Design and Implementation of an Online and Cost-Effective Attendance Management System Using Smartphones and Cloud Services

M. Fahim Ferdous Khan(✉), Taisei Yamazaki, and Ken Sakamura

Faculty of Information Networking and Design (INIAD), Toyo University, Tokyo, Japan
{khan,ken}@sakamura-lab.org, s1f101701823@iniad.org

Abstract. Taking attendance in schools is an important daily activity which is directly related to students' academic performance. Traditional roll-call-based or signature-based approaches for attendance keeping are cumbersome, time consuming, and susceptible to manual errors and fraudulent maneuvers. With a view to digitalizing the process of taking and maintaining attendance record, in this paper, we present a distributed approach to attendance management using smartphones and cloud computing web services that offer high degree security and scalability. The main objective in developing the proposed attendance management system has been an implementation that does not rely on any dedicated infrastructure, and hence reduces cost as much as possible, enabling a wide range of organizations regardless of size and budgetary constraints to adopt such a system. It goes without saying that such cost reduction is inherently associated with some trade-offs which are also discussed in the paper.

Keywords: Attendance management system · Cloud services · Identity management · Application programming interface

1 Introduction

Grand initiatives like Industry 4.0 [19] and Society 5.0 [20] are proliferating at national, regional, or international levels. These endeavors aim at merging cyberspace and physical space by creating a super-smart and highly automated society. As we march toward making such visions a reality, different services that we experience on a day-to-day basis are going through rapid digital transformation. These services encompass all basic sectors of the society including administration, finance, health, and education. In the education sector, school attendance has traditionally been recorded by roll or name calls by instructors, or by having the students sign on a piece of paper that is passed around the class. Such manual and paper-based techniques are time consuming and prone to human errors. In large classrooms, there may also be problems of proxy attendance. Hence, significant amount of effort has been devoted to automating the process of taking attendance. In many affluent universities, students are provided with ID cards equipped with built-in integrated circuits. Such ID cards can be read by a card reader

T. Hara and H. Yamaguchi (Eds.): MobiQuitous 2021, LNICST 419, pp. 650–664, 2022.
https://doi.org/10.1007/978-3-030-94822-1_42

installed in classrooms, and hence students' attendance can be recorded. Unfortunately, many universities cannot afford such expensive systems. Therefore, there is a need for a cost-effective attendance management system which does not rely on a dedicated infrastructure. On this premise, in this paper, we present the design and implementation of a cost-effective attendance management system using smartphones and cloud service. The system is implemented using user (student) smartphones in combination with near-field communication (NFC) readable cards and Amazon Web Services (AWS). In order to make the system as cost-effective as possible, the focus has been on using already available components. Smartphones enjoy a high degree of penetration rate all over the world [21]. For the NFC readable cards, any general-purpose IC card or smartcard can be used, including the ones that are widely used for accessing public transportation in many parts of the world.

The rest of this paper is organized as follows. Section 2 discusses the major attendance management systems reported in the literature. Section 3 introduces our proposed attendance management system, and Sect. 4 explains its design and implementation. Section 5 evaluates our system and compares it merits and demerits to those of other systems. Finally, Sect. 6 concludes the paper.

2 Related Work

Different mechanisms have been proposed in order to automate the attendance management systems. Most of the approaches found in the literature can be categorized in one of the four approaches discussed below.

2.1 Smartphone-Based Attendance Management Systems

Noor et al. proposed an attendance management system for the purpose of contributing to paperless management, preventing data loss due to loss of paper records, and minimizing the cost of systemization [1]. This system scans and registers the student data in an online database which the class teacher can later download from a designated web server. Assuming that the teacher's smartphone is Android, no additional cost for hardware is required. It is necessary to install the apk file on the teacher's Android device. After launching the application with ID and password to log in, the app can scan student ID cards using the built-in camera of the smartphone. The fact that the faculty member can check the student ID card one by one using this system can be expected to be effective in preventing fraud, but it is also time-consuming and labor intensive. Smartcard is often used as an important component in many of the other system reported below.

2.2 RFID-Based Attendance Management Systems

A number of attendance management system for schools and workplaces using RFID technology have been proposed in the literature [2–5]. The basic idea is to have the students or employees carry some sort of RFID tag and read those by a reader installed in front of the entrance of classroom or event space. The read data is collected in a database from where administrators can access attendance record using web interfaces.

2.3 Bluetooth-Based Attendance Management Systems

Approaches based on Bluetooth communication have also been proposed [6, 7]. These approaches rely on the ability of an application installed on the instructor's mobile phone to query students' mobile phones to confirm their attendance. Such Bluetooth communication-based systems can also be used in combination with RFID [6].

