p–ISSN: 2723 - 6609 e-ISSN: 2745-5254
Vol. 4, No. 12 December 2023 http://jist.publikasiindonesia.id/

# DEVELOPMENT OF MOBILE-BASED ATTENDANCE MANAGEMENT SYSTEM APPLICATIONS

**Alexander Achmad Khan[1*], Diki Gita Purnama[2*], Arya Maulana Kurniawan[3]**
Paramadina University Jakarta, Indonesia
Email : alexander.khan@students.paramadina.ac.id[1], diki.purnama@paramadina.ac.id[2*], arya.kurniawan@students.paramadina.ac.i[3]

*Correspondence

## ABSTRACT

**Keywords:** attendance; agile; mobile applications; android; human error.

Components in evaluating student academic achievement can be seen from several things, including attendance. Recording attendance or absenteeism in courses is one of the factors for assessing academic performance and student activities in the learning process. Errors in recording attendance are human errors that can be detrimental to students. This research aims to develop a mobile-based application with the Android platform to overcome errors in attendance. Application software development uses the Agile method; the research results are applications that can overcome human error problems, and development using the Agile method can be done more quickly.

## Introduction

Currently, information technology (IT) is developing rapidly; IT also has an impact on improving processes in various sectors, including the education sector. The presence of students in the educational process is essential because attendance is an activity that proves that students carry out lecture activities (Setiawan & Muhaqiqin, 2021); attendance is also one of the evaluations of student academic performance in the learning process (Andriawan & Hamid, 2023).

His research was conducted with a case study at Paramadina University; meetings are held 16 times in a one-semester class, and students must attend lectures as much as 80% of the time. Academics will leave absent students or alpha three or more times without information (Musthofa, 2019). The problem that often occurs is that the attendance recording is not careful, so student attendance is not recorded properly due to human error. Another thing is that attendance is still carried out conventionally and is considered inefficient, so an application is needed to manage student attendance records (Munthe, 2022).

Recording student attendance is essential to describe student performance; bad recording can harm students or not describe actual performance. Errors in recording can generally occur due to human error factors (Rahmat & Yunanto, 2017), such as lecturers needing to record student attendance or students remembering to write attendance. This happens because the recording of events is still manually done by filling out forms or paragraphs when students attend each lecture. Manual or conventional recording is also considered inefficient because it can reduce learning time (Yeni, 2018). An IT-based application is needed to record attendance and manage student attendance well.

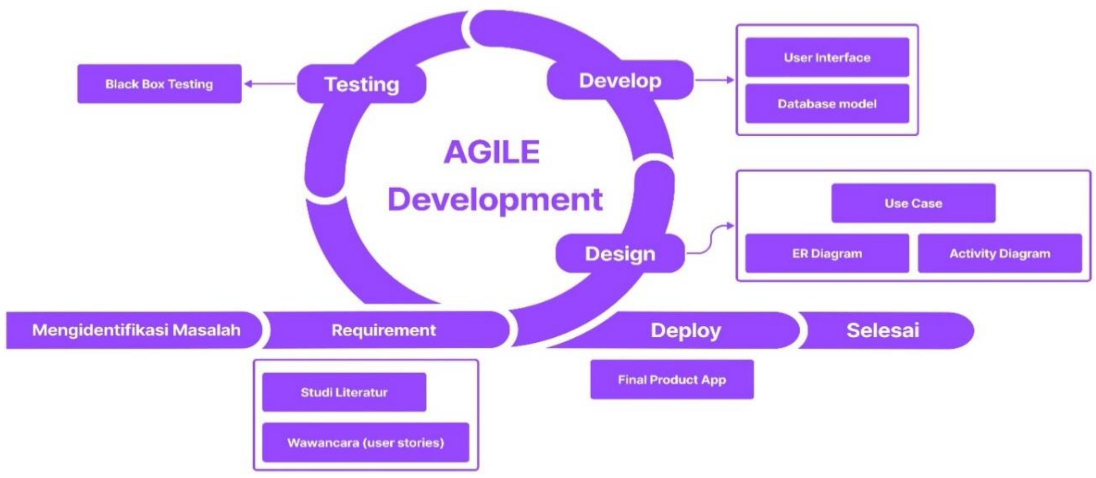
This study aims to develop student attendance recording using a mobile-based application, in this mobile application, using the Android platform. The selection of Android as the application platform, according to Global Stats Stats Counter data with the title "Mobile Operating System Market Share Indonesia" in the Android mobile platform market in Indonesia from June 2022 to June 2023 amounted to 89.66% of the population in Indonesia while for iOS platform users amounted to 10.19% (Tamtelahitu, Sambono, & Unenor, 2021).

