“EMS App” Documentation

Over View:

Our EMS app stands for “Employee Management System”. We build this app for office purpose to track employee status, make official work procedure easier to all.

After use this app an Employee can give attendance through app and check their attendance. Movable report generation, executive can maintain their task and give report to authorize member and also office application form like Leave application, requisition form, convenience form etc. are available.

Introduction:

In our App there are 3 section of Java coding,

- Activity
- Adapter
- Model

1. In Activity section there are 26 activity running,

- MainActivity
- UserSignInActivity
- UserHomePageActivity
- AdminControllerActivity
- AddEmployeeActivity
- EmployeeDetailsActivity
- UserDetailsActivity
- OfficeInAndOutActivity
- CheckEquipmentActivity
- AddEquipmentActivity
- MovableReportActivity
- CheckAttendanceActivity
- CheckMovableEventsActivity
- AttendanceHistoryActivity
- MovementHistoryActivity
- CheckUserAttendanceHistoryActivity
- CheckUserMovementHistoryActivity
- UserAttendanceListActivity
- UserMovementListActivity
- ExecutiveReportActivity
- ExecutionReportHistoryActivity
- ExecutionReportDetailsActivity
- SplashActivity
- UpdateUserInfoActivity

2. In Model Section there are 8 model running,

- Equipment
- Attendance
- Department
- Employee
- Execution
- FeedBack
- Month
- UserRole

3. In Adapter Section there are 11 adapters running,

- AttendanceAdapter
- DepartmentAdapter
- EmployeeAdapter
- EquipmentAdapter
- ExecutionAdapter
- FeedBackReportAdapter
- MonthAdapter
- MovableAdapter
- UserListForAttendanceAdapter
- UserListForMovementAdapter
- UserRoleAdapter

Activity Description:

SplashActivity: A beautiful screen with define some imagination Animation for the Application. There is only some default code to run that animation.

MainActivity: In this activity there is an option to select user role like Admin, General Manager, Managing Director, General Employee etc. and then continue button to Enter SignIn Page.

UserSignInActivity: In this activity User can sign in using their PG id which is provided by the office. There are two type of major user one is Admin and other is User. Admin user name and password is fixed and user only use their PG id and given password for login.

AdminControllerActivity: After using sign in option an Admin can have its Home page where they can see Employee list and, in this activity, they have a button for add Employee.

AddEmployeeActivity: In this activity Admin fill a form for an Employee and add them to the Employee list.

UserHomePageActivity: In this Activity there are 9 options for User which visibility vary to their User Role like, Personal Info, Office in And Out, Equipment report, Movement report, Attendance check, Movement check, Executive report, Service report, Office application

EmployeeDetailsActivity & UserDetailsActivity: In this 2 Activity User and Admin can see the User's Full information and update their information.

OfficeInAndOutActivity: In this Activity there are 2 buttons, one is start and other is finish. When one entered office and clicks Start button immediately the clicked time and location detect by the app and When one exit from office and clicks Finish button immediately the clicked time and location detect by the app.

CheckEquipmentActivity: In this Activity user can check their equipment which they have or got from office.

AddEquipmentActivity: In this Activity User can add their new equipment which they have or got from office.

MovableReportActivity: In this Activity there are 2 buttons, one is start and other is finish and one edit text. When one Move somewhere else for office purpose and clicks Start button immediately the clicked time and location detect by the app and the edit text take the reason of movement and When one entered office after finish task and clicks Finish button immediately the clicked time and location detect by the app.

CheckAttendanceActivity: In this Activity there are different type of User Interface for different UserRole. If User is General employee than there is current month attendance report list and a History button in toolbar one can see month wise attendance by this history button in **AttendanceHistoryActivity**. But if User is HR_Admin than there are extra 2 button self-Attendance and User-Attendance. In user Attendance HR-Admin can see the whole user list in **UserAttendanceListActivity** and also see the individual user attendance month wise in **CheckUserAttendanceHistoryActivity**.

CheckMovableEventsActivity: In this Activity there are different type of User Interface for different UserRole. If User is General employee than there is current month Movement report list and a History button in toolbar one can see month wise Movement report by this history button in **MovementHistoryActivity**. But if User is HR_Admin than there are extra 2 button self-Movement and User-Movement. In user Movement HR-Admin can see the whole user list in **UserMovementListActivity** and also see the individual user Movement month wise in **CheckUserMovementHistoryActivity**.

ExecutiveReportActivity: In UserHomePageActivity if user is not an executive or HR-Admin than there are no execution events exist, else there is an executive report option where an executive can make a report for any support call and submit it to the desire user role like GM, MD, Team-Coordinator, project manager etc. one can see the history of reporting by clicking the history icon in toolbar in **ExecutionReportHistoryActivity**. In the **ExecutionReportHistoryActivity** one can find the report through the month name. if user is HR-Admin than they cannot view the report generation option they only see the report list if they desire of the report.