2.4 Biometrics-Based Attendance Management Systems

As strong and definitive indicators of identity, biometrics have been incorporated in attendance management systems. Several biometrics-based systems are reported. Examples include fingerprint-based systems [8, 9], iris-recognition-based systems [10, 11], and face-recognition-based systems [12–14]. All of these rely on special hardware to read biometric data and analyze those using pattern recognition or deep-learning algorithms.

3 Overview of the Proposed Attendance Management System

The components of our proposed attendance management system are as follows. (1) FeliCa-based contactless smartcards that are commonly used for accessing public transport. For example, in Japan – where this research is based – Suica [24] or Pasmo [25] cards are widely used by commuters. In many other metropolitan areas of the world similar smartcards are used. Each FeliCa card has a unique 8-byte Manufature ID (IDm) [15] which is used as an important piece of information in our system. (2) Smartphone: iPhone is used in our implementation, but Android phones can also be used. (3) Amazon Web Services (AWS) [16] is used for storage and manipulation of attendance record.

The system sends an HTTP request to AWS together with JSON data using the POST method and records it in the database by program processing. This record serves as the attendance record of a student. The JSON data requires student ID, email address, name, operation code, and IDm. The iOS shortcuts app was used as a means to send HTTP requests. The AWS services that are used in the system are Lambda, DynamoDB, API Gateway, IAM, and S3. For the programming language of Lambda Python 3.8 was used.

It is necessary to read the FeliCa card with the reader application in advance to extract its IDm. In our implementation, we used an application called "Japan NFC Reader-Card Reader" [17] which can be downloaded from Apple App Store for free. Alternatively, a password and hash value can be used.

As discussed later in Sect. 4, this system has three functions: Register, Attend, and Check. Students can switch functions by using different operation codes.

Case-1: Using iPhone. We use the Shortcuts app provided by iOS 12.0 or higher [23]. This application has a *shortcut* function that allows one to freely customize a process by combining actions. A shortcut created this way can also be configured to be automatically triggered by an event; this phenomenon is called automation. Our implementation utilizes this automation function. When an IC card is touched on the upper-back side of an iPhone, the NFC reader reads the card and sends an HTTP request with JSON data to the set URL and receives a response. For multiple lessons, it is also necessary to have a screen that allows the user to select for which lesson to send the request to. Figure 1(a) shows

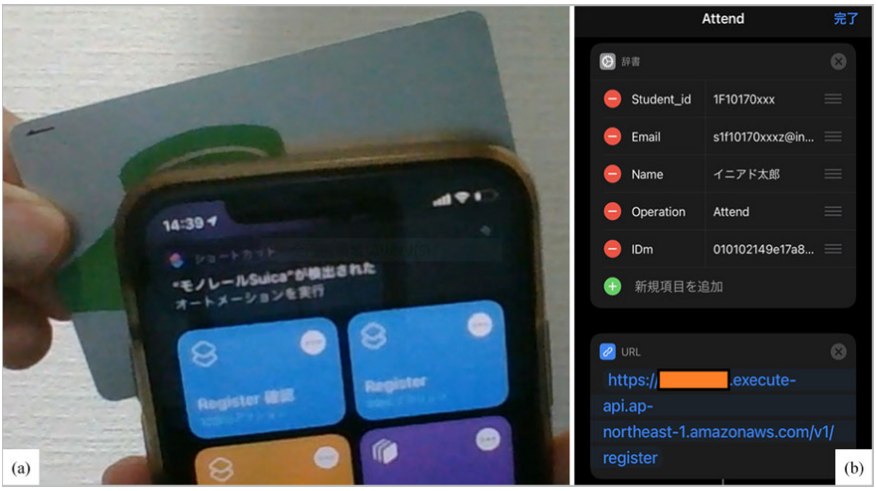

Fig. 1. (a) Touching Suica card on the iPhone's NFC reader and using the automation function of iOS Shortcuts app, (b) Information sent as attendance record includes student ID, e-mail, name, operation and IDm of the smartcard (in this example Suica) used

how IC card is touched on the iPhone and the shortcut is called using the automation function, and it is executed by tapping the banner at the top. Figure 1(b) shows what information is sent for the "Attend" operation.

Case-2: Using Android-Based Phone. Our system employs a method of sending HTTP requests to AWS from a student's personal device, assuming that the iOS Shortcuts app is used on the iPhone, but the same can be done on Android. An application called "HTTP Request Shortcuts" is available for free from the Google Play Store [18]. JSON data can be sent at the time of request and saved as a shortcut.

4 Design and Implementation

Now, referring to Fig. 2 and 3, we explain in detail how the different AWS components work together to realize the proposed attendance management system. We also explain how the attendance information can be handled by faculty members.