Application software development in this study uses agile methods. The agile methodology is a short-term system development type that requires rapid adaptation in the form of any change. In addition, agile is flexible and transparent, improves quality and customer experience, and is predictable compared to other SDLC models (Pratasik & Rianto, 2020).

In Agile, the document's requirements are functional and non-functional features, the results of interview analysis, observation, and literature study (Dinasari, Budiman, & Megawaty, 2020). As for the design stage, such as making UML and the design of the application system, for testing with the black box testing method, namely to ensure the approach made by its structure and function can be used correctly, and at the application deployment stage is carried out by distributing the application based on the Android mobile that has been built.

## Research Methods

The development of attendance system applications in this study, in collecting data using qualitative methods and development methods using Agile development. The data collection process is carried out by observation and semi-structured interviews. At the same time, application development goes through several stages, namely the system analysis stage, the design stage, the application development stage, the testing stage, the application deployment stage, and the revision and evaluation stage. The following are the stages of the research framework for developing attendance system applications.



The research framework above is developing a mobile-based attendance system application in this study, which is divided into six stages. The first stage is identifying problems in the field, namely at Paramadina University. Then, in the requirement stage, data sources are collected from several references to previous research literature studies, Paramadina student interviews with user stories approach, and observation.

The next step of the requirements stage is design, which consists of system design, namely designing, UML modeling with use cases, entity relationship diagrams, and activity diagrams. Then, we move to the development stage, namely, writing the program code using the Flutter framework in Dart language to build mobile user interfaces and the database model using MySQL. After the scene, a test is carried out using a black box with a table to determine the suitability of the features developed for users to repeat the cycle stages. Suppose the last step after the location of the cycle is complete. In that case, the mobile application goes into deployment, namely the distribution of the application based on Android mobile that Paramadina students can use.

## Results and Discussion

### First Level Sprint

The first sprint stage in the Agile process is to perform requirements. In this phase, orientation will solve the needs of the students.

### Requirements (Planning)

The first stage in the agile process is to perform requirements. In this phase, orientation will solve the user's problem, i.e., students. At the planning stage, the author and students will work together to design and build a system (Sevtiana, Cahyadi, & Setiawan, 2023). This stage can be done for several days, depending on the size of the application developed. Moreover, based on the results of the planning analysis, the features that will be built into the attendance application for students in the development of the attendance application include the features described in the table below.

**Table 1**
**Functional features of the application**

| Useful Features of the App |
|---|
| Sign in to the app. |
| View course schedules |
| Conduct course schedule attendance |
| Notification of attendance fails or needs to be on time. |

**Table 2**
**Non-functional features**

| Non-Functional Features of the Application |
|---|
| Orange-themed app system design |
| Use of JSON web token |

After the planning stage has been analyzed and a requirement document is made, enter the agile design stage.

**Design**

The initial stage of Agile design is the research framework, which refers to documentation design, such as designing a UML creation system consisting of use case design, entity relationship diagrams, and activity diagrams.

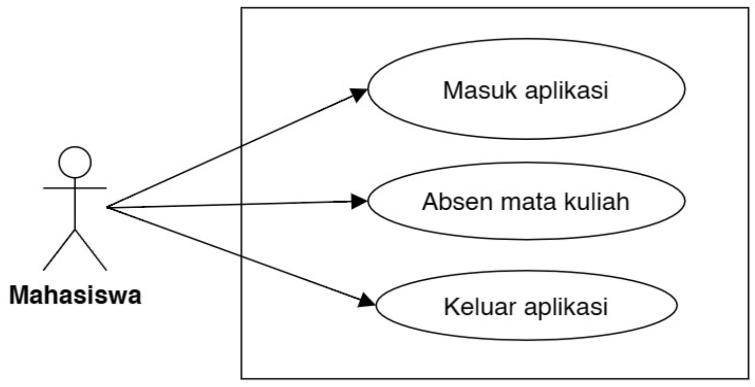


Figure 2. Student use case

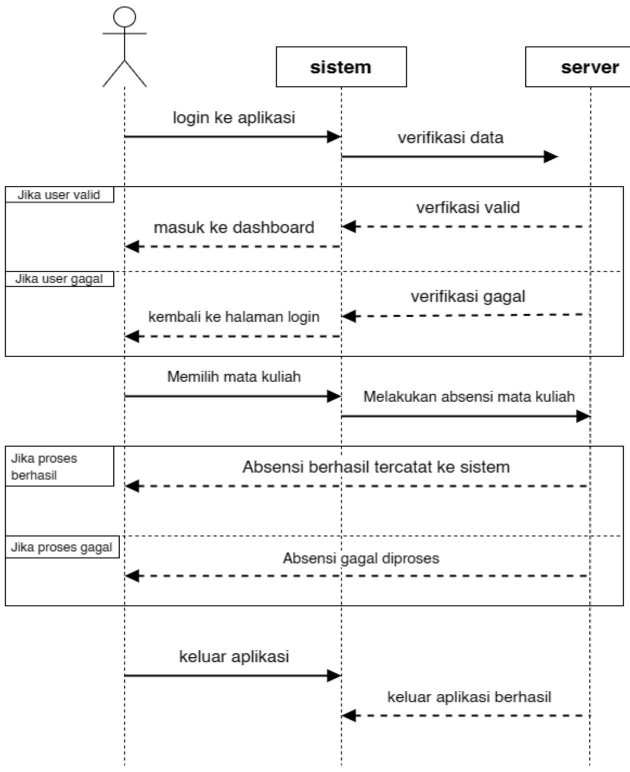
Use case figure 2, which describes the actor who acts in the application and has a role like a student actor. The complete explanation can be seen in the table below.

