collected by statistics if the app releases the clock once the user answers the problem correctly. Experiments have shown that all operations in different types unlock the screen and stop the clock. Add operations has the most correct rates, which means our user are works more better in Add operations. This experiment could explain that the operation types do have a obvious impact on the arrange results. The average using time (in minutes) of 4 different types of the operations shows below:

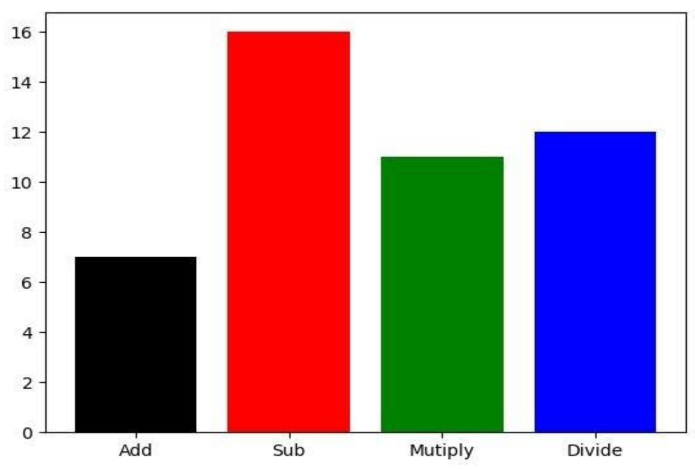


Figure 6. Result of experiment 1 (1)

A good user experience is as important as a good product. So a perfect solution should have excellent user experience feedback.In order to prove that our solution has the best user feedback, we specially designed a user experience questionnaire base on the US system usability questionnaire rules. We statistics the feedback result from 100 users, Track the user's data for 5 days, let them explore freely on the functionality. we divide those users into Five different groups. The first group of users ages from 10 - 20, the second group of users ages from 20 - 30, the third group of users ages from 30 - 40, the fourth group of users ages from 40 - 50, the fifth group of users ages from 50 - 60The goal of the first experiment is to verify high feedback scores shows high performance We collect the feedback scores form these 5 different group of users and analyze it.Experiments have shown that users who ages from 10 - 20 give the highest result feedback to our app. Which may because of the age between those range are more likely to have trouble with get up from bed. The experiment graph shows below:

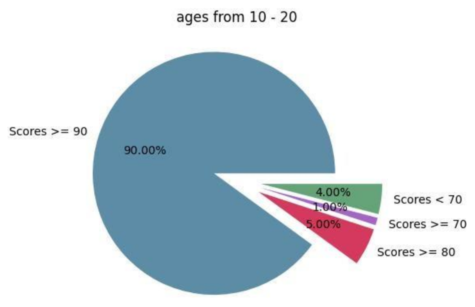


Figure 7. Result of age 10-20

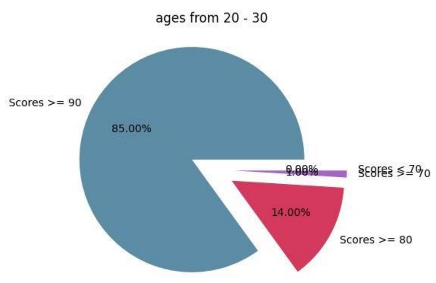


Figure 8. Result of age 20-30

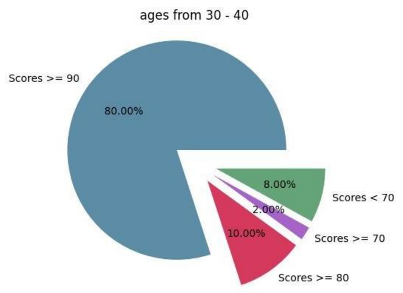


Figure 9. Result of age 30-40

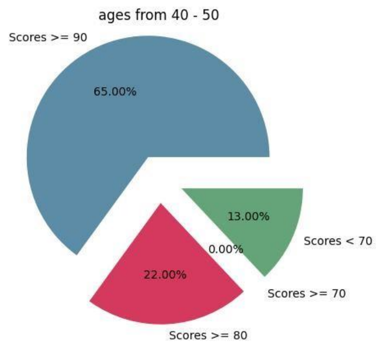


Figure 10. Result of age 40-50

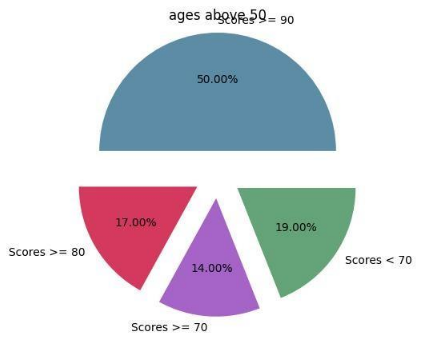


Figure 11. Result of age above 50

5. RELATED WORK

Oh, et al.perform and analysis of a Wake-Up Task-Based Mobile Alarm App [11]. In this analysis, they compare different features: touching a button, taking a picture, shaking the device, and solving math problems at the time that the alarm fires. The analysis shows that users who set harder tasks, math or picture tasks, do not fall asleep again as compared to the normal or easy tasks that alarms snooze multiple times. Similar to this mobile app, we use math equations to help users to be aware and not fall asleep after the alarm is fired.

