

ENTERPRISE LEAVE AND PAYROLL MANAGEMENT SYSTEM

Abstract

Leave management system plays a major role in payroll system. A system which helps in recording, managing, and tracking employees leave requests.

The existing system is a web-based solution built on .NET and Oracle Database. Although the interface is compatible for personal computers, laptops, and mobile devices, it does not provide a better user-friendly experience for mobile users and doesn't necessarily provide faster access to leave related transactions as it needs better network bandwidth. This raised the need for a mobile application which empowers the user with ability to request and accept leaves on-the-go with the simple, user-friendly application design with limited network bandwidth. The main aim of the project is to develop an android application which contributes to the existing leave management web interface of the payroll system.

Objectives of the expected android application are: To develop an application which provides better User Experience To design efficient User Interface which increases the performance of the application process while ensuring efficient usage of memory and reduce heavy usage of resources such as processing power and storage

Deliver a completely automated, self-reliant application which can work seamlessly with the existing web application without causing any interoperability issues.

Keywords: Leave Management, Android Development, .NET, Oracle Database, PHP
Page

Authors

Sri Chakra Raj

Information Technology
Chaitanya Bharathi Institute of Technology
ugs19114_It.Sri@Cbit.Ac.In

B. Veera Jyothi

Information Technology
Chaitanya Bharathi Institute of Technology
jyothibadnal@gmail.com

L. Suresh Kumar

Mechanical Engineering
Chaitanya Bharathi Institute of Technology
lsureshkumar_Mech@Cbit.Ac.In

P. V. R. Ravindra Reddy

Mechanical Engineering
Chaitanya Bharathi Institute of Technology
ravindrareddypr_Mech@Cbit.Ac.In

PROJECT REQUIREMENTS

This Android Application was built on Higher End Device to ensure that the development process is carried out smoothly, which requires quick engagement of various supporting software for testing and debugging the application at every stage.

SOFTWARE REQUIREMENTS

- Android Studio
- Visual Studio
- PHP
- Oracle XE Database

HARDWARE REQUIREMENTS

- Intel Core i5 9th Gen
- 16GB DDR4 RAM
- 8GB Nvidia GTX 1650 MaxQ Graphics

I. INTRODUCTION

An android app for leave management is a software application that is designed to streamline the process of requesting, tracking, and approving leave requests for employees. It allows employees to easily submit leave requests through their mobile devices, and it also provides a convenient way for HR managers and supervisors to review and approve or deny these requests. The app typically includes features such as a calendar view, notifications, and the ability to attach documents or notes to leave requests. It can also provide real-time updates on the status of leave requests and allow employees to see their remaining leave balances. By automating the leave management process, the android app can help to improve efficiency and reduce the workload for HR staff, while also providing employees with a convenient way to manage their time off.

This Leave Management project is developed using Android Studio and PHP Server Pages. The application implements the leave management system within 2 levels of hierarchy - i.e., Employee and Manager. It is designed to notify the requestor as well as the requestee at each step from leave request to acceptance/cancellation. The application provides the following interfaces which will be detailed explained in the following section:

- Login View
- Profile view - include employee details such as Balance leaves, Reporting Manager Name, Department of Work.
- Leave Request View
- Leave Approval View (for Hierarchy Level - 2)

II. IMPLEMENTATION

The developed application contains several functionalities. In this section, the entire application is explained through basic segments which define each functionality of the application.

1. Login Segment

The Login Interface provides two input fields

- User ID: Employee ID
- Password: Minimum 8 Characters Field

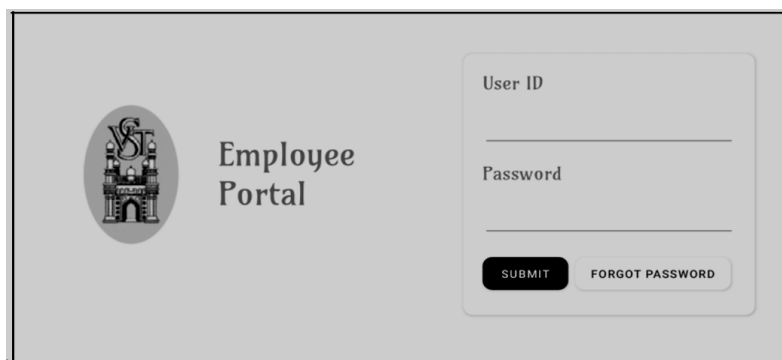


Figure 1: Login View

Through this interface an employee can login to perform tasks such as request for a leave, check their profile for availability of leaves, approve or cancel leave requests based on their role. This interface also provides a forgot password option, through which employee can reset their password after proper validation. The validation is processed through a 5-digit OTP (one-time-password) sent to the registered email address of the employee.