**Table 3**
**Description of Student Actor**

| Actor | Explanation |
|---|---|
| Student | It describes actors who can access application activities such as course absences. |

**Table 4**
**Description of use case**

| ID | Use Case | Explanation |
|---|---|---|
| UC_01 | Application Login | In use cases, it is an actor activity that students can do to enter the attendance system application. |
| UC_02 | Course Attendance | In this use case, it is an activity carried out by students to make up for the absences of the courses that were followed. |
| UC_03 | Exit Application | In use cases, it is an actor activity that students can do to exit the attendance system application. |



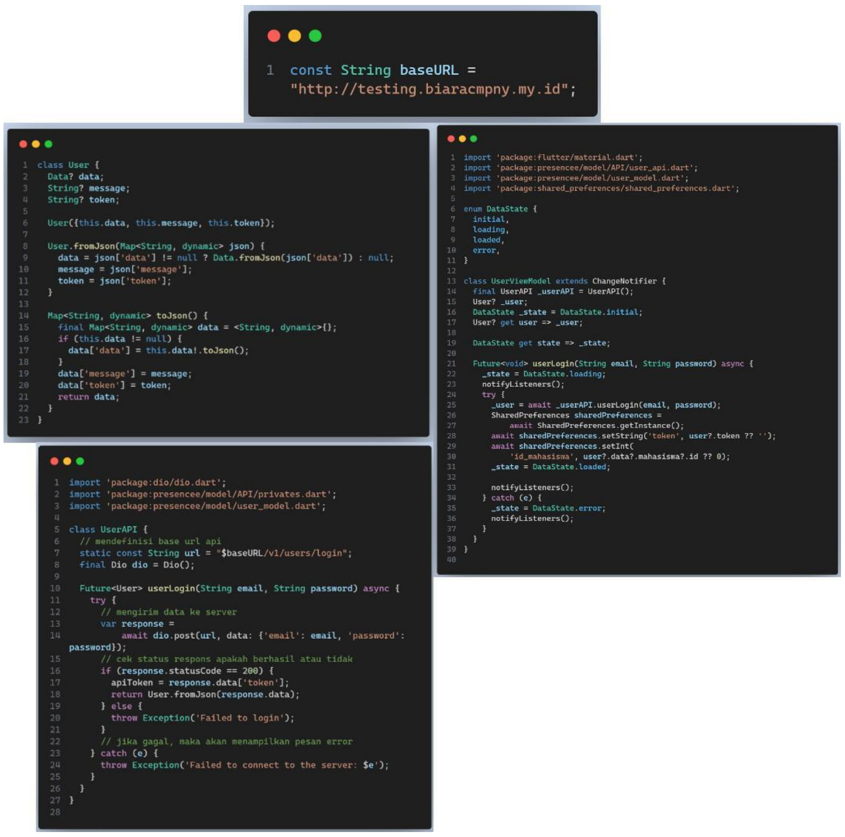
**Gambar 3. Activity diagram**

The activity diagram displays the main activities carried out by students to make attendance in the courses being followed, starting from entering the application that has been registered from the system and then, if you enter the application successfully, entering the display page of the course being followed. Then the student makes attendance if successful, it will be sent to the lecturer and the lecturer will confirm the student's attendance.

**Develop**

After the design stage in the first sprint, develop the appropriate interface system of requirements. The development stage is the stage of the process of writing program code on applications using the Flutter framework. The development also uses Android Studio to help visualize applications like Android emulators. Then, use the dart language with the flutter framework for the program code here (Harahap et al., 2023).