4.1 AWS API Gateway

The API Gateway issues a URL and connects an incoming HTTP request to another service. In this system, a URL for students to send is created as a REST API, and it functions as an API for operating Lambda, which will be described later in this section. This service uses resources, functions, deployment stages, and resource policies. A resource becomes a part of the URL, and if multiple resources are created, multiple services can be linked to one API simply by changing a part of the URL. To prevent complications in the proposed system, we created only "register" as a resource. Methods

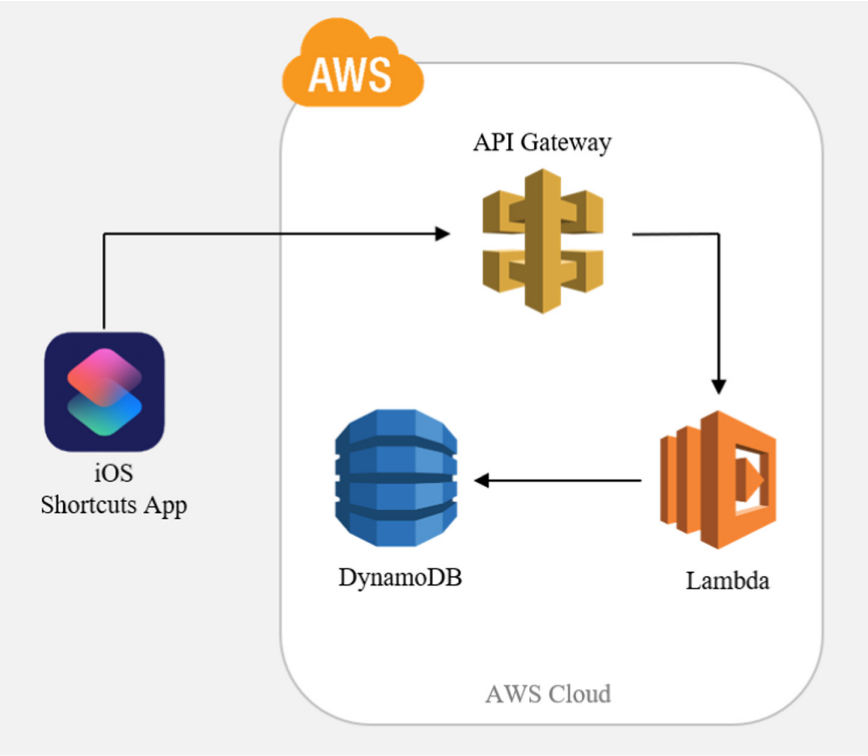

Fig. 2. The main AWS components utilized in the proposed attendance management system

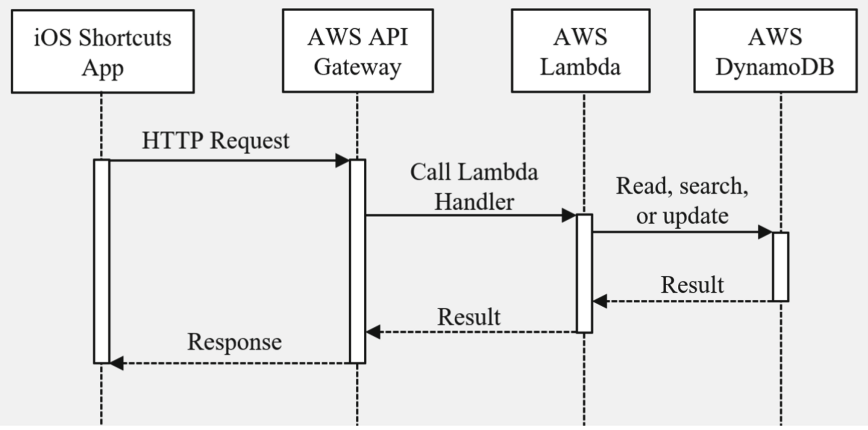

Fig. 3. Sequence diagram depicting the interaction between iOS Shortcuts app and major AWS components

can also be set for each resource, but only POST was created. Resource policy plays the most important role in this service. In attendance management systems via the Internet, in order to prevent fraud, it is necessary to limit from where students can send HTTP requests. The resource policy can control access to the API and is registered as JSON data. Since this system is premised on sending HTTP requests from the university, we adopted a whitelist type that accepts only the global IP address of the campus network where VPN blocking is enabled. Requests are rejected if the student is not connected to campus Wi-Fi.

4.2 AWS DynamoDB

DynamoDB is a NoSQL database on AWS. This system creates at least two tables: Student_info and Virtual_lecture. Assuming that the number of lessons is N, the number of tables needs to be N + 1 (i.e., N separate tables for N lectures). The rest of the explanation assumes that there is only one lecture, and hence two tables.