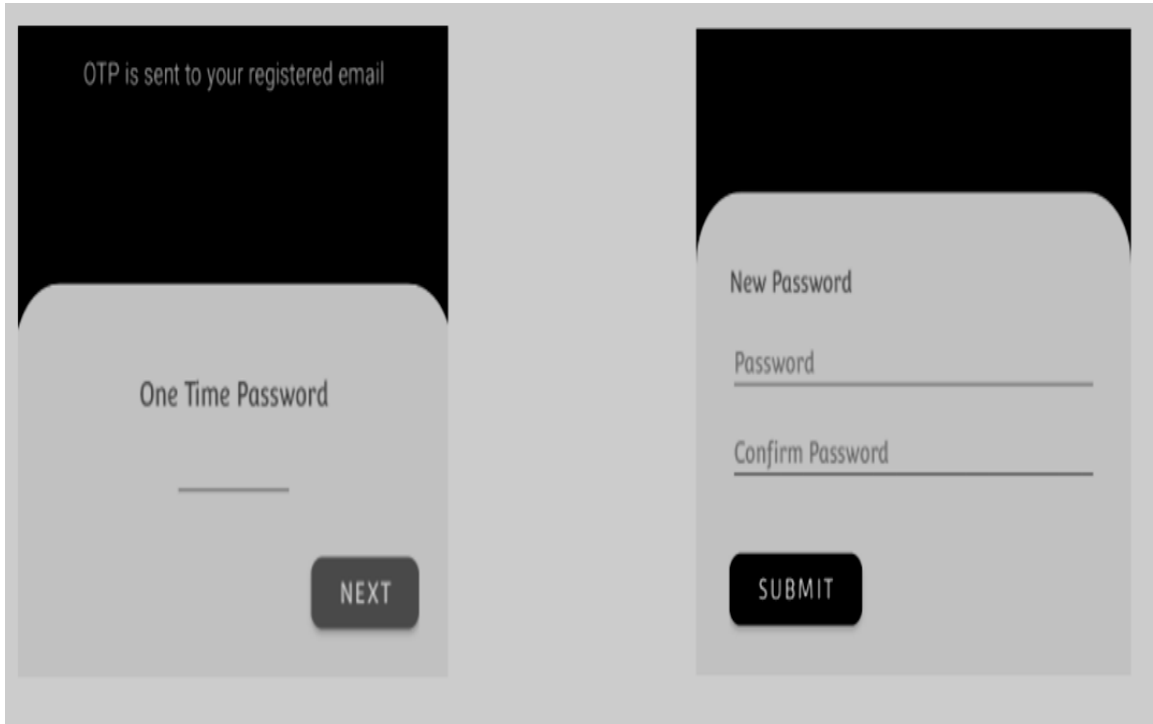


Figure 2: Login Interface

Figure 3: Forgot Password

III. POST LOGIN SEGMENT

Post Login interface provides options such as Apply for Leave, View Profile and Approve Leave (as per role). Employee can also view their current pending leaves (see Fig 4).

The small component at the start displays employee information such as employee name and employee id. The pending leave section is a scrollable (recycler) view which scales as per the need. Recycler view helps in reducing the physical load on the interface rendering.

Each component of the pending leave section (see Fig 5) presents the details of the leave such as Leave Start Date and Leave End Date.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/background_main"
    android:padding="5dp"
    tools:context=".ProfileActivity">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:background="@color/cream_white"
        android:minHeight="?attr/actionBarSize"
        android:theme="?attr/actionBarTheme"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:title="Profile"
        app:titleTextColor="@color/vst_blue_toolbar">

        <ImageView
            android:id="@+id/imageView3"
            android:layout_width="40dp"
            android:layout_height="40dp"
            android:layout_gravity="center_vertical|end"
            app:srcCompat="@drawable/image" />
    </androidx.appcompat.widget.Toolbar>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Figure 4: Backend Code Sample for Post Login



Figure 5: Post Login Interface

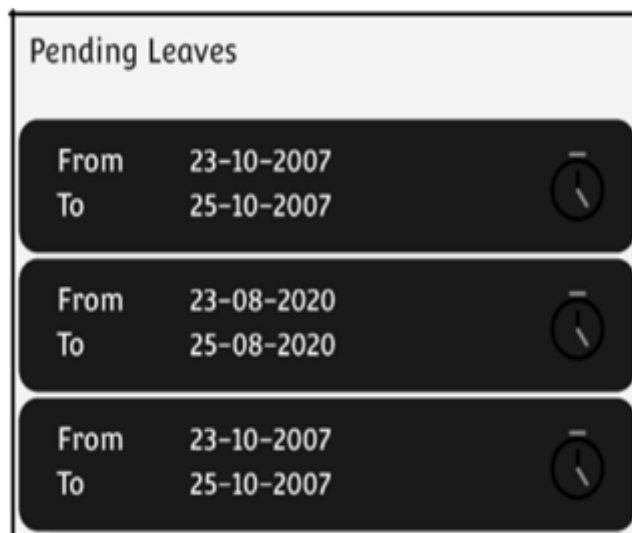


Figure 6: Pending Leaves Section

IV. PROFILE SEGMENT

Profile Interface displays all information of the employee related to leave management system such as employee name, employee's reporting authority, employee's department. (See Fig3.2)