The connection for the database in the application here is created as a model to retrieve user data to perform the entry stage to the application with the help of Postman tools as an aid to retrieve data from the API; this API defines a reusable building block and also allows modular functionality to be incorporated into the end-user application [12]. Figure 4 illustrates the modeling created for an API fetched from a student server if it goes inside its application. First, construct a constant value, which is the destination value of the URL Address for which the data will be retrieved. Then, a user class model is created to validate the data if the user enters a username and password.



```dart
const String baseURL =
  "http://testing.biaracmpny.my.id";
```

```dart
class User {
  Data? data;
  String? message;
  String? token;

  User({this.data, this.message, this.token});

  User.fromJson(Map<String, dynamic> json) {
    data = json['data'] != null ? Data.fromJson(json['data']) : null;
    message = json['message'];
    token = json['token'];
  }

  Map<String, dynamic> toJson() {
    final Map<String, dynamic> data = <String, dynamic>{};
    if (this.data != null) {
      data['data'] = this.data!.toJson();
    }
    data['message'] = message;
    data['token'] = token;
    return data;
  }
}
```

```dart
import 'package:flutter/material.dart';
import 'package:presencee/model/API/user_api.dart';
import 'package:presencee/model/user_model.dart';
import 'package:shared_preferences/shared_preferences.dart';

enum DataState {
  initial,
  loading,
  loaded,
  error,
}

class UserViewModel extends ChangeNotifier {
  final UserAPI _userAPI = UserAPI();
  User? _user;
  DataState _state = DataState.initial;
  User? get user => _user;

  DataState get state => _state;

  Future<void> userLogin(String email, String password) async {
    _state = DataState.loading;
    notifyListeners();
    try {
      _user = await _userAPI.userLogin(email, password);
      SharedPreferences sharedPreferences =
          await SharedPreferences.getInstance();
      await sharedPreferences.setString('token', user?.token ?? '');
      await sharedPreferences.setInt(
          'id_mahasiswa', user?.data?.mahasiswa?.id ?? 0);
      _state = DataState.loaded;

      notifyListeners();
    } catch (e) {
      _state = DataState.error;
      notifyListeners();
    }
  }
}
```

```dart
import 'package:dio/dio.dart';
import 'package:presencee/model/API/privates.dart';
import 'package:presencee/model/user_model.dart';

class UserAPI {
  // mendefinisi base url api
  static const String url = "$baseURL/v1/users/login";
  final Dio dio = Dio();

  Future<User> userLogin(String email, String password) async {
    try {
      // mengirim data ke server
      var response =
          await dio.post(url, data: {'email': email, 'password':
password});
      // cek status respons apakah berhasil atau tidak
      if (response.statusCode == 200) {
        apiToken = response.data['token'];
        return User.fromJson(response.data);
      } else {
        throw Exception('Failed to login');
      }
      // jika gagal, maka akan menampilkan pesan error
    } catch (e) {
      throw Exception('Failed to connect to the server: $e');
    }
  }
}
```

**Figure 4. Program code**

After making user modeling, namely implementing interface features using flutter in the Android emulator, among others made from the interface system, as shown in Figure 5 below.

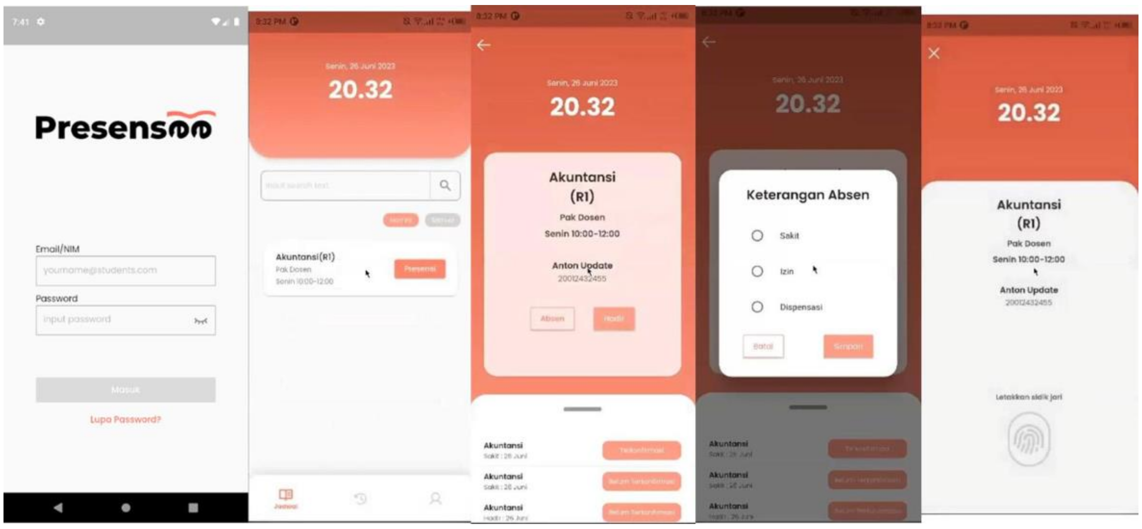


**Figure 5. Application interface**

## Testing

Application testing is carried out using black box testing, which is testing carried out to observe the results of program execution through test data and check each functional software; the tests carried out can be seen in Table 5, which is a test of all features tested.