The Student_info table includes the following *attributes* (akin to *field* in relational database): Student_id (String, Partition key), Email (String, Sort key), Course (String), Grade (Number), Name (String), IDm (String). The Virtual_lecture table includes the following attributes: Student_id (String, Partition key), Email (String, Sort key), Class_id (String)Name (String), ClassData (Number). The combination of partition key and sort key is referred to as *composite primary key* in DynamoDB.

The attribute specified in the key is the information that Lambda needs to access the database. By specifying multiple attributes as keys, erroneous registration can be prevented if one is detected as incorrect.

```
 Key = {
       'Student_id': student_id,
       'Email': email
}
```

There is also a function called Item Explorer that allows one to search for items in the table by entering the partition key or sort key. It is useful for large classes or when data need to be looked up immediately.

4.3 AWS IAM

IAM (Identity and Access Management) is a service that allows to create roles by attaching policies. Roles assign and authorize other services when they are created. In this system, Lambda needs permission to operate on DynamoDB, so “AmazonDynamoDB-FullAccess”, and “AWSLambdaDynamoDBExecutionRole” are set so that Lambda has permission to fully access DynamoDB and update data. In addition, various policies are prepared depending on the application. For example, if read-only access is desired, the privilege level can be set to “AmazonDynamoDBReadOnlyAccess”.

4.4 AWS Lambda

The python program for the handler function is contained in AWS Lambda. The handler function is called after passing through the API Gateway whitelist type IP address filter. The transmitted JSON data is stored in the dictionary object event. The three functions are called according to the extracted operation code. The following example is a part of the program that refers to the operation code contained in event and acquires other data as well.

```
operation = event ['Operation']
```

The days and times when classes are held are stored in a data structure. For example, in the case of the class in the 3rd period on Friday, the combination of (day (str type), time (int type)) is ("Fri", 3). Requests are accepted only on the days of the week when classes are held.

There are four functions using which Lambda operates on DynamoDB: put_item, get_item, update_item, and delete_item. When executed, a response is returned with the HTTP status code, so a student can check whether the data operation was successful by referring to the status code and confirming that it is 200.

When DynamoDB operation is performed, metadata is returned, which is long and unnecessary for the students, and difficult to see on smartphones. Therefore, the response returned to students are formatted to include are HTTPStatusCode, Message, and other data for keeping the amount of information to a minimum.

Next, we discuss the three main functions provided by our attendance management system: Register, Attend and Check. These functions are defined in a file name lambda_function.py.

Register Function. Students must first register their information to the attendance management system, and the JSON data mentioned above must be saved in the Student_info table. For this initial registration, "Register" is specified in the operation code when sending an HTTP request. Only the grade is Number type, and the other attributes are String type, and these attributes saved using the put_item function. Among these, the student ID number is the partition key and the email address is the sort key. Data other than the operation code is updated for each execution. If the FeliCa card used is changed, the IDm is read again, and the database is updated using this function with the new IDm. The response includes the HTTP status code, message, and registered data. Definition of the Register function using put_item function is shown below as an example.

```
def operation_register(student_id, course, email, grade, name, idm):
    Response = student.put_item(
        Item = {
            'Student_id': student_id,
            'Course': course,
            'Email': email,
            'Grade': grade,
            'Name': name,
            'IDm': idm,
        }
    )
```

Attend Function. This function is used by students to register attendance information. The table that is updated is called Virtual_lecture, and information is saved on a class-by-class basis. Students specify "Attend" as the operation code. There are some conditions for this function to register attendance information in the database, and it is executed only when all of them are satisfied. Processing and determination of conditions are performed in the following order.

1) If the operation code is Attend, immediately save the date and time in the variable Datetime, and check that day of the week matches the day of the week when the class is held. If the day of the week is different, there will be no lessons and the request will be invalidated.
2) Execute the operation_attend function. Pass the student ID number, email address, name, request date and time, class time limit, and IDm as arguments.
3) Generate a key for registering in DynamoDB from the request date and time using the strftime function and save it in the variable Class_today. The format is "Class_yyyymmdd". For example, if the request date and time is December 23, 2020, then Class_today = Class_20201223.
4) Execute the timecmp function of the timetable.py module described later in this section. The lambda_function.py program imports this timetable.py module. Compare the time when the request was received with the reception start time, class start time, and the time when lateness is recognized for each class period (set to 20 min after the class start time in our implementation). A numerical attendance score is returned as follows. Normal attendance is 2, late arrival is 1, absenteeism is 0. However, if the request time is earlier than the acceptance start time, error code -1 is returned, the function is interrupted, and the request is invalidated.
5) Specify the student ID number and email address as keys from the Student_info table, and retrieve the data. If these two keys are correct, the data can be retrieved as an "Item" (akin to "row" in relational database). If there is an error, there is no corresponding data, so nothing is entered in the as attributes of "Item". In such cases, error code -2 is returned, and the request is invalidated.
6) Similarly, it is determined whether the IDm registered in the Student_info table and the IDm of the JSON data at the time of HTTP request match. If they are different, error code -3 is returned, and the request is invalidated.