ExecutionReportDetailsActivity: In this Activity user can see the details of the report and there is also report feedback from the desire user as a list. HR-Admin can send feedback and executive can only see the feedback.

Model Description:

A model class is typically used to "model" the data in your application. For example, you could write a Model class that mirrors a database table, or a JSON

Attendance: In this class we define the attendance details Information which we take from all our user like,

- PG ID
- Date
- Start time
- Finish time
- Start location
- Finish location
- Movement reason for movement Report

There are 4 constructors for 4 different uses, getter setter method and Serializable is implements in this class.

Department: In this class it only returns the department name which contains one constructor and one getter and one setter method. This class use for Department Dropdown name returning.

Equipment: In this class it only returns the Array list of Equipment's names which contains one constructor and one getter which return list of equipment's names and one setter method. This class use for Equipment Dropdown name returning.

Employee: In this class we define the employee details Information which we take from all our user like,

- Name
- Email
- Phone
- NID No
- Current city
- Current location
- Village
- Upazilla
- Zilla
- Division
- User Role
- PG Id
- Department
- Designation
- Joining date
- Password

There are 1 constructor for user data registration and for all individual data there are getter and setter method. Serializable is implements in this class for reuse the data.

Execution: In this class we define the employee details Information which we take from all our user like,

- Array list of access person
- Caller name
- Caller contact
- Calling date
- Caller Organization
- Caller address
- Caller quire
- Our given advice
- Remarks

There are 1 constructor for execution data uses and for all individual data there are getter and setter method. Serializable is implements in this class for reuse the data.

FeedBack: In this class we define the feedback details Information which we take from all our user like,

- Feedback time
- Feedback sender
- Feedback message

There are 1 constructor for feedback data uses and for all individual data there are getter and setter method. Serializable is implements in this class for reuse the data.

Month: In this class it only returns the month name which contains one constructor and one getter and one setter method. This class use for Month Dropdown name returning.

UserRole: In this class it only returns the month name which contains one constructor and one getter and one setter method. This class use for UserRole Dropdown name returning.

Adapter Description:

An Adapter object acts as a bridge between an Adapter View and the underlying data for that view. The Adapter provides access to the data items. The Adapter is also responsible for making a View for each item in the data set.

Attendance Adapter: In this adapter we define an array list for user attendance from **Attendance** class. We also define a custom view named **attendance_view.** We also create an object of Attendance class which contains attendance information for individual. We get the data and set the information to **attendance_view** xml file. To see self-attendance details there is item click option then user go to **SelfAttendanceDetailsActivity**.

Department Adapter: This adapter is made for department name showing in a dropdown menu. In this adapter we define an array list for user Department from **Department** class

Month Adapter: This adapter is made for month name showing in a dropdown menu. In this adapter we define an array list for Month name from **Month** class

UserRole Adapter: This adapter is made for UserRole name showing in a dropdown menu. In this adapter we define an array list for user role from **UserRole** class

Employee Adapter: In this adapter we define an array list for user information from **Employee** class. We also define a custom view named **employee_list.** We also create an object of Employee class which contains employee information for individual. We get the data and set the information to **employee_list** xml file. To see employee information in details there is item click option then user go to **EmployeeDetailsActivity**.

Equipment Adapter: In this adapter we define an array list for user equipment information from **Equipment** class. We also define a custom view named **equipment_view.** We also create an object of Equipment class which contains equipment information for individual. We get the data and set the information to **equipment_view** xml file.

Execution Adapter: In this adapter we define an array list for executive report information from **Execution** class. We also define a custom view named **execution_report.** We also create an object of Execution class which contains executive report information for individual. We get the data and set the information to **execution_report** xml file. To see executive report information in details there is item click option then user go to **ExecutionReportDetailsActivity**.

FeedBackReport Adapter: In this adapter we define an array list for feedback of access user for execution information from **FeedBack** class. We also define a custom view named **feedback_view.** We also create an object of **FeedBack** class which contains feedback of access user for execution information for individual. We get the data and set the information to **feedback_view** xml file.

Movable Adapter: In this adapter we define an array list for movement report information from **Attendance** class. We also define a custom view named **attendance_view.** We also create an object of Attendance class which contains movement report information for individual. We get the data and set the information to **attendance_view** xml file. To see movement report information in details there is item click option then user go to **SelfMovableDetailsActivity**

UserListForAttendance Adapter: In this adapter we define an array list for user information from **Employee** class. We also define a custom view named **employee_list.** We also create an object of Employee class which contains employee information for individual. We get the data and set the information to **employee_list** xml file. To see individual user's attendance information in details there is item click option then user go to **UserAttendanceListActivity**.

UserListForMovement Adapter: In this adapter we define an array list for user information from **Employee** class. We also define a custom view named **employee_list.** We also create an object of Employee class which contains employee information for individual. We get the data and set the information to **employee_list** xml file. To see individual user's movement information in details there is item click option then user go to **UserMovementListActivity**.