**Table 5**
**Black box testing**

| Testing Activities | Description of the test scenario | Test Conclusion |
|---|---|---|
| Log into the app. | Students open the application and display an email form and password. | Succeed |
| Conduct course attendance | Students select the courses attended, then select the "present" button. | Succeed |
| Confirm attendance by using fingerprint, location, and photo features | After pressing the "present" button, students can confirm their attendance by asking for photos and fingerprints. | Succeed |
| Perform class absenteeism | Students select courses that are not attended and then select the "absence" button. | Succeed |
| Successful attendance notification | Students will be warned if they have done class attendance. | Succeed |
| Attendance notification if late | Students will be given an announcement when the class is over. | Fail |

## Second Level Sprint

The second stage in the agile iteration process is to continue the previous iteration process. In this phase, orientation will complete the rest of the main features.

## Requirements (Planning)

The following need obtained from the analysis of documents needed for students can be explained in Table 6, which is the second sprint process.

**Table 6**
**Functional features**

| Functional Features of the App |
|---|
| View student profile information. |
| View a class's attendance history. |

## Design

The design carried out in the second stage of the sprint is to add some functional features from students and make additional use cases for students, as explained in Figure 6 below.

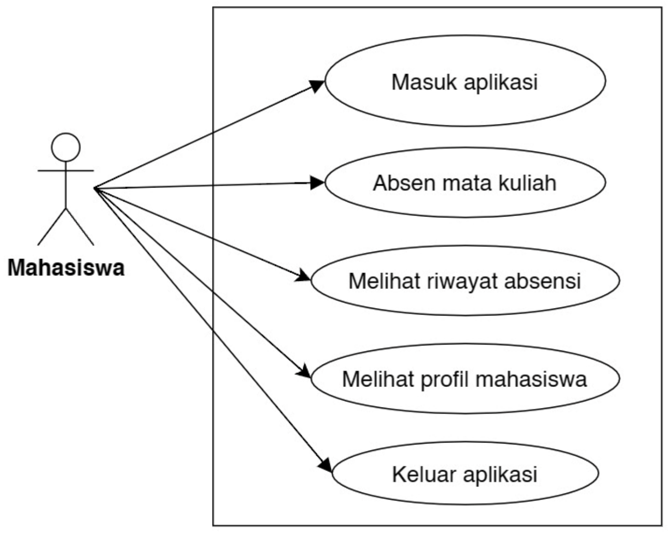


**Figure 6. Student use case**

Figure 6 in the second iteration process shows the use case design used by student actors consisting of use cases entering the application, absent courses, viewing attendance history, viewing student profiles, and exiting the application. Table 7 shows descriptions of student use cases.

**Table 7**
**Student use case description**

| ID | Use Case | Explanation |
|---|---|---|
| UC_01 | Application Login | In use cases, it is an actor activity that students can do to enter the attendance system application. |
| UC_02 | Course Attendance | In this use case, it is an activity carried out by students to make up for the absences of the courses that were followed. |
| UC_03 | View attendance history | For use cases, this is an activity carried out by students to see the entire attendance history of courses that have been followed. |
| UC_04 | View Student Profile | In this use case, it is an activity carried out on students to see student profile data after entering the application. |
| UC_05 | Exit Application | In use cases, it is an actor activity that students can do to exit the attendance system application. |