7) Extract the data of the Virtual_lecture table. At the first execution, there is no student data in the Virtual_lecture table and the "Item" is None (i.e., empty table), so create a new "Item" using put_item method. The registered data refer to the JSON data when the HTTP request is sent. ClassData that records attendance information is initialized in a dictionary type object and recorded for each date and time corresponding to each class period. If the response is not 200, error code -4 is returned, the request is invalidated, and a message is returned to the user asking him to retry.
8) Try to register attendance information in the Virtual_lecture table using the try-except statement. If the ConditionExpression parameter is set in the update_item function that updates the database, the update will be executed according to the UpdateExpression parameter when the condition is matched. This condition specifies that "attendance information on the day of class does not exist in the database". Without this condition check, if multiple requests are sent, the last request will be overwritten. Since the attendance score is calculated at this point, there is a possibility that the score will decrease, such as the one with 2 recorded in normal attendance becoming 1 as late attendance. If this condition is not met, an exception occurs because attendance information has already been registered. Accordingly, the request is invalidated. The next two lines of code are a part of the program code that creates the ConditionExpression and UpdateExpression parameters.

```
condition_text = 'attribute_not_exists(ClassData.{})'.format(Class_today)
update_text = 'set ClassData.{}=:s'.format(Class_today)
```

(The ': s' part of update_text refers to the variable score.)

When the above conditions 1,4,5,6,7,8 are matched, it is saved as a dictionary object in ClassData of Virtual_lecture. The value of Class_today in step 3 is important. Based on this, the attendance score is calculated in step 4. As shown in Fig. 4(a), the response returns the HTTP status code, message, name, class name, and time accepted by AWS.

Check Function. After submitting attendance, students may want to check whether their attendance was properly recorded or not. The Check function serves this purpose. This function allows checking of the registration status of attendance information from the database for each class registered with the Attend function mentioned above. For IDm authentication, specify the student ID number and email address as keys from the Student_info table and retrieve them with get_item. If the two keys are different, error code − 2 is returned, and if the IDm is different, error code − 3 is returned to invalidate the request. If there is no error, the requested data are fetched from the Virtual_lecture table by the get_item function in the same way, and the database information, message, and HTTP status code are returned. Figure 4(b) shows the result of executing the Check function.

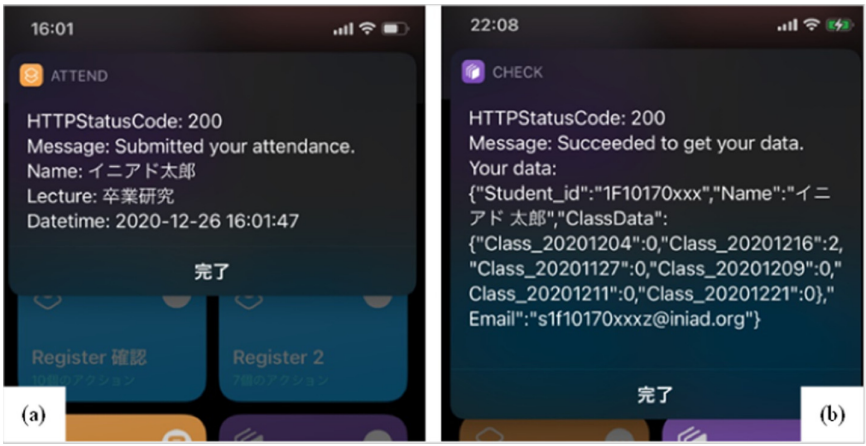

Fig. 4. (a) Screen output of successful attendance submission, (b) screen output of attendance check function

4.5 Time Comparison Method

The timetable.py module contains the class timetable and timecmp function. In the timetable, the reception start time of each time period, the class start time, and the time when lateness is recognized can be referred to in a dictionary type object. In our implementation, the reception start time is set to 30 min before the 1st and 3rd periods and 15 min before the 2nd, 4th, 5th and 6th periods. The timecmp function takes two parameters: the time when the request was received by AWS and the class period corresponding to the day of the week, and outputs a score by comparing it with the reception start time according to the timetable. If a request is received before the reception start time, − 1 is returned and the request is invalidated. As mentioned before, normal attendance, late attendance and absenteeism are given a scores of 2, 1, and 0 respectively.