Fagerjord presented a study of mobile applications [12]. The author says that we are moving to a sensor era and the mobile app would not diminish the access to the Web since mobile apps need access to the web as part of their important feature. One of the mobile apps that the author describes is the Sleep cycle that utilizes touch screen and the accelerometers to calculate the sleep cycle and matching sleep cycle to wake-up time. As different from our app, we do not compare the desired wake-up time with the average sleep cycle to wake-up users; we use the questions at the time that the alarm fires, so the users would be more aware and not fall asleep again. Ilhan, et al. presented a study of using Gamification in mobile applications related to subjective well-being and sleep/wake activities [13]. The study shows that Gamification can motivate people to begin their day and improve their sleep-wake behaviors.

6. CONCLUSIONS

In this project, we proposed an Android mobile application that helps people to wake-up and not fall asleep again [14]. The interactive alarm app provides a wake-up task that involves mental efforts. Users must solve mathematical equations or answer their customized problems and/or questions in order to stop the ringtone otherwise the alarm would fire again until all the questions are responded. To develop the mobile app, we use Firebase realtime database to store alarms for each user and flutter_local_notifications and android_alarm_manager packages to play the ringtone in the background and schedule alarms.

Some limitations that we encountered during this study is that we could only develop the interactive alarm app for only Android devices since some of the flutter packages can be only set up in Android devices. Moreover, we had a time limitation,so we could not develop the feature in which users can include their own ringtone for their alarm [15].

As for future work, we desired to build and deploy the interactive alarm in the IOS platform since some of the features that we use in our app are only for the Android platform. Also, we would like to add a feature where users can input their own ringtone instead of using the ringtone provided for the app since we believe that adding the customized ringtone can help users to wake-up easier.

REFERENCES

[1] Ferrara, Michele, and Luigi De Gennaro. "How much sleep do we need?." Sleep medicine reviews 5.2 (2001): 155-179.
[2] Murphy, Joseph F., et al. "Academic press: Translating high expectations into school policies and classroom practices." Educational Leadership 40.3 (1982): 22-26.
[3] Dignum, Frank, et al. "Meeting the deadline: Why, when and how." International Workshop on Formal Approaches to Agent-Based Systems. Springer, Berlin, Heidelberg, 2004.
[4] Robin, Pierre-Yves F. "Note on effective pressure." Journal of Geophysical Research 78.14 (1973): 2434-2437.
[5] Bryant, Penelope A., and Nigel Curtis. "Sleep and infection: no snooze, you lose?." The Pediatric infectious disease journal 32.10 (2013): 1135-1137.
[6] Izadi, Iman, et al. "An introduction to alarm analysis and design." IFAC Proceedings Volumes 42.8 (2009): 645-650.
[7] Meisner, David, and Thomas F. Wenisch. "Dreamweaver: architectural support for deep sleep." ACM SIGPLAN Notices 47.4 (2012): 313-324.
[8] Lines, Lorna, and Kate S. Hone. "Eliciting user requirements with older adults: lessons from the design of an interactive domestic alarm system." Universal Access in the Information Society 3.2 (2004): 141-148.
[9] Iverson, Kenneth E. "A programming language." Proceedings of the May 1-3, 1962, spring joint computer conference. 1962.
[10] Godefroid, Patrice, Nils Klarlund, and Koushik Sen. "DART: Directed automated random testing." Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation. 2005.
[11] Oh, Kyue Taek, et al. "Analysis of a Wake-Up Task-Based Mobile Alarm App." Applied Sciences 10.11 (2020): 3993.
[12] Dieter, Michael, et al. "Multi-situated app studies: Methods and propositions." Social Media+ Society 5.2 (2019): 2056305119846486.

[13] Ilhan, Ayse Ezgi, Bahar Sener, and Huseyin Hacihabiboglu. "Improving Sleep-Wake Behaviors Using Mobile App Gamification." Entertainment Computing 40 (2022): 100454.

[14] Ma, Li, Lei Gu, and Jin Wang. "Research and development of mobile application for android platform." International journal of multimedia and ubiquitous engineering 9.4 (2014): 187-198.

[15] Doz, Yves, and Keeley Wilson. Ringtone: Exploring the rise and fall of Nokia in mobile phones. Oxford University Press, 2017.