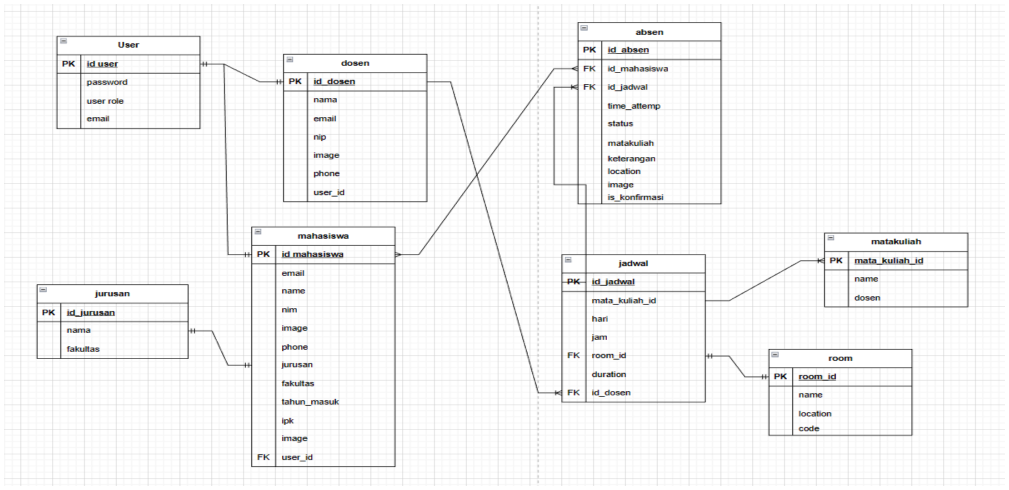
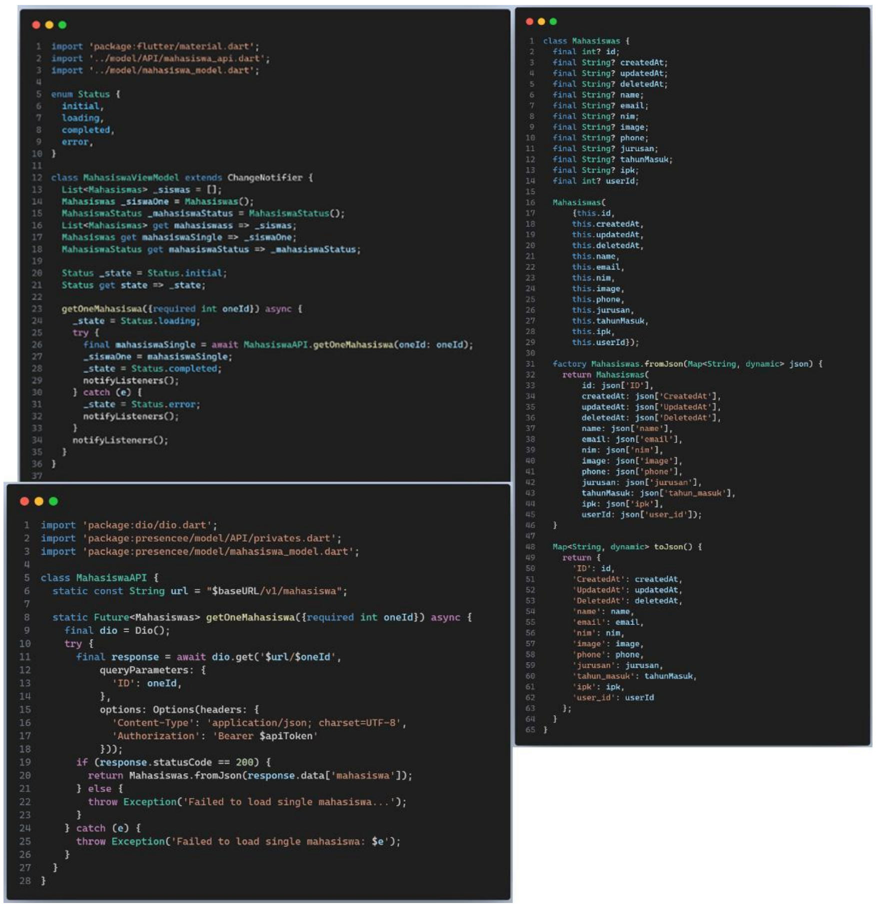


**Figure 7. ERD table**

The entity relationship diagram design can be seen in Figure 7, consisting of 8 tables that will be developed in the attendance application; the table consists of a user table for students to enter the application, then a student table consisting of student biodata, a lecturer table also consists of lecturer biodata, for the department table is specialized separately from student data so that when queried it becomes easier when filtered students. The schedule table is a table that contains courses, course rooms, and lecturers. Moreover, the last table is the timesheet table, which consists of schedules and students. MySQL will be used from the ERD table because the open-source platform is most widely used after Oracle databases.

**Development**

After the user stage of the first sprint is built, the next step is to model the student data used for server requests. This is similar to the previous process for registering students in the application before defining the function of retrieving data from the server, as shown in Figure 8 below.