4.6 Handling of Attendance Data by Faculty Member

If faculty members want to organize attendance data at the end of the semester, DynamoDB data can be uploaded to Amazon S3 and downloaded locally. Amazon S3 is a cloud storage service where S3 means Simple Storage Service. In S3, one group is called a bucket, and it is created by giving it a name. A folder can be created in it. In our implementation, the bucket name is univ-attend, and a virtual_class folder is created in it. Folders for other classes can be created in the bucket in the same way. Next, the data from the DynamoDB Virtual_lecture table to the S3 virtual_class folder can be exported. The detailed procedure is as follows.

1) Select the Virtual_lecture table from the DynamoDB console and select Streams and Exports.
2) Select Export to S3 to browse or enter your destination S3 bucket. For example, “s3://univ-attend/virtual_class/”.

3) Choose the S3 bucket owner. Make additional settings. Set export, file format, and encryption key type from a specific point in time or current time in the last 35 days. Select the current time, DynamoDB JSON, and Amazon S3 key, respectively, and select "Export" to export to S3.
4) Download the data locally from S3. A json.gz file is created in univ-attend/virtual_class/AWSDynamoDB /******/data, so select it, download it, and unzip it using the gunzip command. The ****** part is automatically assigned to a unique number when exported. The unzipped JSON file contains student attendance data.

Alternatively, any table can be directly downloaded as a csv file from DynamoDB console, but we have experienced formatting and encoding issues in files downloaded in this way especially if the table contains entries written in non-roman characters.

5 Evaluation

We tested the behavior of the proposed system in order to verify whether it can prevent unauthorized attendance and incorrect registration.

5.1 Testing

For verifying the correctness of operation of the system, the following six cases are considered.

Access from Off-Campus Network. The API Gateway resource policy described earlier rejects HTTP requests from locations other than the university. Figure 5(a) shows a request when the connection with Wi-Fi is disconnected. Access from the 4G network, for example (in the upper right of the image) is not allowed. VPN blocking is enabled.

Requests When There is no Class. Requests on days/times when there are no class are rejected. Figure 5(b) shows a request on a non-class day.

Incoming Requests Before the Reception Time. In our implementation, reception is 30 min (1st and 3rd period) or 15 min (2nd, 4th, 5th and 6th period) before the class start time. Any request before the reception time is rejected. Figure 5(c) shows a message informing a student that the reception time has not started yet.

Requests with Incorrect Key Specification. When searching or updating the DynamoDB database with Lambda, it cannot be accessed if the key specification is incorrect. In our system, the partition key is assigned to Student_id and the sort key is assigned to Email attribute. If either is incorrect, the desired item cannot be found. Figure 5(d) shows the message when the keys are incorrect.

Request with Incorrect IDm. The IDm is required when using the three functions, but if it is incorrect, the request is rejected. Figure 5(e) shows a message when the IDm is different from what is expected, prompting the student to confirm.

Multiple Requests. After the first successful request, subsequent requests are rejected in order to prevent the score from being overwritten (Fig. 5(f)).

