Attendance Management System with Facial Recognition

Overview

The Attendance Management System integrates facial recognition technology to simplify attendance tracking in educational institutions. This comprehensive solution optimizes the process of marking attendance, enhancing efficiency. Through seamless integration of facial recognition, the system offers a reliable and efficient method for monitoring attendance, contributing to smoother administrative processes within educational settings.

Scope

The scope of this project encompasses the development of a user-centric attendance marking application in two distinct phases:

- **Phase 1:** The application will offer a virtual attendance marking service, ensuring the student's presence in class, thus automating the repetitive and tedious work of taking attendance in every class.
- **Phase 2:** A dashboard section will be introduced, where professors can upload messages, notices, or assessments for the students.

Objectives

From Professor's End:

1. Professors will be able to create classrooms and schedule classes.
2. They can create short windows within the period timings in which the students will be able to mark their attendance.
3. They will have access to attendance records of each student they are teaching.
4. Professors can also manually mark attendance of students in case of emergency or error in recognition by the Machine Learning Model.

From Student's End:

1. Students can join the classroom created by the professors of their respective subjects.
2. Students will be able to mark their attendance using the facial recognition model.
3. They will be able to check their attendance percentage in each subject.

Technologies Used

- **Frontend:**
  - Flutter
- **Backend:**
  - Node.js
  - Express.js
  - Firebase
  - MySQL
  - Machine Learning for facial recognition

Facial Recognition

Overview

- Mobilenet Model is used to extract features from images.
- Mobilenet is a lightweight model which can be used on any edge hardware.
- Features extracted are recognized image by measuring distance between extracted features.

Technologies Used

- Tensorflow
- Keras
- Flask

Implementation details

- Images are fetched from firebase database which are used to match with given face
- The URL for the face recognition request is opened via a Flask application. When a request is sent to the recognition endpoint, it returns a JSON response with the details of the detected faces.

-

Technical Overview :

Backend Implementation

Server Configuration

```
const express = require("express");
const bodyParser = require("body-parser");
const dotenv = require("dotenv");
dotenv.config({ path: ".env-local" });

const PORT = process.env.PORT || 8000;
const userroute = require("./router/userroute");
const studentroute = require("./router/studentroute");
const adminroute = require("./router/adminroute");
const classroomroute = require("./router/classroomroute");
const attendanceroute = require("./router/attendanceroute");

const app = express();
app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());
app.use(express.json());

app.use("/admin", adminroute);
app.use("/user", userroute);
app.use("/student", studentroute);
app.use("/classroom", classroomroute);
app.use("/attendance", attendanceroute);

app.listen(PORT, () => {
  console.log("Server is running on ", PORT);
});
```

Dependencies

```
{
  "name": "backend",
  "version": "1.0.0",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server.js",
    "dev": "nodemon server.js"
  },
  "dependencies": {
    "body-parser": "^1.20.2",
    "dotenv": "^16.4.5",
    "express": "^4.19.2",
    "mariadb": "^3.3.0",
    "nodemon": "^3.1.0"
  },
  "description": ""
}
```

Frontend Implementation

Main Dart File

```
Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(options:
DefaultFirebaseOptions.currentPlatform);
  runApp(
    MultiProvider(
      providers: [
        ChangeNotifierProvider(create: (context) => ImageData()),
        ChangeNotifierProvider(create: (context) => ImageData1()),
      ],
      child: MainApp(),
    ),
  );
}

class MainApp extends StatelessWidget {
  MainApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      navigatorKey: kNavigatorKey,
      initialRoute: AppPages.appEntry,
      onGenerateRoute: CustomNavigator.controller,
    );
  }
}
```

Pubspec.yaml(Packages used):

```yaml
name: ams_flutter
description: "A new Flutter project."
publish_to: "none"
version: 0.1.0

environment:
  sdk: ">=3.3.0 <4.0.0"

dependencies:
  equatable: ^2.0.5
  flutter:
    sdk: flutter
  flutter_bloc: ^8.1.4
  flutter_screenutil: ^5.9.0
  font_awesome_flutter: ^10.7.0
  google_nav_bar: ^5.0.6
  google_fonts: ^4.0.4
  internet_connection_checker: ^1.0.0+1
  getwidget: ^4.0.0
  image_picker: ^1.0.7
  firebase_auth: ^4.18.0
  firebase_core: ^2.32.0
  fluttertoast: ^8.2.4
  http: ^0.13.4
  table_calendar: ^3.1.1
  firebase_storage: ^11.7.7
  cloud_firestore: ^4.17.5
  provider: ^6.1.2
  cross_file_image: ^1.1.0

dev_dependencies:
  flutter_test:
    sdk: flutter
  flutter_lints: ^2.0.0

flutter:
  assets:
    - Assets/images/
    - Assets/icons/
  uses-material-design: true
```

Folder Structure

Frontend

```
ams_flutter
├── .dart_tool
├── android
├── Assets
├── build
├── ios
├── lib
│   ├── core
│   │   ├── constants
│   │   ├── error
│   │   └── helpers
│   ├── features
│   │   ├── onboarding
│   │   │   ├── data
│   │   │   ├── datasource
│   │   │   └── presentation
│   │   │       ├── Admin
│   │   │       └── User
│   │   │           ├── bloc
│   │   │           ├── Pages
│   │   │           └── widget
│   ├── route
│   └── services
│       ├── userapiservices.dart
│       ├── firebase_options.dart
│       └── main.dart
├── linux
├── macos
├── web
├── windows
├── .flutter-plugins
└── .flutter-plugins-dependencies
```

Backend

```
Backend
├── controllers
├── helpers
├── middleware
├── model
├── node_modules
├── router
├── .env-local
├── .gitignore
├── package-lock.json
├── package.json
└── server.js
```

Getting Started

Prerequisites

- Node.js
- Flutter SDK
- Firebase account

Installation

1. **Clone the repository:**

   ```
   git clone https://github.com/HackSlashNITP/AMS
   ```

2. **Backend Setup:**

   ```
   cd Backend
   npm install
   npm run dev
   ```

3. **Frontend Setup:**

   ```
   cd ams_flutter
   flutter pub get
   flutter run
   ```

Usage

1. **Professor's End:**
   - Create classrooms and schedule classes.
   - Create short windows for attendance marking.
   - Access attendance records.
   - Manually mark attendance if needed.
2. **Student's End:**
   - Join classrooms created by professors.
   - Mark attendance using facial recognition.
   - Check attendance percentage.