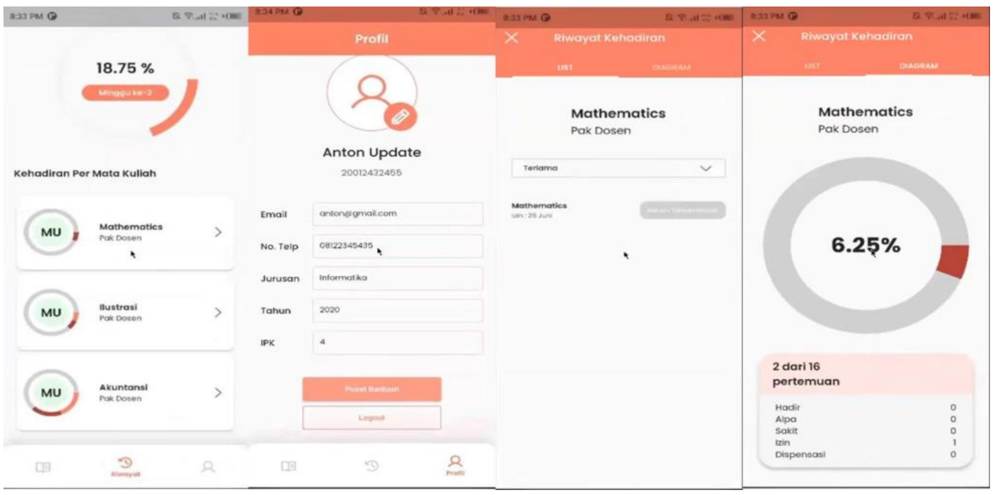
```dart
import 'package:flutter/material.dart';
import '../model/API/mahasiswa_api.dart';
import '../model/mahasiswa_model.dart';

enum Status {
  initial,
  loading,
  completed,
  error,
}

class MahasiswaViewModel extends ChangeNotifier {
  List<Mahasismas> _siswas = [];
  Mahasiswas _siswaOne = Mahasiswas();
  MahasiswaStatus _mahasiswaStatus = MahasiswaStatus();
  List<Mahasismas> get mahasiswass => _siswas;
  Mahasiswas get mahasiswaSingle => _siswaOne;
  MahasiswaStatus get mahasiswaStatus => _mahasiswaStatus;

  Status _state = Status.initial;
  Status get state => _state;

  getOneMahasiswa({required int oneId}) async {
    _state = Status.loading;
    try {
      final mahasiswaSingle = await MahasiswaAPI.getOneMahasiswa(oneId: oneId);
      _siswaOne = mahasiswaSingle;
      _state = Status.completed;
      notifyListeners();
    } catch (e) {
      _state = Status.error;
      notifyListeners();
    }
    notifyListeners();
  }
}

```

```dart
import 'package:dio/dio.dart';
import 'package:presencee/model/API/privates.dart';
import 'package:presencee/model/mahasiswa_model.dart';

class MahasiswaAPI {
  static const String url = "$baseURL/v1/mahasiswa";

  static Future<Mahasiswas> getOneMahasiswa({required int oneId}) async {
    final dio = Dio();
    try {
      final response = await dio.get('$url/$oneId',
          queryParameters: {
            'ID': oneId,
          },
          options: Options(headers: {
            'Content-Type': 'application/json; charset=UTF-8',
            'Authorization': 'Bearer $apiToken'
          }));
      if (response.statusCode == 200) {
        return Mahasiswas.fromJson(response.data['mahasiswa']);
      } else {
        throw Exception('Failed to load single mahasiswa...');
      }
    } catch (e) {
      throw Exception('Failed to load single mahasiswa: $e');
    }
  }
}
```

```dart
class Mahasiswas {
  final int? id;
  final String? createdAt;
  final String? updatedAt;
  final String? deletedAt;
  final String? name;
  final String? email;
  final String? nim;
  final String? image;
  final String? phone;
  final String? jurusan;
  final String? tahunMasuk;
  final String? ipk;
  final int? userId;

  Mahasiswas(
      {this.id,
      this.createdAt,
      this.updatedAt,
      this.deletedAt,
      this.name,
      this.email,
      this.nim,
      this.image,
      this.phone,
      this.jurusan,
      this.tahunMasuk,
      this.ipk,
      this.userId});

  factory Mahasiswas.fromJson(Map<String, dynamic> json) {
    return Mahasiswas(
        id: json['ID'],
        createdAt: json['CreatedAt'],
        updatedAt: json['UpdatedAt'],
        deletedAt: json['DeletedAt'],
        name: json['name'],
        email: json['email'],
        nim: json['nim'],
        image: json['image'],
        phone: json['phone'],
        jurusan: json['jurusan'],
        tahunMasuk: json['tahun_masuk'],
        ipk: json['ipk'],
        userId: json['user_id']);
  }

  Map<String, dynamic> toJson() {
    return {
      'ID': id,
      'CreatedAt': createdAt,
      'UpdatedAt': updatedAt,
      'DeletedAt': deletedAt,
      'name': name,
      'email': email,
      'nim': nim,
      'image': image,
      'phone': phone,
      'jurusan': jurusan,
      'tahun_masuk': tahunMasuk,
      'ipk': ipk,
      'user_id': userId
    };
  }
}
```

**Figure 8. Defining student functions**

Figure 8 defines a student class consisting of a function to retrieve data for one student, then develop an interface for the class attendance history feature and a profile feature page on students, which can be seen in the following figure.



Anton Update
20012432455

| | |
|---|---|
| Email | anton@gmail.com |
| No. Telp | 08122345435 |
| Jurusan | Informatika |
| Tahun | 2020 |
| IPK | 4 |

2 dari 16 pertemuan

| | |
|---|---|
| Hadir | 0 |
| Alpa | 0 |
| Sakit | 0 |
| Izin | 1 |
| Dispensasi | 0 |

**Figure 9. Results of the second stage sprint interface**

## Testing

Testing is repeated to ensure the new feature test runs well; Table 8 shows black box testing.