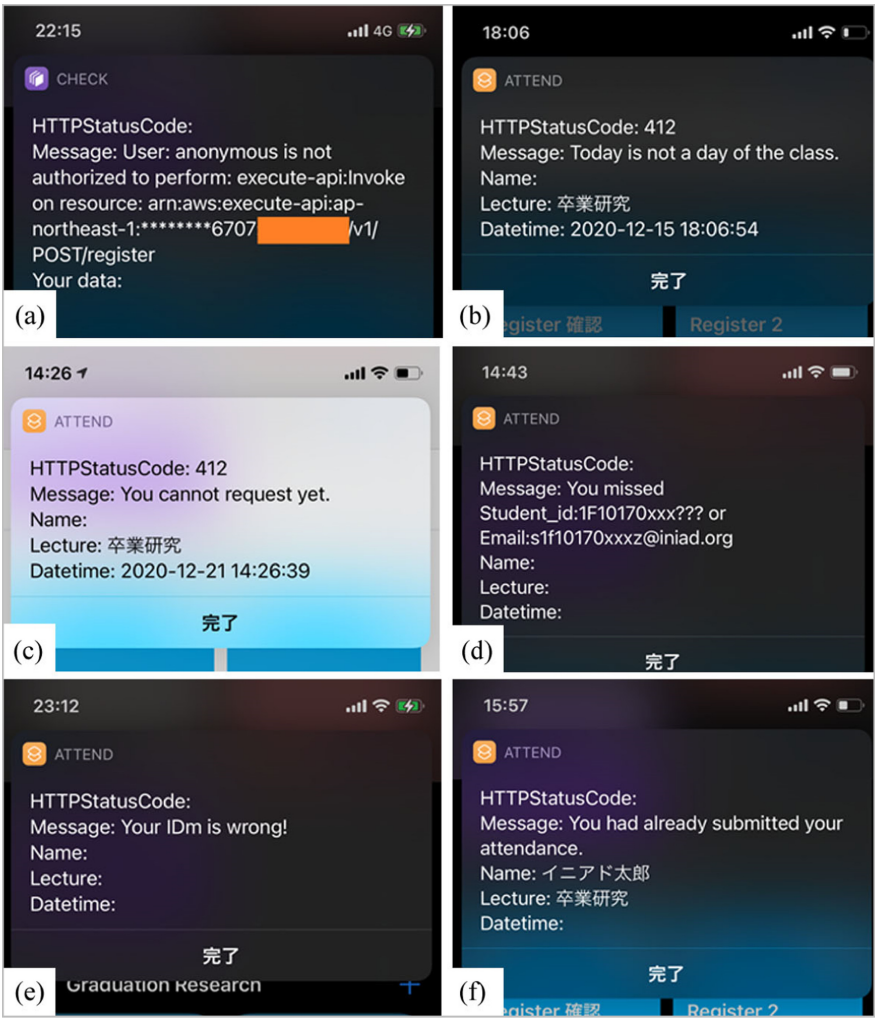

Fig. 5. (a) Screen output for (a) request from off-campus network, (b) request for non-existent class, (c) request before class reception time, (d) request with incorrect key, (e) request with incorrect IDm, and (f) multiple request

5.2 Discussion

Our proposed attendance management system does not require any dedicated hardware or infrastructure. Smartphones are part and parcel of modern day-to-day life. FeliCa-based cards are also used widely by commuters in many cities. In the attendance management system, students can use the same FeliCa card they use for commuting. Services of AWS is not free, but it is not expensive either as long as requests are as simple as the ones used in our system. The cost incurred for AWS provides reliability, security and scalability as AWS services are fully managed by Amazon Inc. It can be argued that

using cloud services is a better option rather than storing data in enterprise's own local servers considering management and maintenance issues.

One drawback of our system is that it cannot recognize the precise location of the student. It relies on requests coming from university network (with VPN blocking), so a student who may not be in the classroom can still be able to register attendance from somewhere else in the campus as long as he is connected to the campus Wi-Fi. Approaches that use GPS tracking also has the same problem as GPS measurement is not precise. This problem cannot be solved without introducing other components in the system, such as RFID or Bluetooth where physical proximity to the venue is required, for which additional hardware (reader, Bluetooth-enabled tag, or microcontroller as a separate token) and/or Bluetooth facility on mobile phones, is necessary. Other systems which are based on biometrics recognition are inherently expensive as dedicated hardware have to be purchased. These approaches are privacy-invasive as well. Moreover, the reader areas can get congested and may introduce a single point of failure in the system. On the other hand, our system takes a more distributed approach as students can register their attendance from anywhere in the class.

Preventing colluding students is also a challenge in our proposed system. A student may give his card to another student who can act as a proxy. To counter this, IMEI associated with each smartphone can be made a part of the request. Even with this addition, it is not completely secure, as – however unlikely it might be – a student can give his smartphone to his proxy. Therefore, in essence, the NFC readable card acts only as a week layer of security in the form of IDm check. To make it foolproof, smartcards based on asymmetric key cryptography approach would be necessary. For example, our eTRON architecture [22] can be used. This will incorporate a slight time lag in the processing of request, and an additional burden for managing cryptographic services. Such highly secure mechanisms are indeed necessary for critical operations, especially the ones involving financial transactions and sensitive private information, but we argue that they are not so important for a cost-effective attendance management system.

From a faculty member's perspective, the most important benefit of our system is that it can completely offload the burden of attendance management from him. He just has to download the attendance record from DynamoDB once or a few times in the semester. This will enable him to focus solely on planning and delivering lectures as attendance records are sent by students' own initiative. To summarize, the main contribution of this paper lies in implementing an attendance management system that uses devices and components that the user already owns, effectively eliminating any cost for additional infrastructure. Properties of the user-owned components are utilized in a systematic manner to manage attendance record leveraging fully managed and highly scalable AWS services.

6 Conclusion

In this paper, we have explained the design and implementation of an online attendance management system that utilizes students' smartphones, general-purpose FeliCa-based smartcards and cloud services. The system prevents fraudulent attendance by using techniques like identity management and IP address filtering. It can recognize three

different levels of attendance, namely, in-time attendance, late attendance, and non-attendance. The system is cost-effective as it does not require any dedicated hardware or infrastructure to be implemented. It is also distributed in nature as students do not have to pass through any dedicated reader area which can often be congested. The trade-offs associated with our proposed system are also discussed in the paper. In short, we believe our system strikes a good balance between performance and cost-effectiveness.

References

1. Noor, S.A.M., Zaini, N., Latip, M.F.A., Hamzah, N.: Android-based attendance management system. In: IEEE Conference on Systems, Process and Control (ICSPC), pp. 118–122 (2015)
2. Koppikar, U., Hiremath, S., Shiralkar A., Rajoor, A., Baligar, V. P.: IoT based smart attendance monitoring system using RFID. In: 1st International Conference on Advances in Information Technology (ICAIT), pp. 193–197 (2019)
3. Nainan, S., Parekh, R., Shah, T.: RFID Technology Based Attendance Management System. Int. J. Comput. Sci. Iss. 10(1) (2013). https://arxiv.org/ftp/arxiv/papers/1306/1306.5381.pdf, Accessed 20 Jan 2022
4. Maramis, G.D.P., Rompas, P.T.D.: Radio frequency identification (RFID) based employee attendance management system. In: 2nd International Conference on Innovation in Engineering and Vocational Education (2017). https://iopscience.iop.org/article/10.1088/1757-899X/306/1/012045/pdf. Accessed 20 Jan 2022
5. Shengli, K., Jun, Z., Guang, S., Chunhong, W., Wenpei, Z., Tao, L.: the design and implementation of the attendance management system based on radio frequency identification technology. In: International Conference on Electronic Science and Automation Control, pp. 189–192 (2015)
6. Lodha, R., Gupta, S., Jain, H., Narula, H.: Bluetooth smart based attendance management system. In: International Conference on Advanced Computing Technologies and Applications (ICACTA), pp. 524–527 (2015)
7. Bhalla, V., Singla, T., Gahlot, A., Gupta, V.: Bluetooth based attendance management system. Int. J. Innov. Eng. Technol. (IJIET) **3**(1), 227–233 (2013)
8. Mittal, Y., Varshney, A., Aggarwal, P., Matani, K., Mittal, V.K.: Fingerprint biometric based access control and classroom attendance management system. In: Annual IEEE India Conference (INDICON), pp. 1–6 (2015)
9. Adetiba, E., Iortim, O., Olajide, A.T., Awoseyin, R.: OBCAMS: an online biometrics-based class attendance management system. African J. Comput. ICT **6**(3), 25–38 (2013)
10. Seifedine, K., Smaili, M.: Wireless attendance management system based on iris recognition. Sci. Res. Essays **5**(12), 1428–1435 (2013)
11. Khatun, A., Haque, A.K.M.F., Ahmed, S., Rahman, M.M.: Design and implementation of iris recognition based attendance management system. In: International Conference on Electrical Engineering and Information Communication Technology (ICEEICT), pp. 1–6 (2015)
12. Arsenovic, M., Sladojevic, S., Anderla, A., Stefanovic, D.: FaceTime — Deep learning based face recognition attendance system. In: IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY), pp. 53–58 (2017)
13. Varadharajan, E., Dharani, R., Jeevitha, S., Kavinmathi, B, Hemalatha, S.: Automatic attendance management system using face detection. In: Online International Conference on Green Engineering and Technologies (IC-GET), pp. 1–3 (2016)
14. Jayant, N.K., Borra, S.: Attendance management system using hybrid face recognition techniques. In: Conference on Advances in Signal Processing (CASP), pp. 412–417 (2016)

15. Sony Corporation – FeliCa Website. https://www.sony.net/Products/felica/. Accessed 20 Jan 2022
16. Amazon Web Services. https://aws.amazon.com/. Accessed 20 Jan 2022
17. Japan NFC Reader. https://japannfcreader.tret.jp/. Accessed 20 Jan 2022
18. HTTP Request Shortcuts – Apps on Google Play Store. https://play.google.com/store/apps/details?id=ch.rmy.android.http_shortcuts&hl=en&gl=US. Accessed 20 Jan 2022
19. Schwab, K.: The fourth industrial revolution: what it means, how to respond. World Economic Forum Homepage. https://www.weforum.org/agenda/2016/01/the-fourth-industrial-revolution-what-it-means-and-how-to-respond/. Accessed 20 Jan 2022
20. Hitachi-UTokyo Lab: Society 5.0: A people-centric super-smart society. Springer Nature Singapore Pte Ltd., Singapore (2020). https://doi.org/10.1007/978-981-15-2989-4
21. Smartphone penetration worldwide. https://www.statista.com/statistics/203734/global-smartphone-penetration-per-capita-since-2005/. Accessed 20 Jan 2022
22. Khan M.F.F., Sakamura, K.: Context-aware access control for clinical information systems. In: Proceeding of International Conference on Innovations in Information Technology (IIT), vol. 2012, pp. 123–128, IEEE (2012)
23. Shortcuts User Guide. https://support.apple.com/guide/shortcuts/welcome/ios. Accessed 20 Jan 2022
24. Suica Hompage. https://www.jreast.co.jp/e/pass/suica.html. Accessed 20 Jan 2022
25. PASMO Hompage. https://www.pasmo.co.jp/visitors/en/normalpasmo/. Accessed 20 Jan 2022