**Table 8**
**Black box testing functional features**

| Testing Activities | Description of the test scenario | Test Conclusion |
|---|---|---|
| View class attendance history. | After entering the application, students select the second menu, history, to see the entire absence in class. | Succeed |
| View a student's profile. | After entering the application, students select the third menu to view the student profile. | Succeed |

## Conclusion

This research resulted in an application that can record and manage attendance based on Android mobile. In application testing, the functions run well. The development of attendance management applications using agile methods runs efficiently and effectively when system development is short and needs to adapt quickly. Documentation of the role in developing with agile methods needs to be more detailed and complete, but the fulfillment of the user's requirements is essential. Repeated sprints are performed to get results that match the user's needs. Problems regarding errors in student attendance can be overcome by recording this mobile-based attendance.

## Bibliography

Andriawan, Didi, & Hamid, Abdul. (2023). Systematic Literature Review: Penggunaan Dan Manfaat Sistem Gudang Data (Data Warehouse) Di Institusi Perguruan Tinggi. *COMSERVA: Jurnal Penelitian Dan Pengabdian Masyarakat*, *3*(06), 2262–2274. https://doi.org/10.59141/comserva.v3i06.998

Dinasari, Wahyuni, Budiman, Arief, & Megawaty, Dyah Ayu. (2020). Sistem Informasi Manajemen Absensi Guru Berbasis Mobile (Studi Kasus: Sd Negeri 3 Tangkit Serdang). *Jurnal Teknologi Dan Sistem Informasi*, *1*(2), 50–57.

Harahap, Fransiska, Andrianto, Richi, Maimunah, Intan, Daulay, Muhammad Gusti Fhaturrahman, Husein, Muhammad, Sampurna, Muhammad, & Harahap, Patimah. (2023). Pembuatan Aplikasi Absensi Berbasis Flutter untuk Meningkatkan Efisiensi Monitoring Kehadiran. *Jurnal Penelitian Teknologi Informasi Dan Sains*, *1*(3), 85–93.

Munthe, Richa Afriana. (2022). Benefits of Company Management Systems with Combination of ERP (Enterprise Resource Planning). *Journal Research of Social, Science, Economics, and Management*, *1*(6), 610–620. https://doi.org/10.59141/jrssem.v1i6.74

Musthofa, Ahmad Rifki. (2019). *Analisis dan Perancangan Aplikasi Sistem Informasi Akademik Berbasis Web: Studi Kasus Sekolah Dasar Negeri Blado 02*. Program Studi Teknik Informatika FTI-UKSW.

Pratasik, Stralen, & Rianto, Indra. (2020). Pengembangan Aplikasi E-DUK Dalam Pengelolaan SDM Menggunakan Metode Agile Development. *CogITo Smart Journal*, *6*(2), 204–216. https://doi.org/10.31154/cogito.v6i2.267.204-216

Rahmat, Rima Irmayani, & Yunanto, Prasetyo Wibowo. (2017). Perancangan Dan Pengembangan Aplikasi Sistem Informasi Monitoring Perkuliahan Dan Kehadiran Mahasiswa Berbasis Web. *PINTER: Jurnal Pendidikan Teknik Informatika Dan Komputer*, *1*(1), 39–50.

Setiawan, Rodi Putra, & Muhaqiqin, Muhaqiqin. (2021). Sistem Informasi Manajemen Presensi Siswa Berbasis Mobile Studi Kasus SMAN 1 Sungkai Utara Lampung Utara. *Jurnal Teknologi Dan Sistem Informasi*, *2*(3), 119–124.

Sevtiana, Agus, Cahyadi, Fani, & Setiawan, Apriansyah. (2023). Perancangan Aplikasi Undangan Pernikan Online Berbasis Mobile Menggunakan Flutter. *Jurnal Manajemen Sistem Informasi*, *1*(1), 12–15.

Tamtelahitu, Trientje Marlein, Sambono, Jorge, & Unenor, Jekris Ebenhaizer. (2021). Perancangan Sistem Absensi Pintar Mahasiswa Menggunakan Teknik QR Code Dan Geolocation. *JIPI (Jurnal Ilmiah Penelitian Dan Pembelajaran Informatika)*, *6*(1), 114–125. https://doi.org/10.29100/jipi.v6i1.1894

Yeni, Hari. (2018). Pengaruh Penggunaan Aplikasi Sistem Kehadiran Berbasis Web

Terhadap Kinerja Pegawai Pada Kantor Camat Tobadak. *Inspiration: Jurnal Teknologi Informasi Dan Komunikasi*, *8*(2), 69–75. https://doi.org/10.35585/inspir.v8i2.2465