Additionally, it provides the available leaves under two categories AL – Annual Leaves and SL – Sick Leaves. This helps employee while availing leaves/requesting leaves. Employee can change his password; it follows the same procedure of identity verification through registered email. This helps in always maintaining security. He/she may logout of their account from the same interface.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/background_main"
    android:padding="5dp"
    tools:context=".ProfileActivity">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:background="@color/cream_white"
        android:minHeight="?attr/actionBarSize"
        android:theme="?attr/actionBarTheme"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:title="Profile"
        app:titleTextColor="@color/vst_blue_toolbar">

    <ImageView
        android:id="@+id/imageView3"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:layout_gravity="center_vertical|end"
        app:srcCompat="@drawable/image" />
```

Figure 7: Frontend XML Code Sample for Profile User Interface

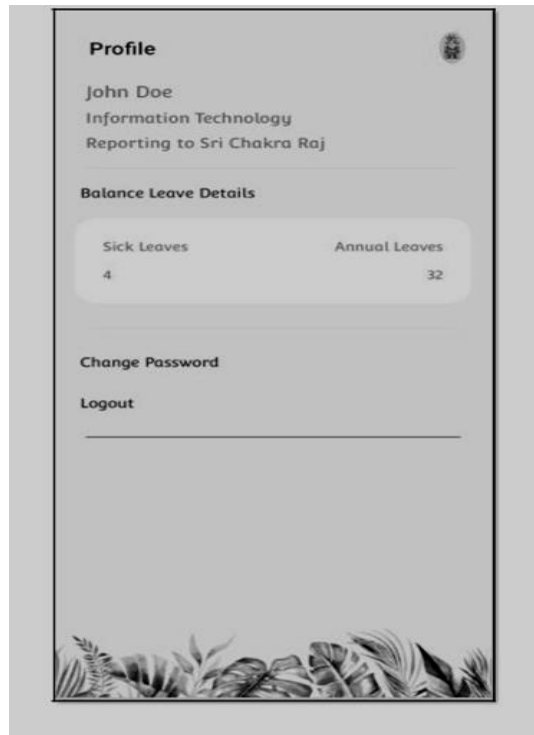


Figure 8: Profile Section

V. LEAVE REQUEST SEGMENT

Employees of the company can apply for leaves through the leave request interface. This interface has 4 simple inputs

- Leave Start Date
- Leave End Date
- Leave Type (AL/SL)
- Reason for the Leave

Each input field is designed for the best user experience, for e.g., Leave start and end date, is linked with a calendar pick which helps in picking dates fast and helps in handling the dates across forms easier. There are few conditional checks for the leave request form based on the type of leave.


```
from.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        final Calendar fcldr = Calendar.getInstance();  
        int fdays = fcldr.get(Calendar.DAY_OF_MONTH);  
        int fmonth = fcldr.get(Calendar.MONTH);  
        int fyear = fcldr.get(Calendar.YEAR);  
        fpicker = new DatePickerDialog( context: Lrequest.this,  
            new DatePickerDialog.OnDateSetListener() {  
                @Override  
                public void onDateSet(DatePicker view, int year, int monthOfYear, int dayOfMonth) {  
                    fromdate.setText(dayOfMonth + "/" + (monthOfYear + 1) + "/" + year);  
                }  
            }, fyear, fmonth, fdays);  
        fpicker.show();  
    }  
});  
to.setOnClickListener(new View.OnClickListener(  
    @Override  
    public void onClick(View v) {  
        final Calendar tcldr = Calendar.getInstance();  
        int tdays = tcldr.get(Calendar.DAY_OF_MONTH);  
        int tmonth = tcldr.get(Calendar.MONTH);  
        int tyear = tcldr.get(Calendar.YEAR);  
        tpicker = new DatePickerDialog( context: Lrequest.this,  
            new DatePickerDialog.OnDateSetListener() {  
                @Override  
                public void onDateSet(DatePicker view, int year, int monthOfYear, int dayOfMonth) {  
                    todate.setText(dayOfMonth + "/" + (monthOfYear + 1) + "/" + year);  
                }  
            }, tyear, tmonth, tdays);  
        tpicker.show();  
    }  
});
```

Do not concatenate text displayed with setText. Use resource string with placeholders.
Provide feedback on this warning Alt+Shift+Enter More actions... Alt+Enter

int year
year – the selected year

Figure 9: Backend Code Sample for Date Selection for Leave Request Segment

Leave Request

Employee Details
John Doe ID - XXXX

From 23/12/2022

To 22/12/2022

Leave Type AL

Reason
Severe Cold

Invalid dates!

Figure 10: Leave Request Section

2. For Annual Leave:

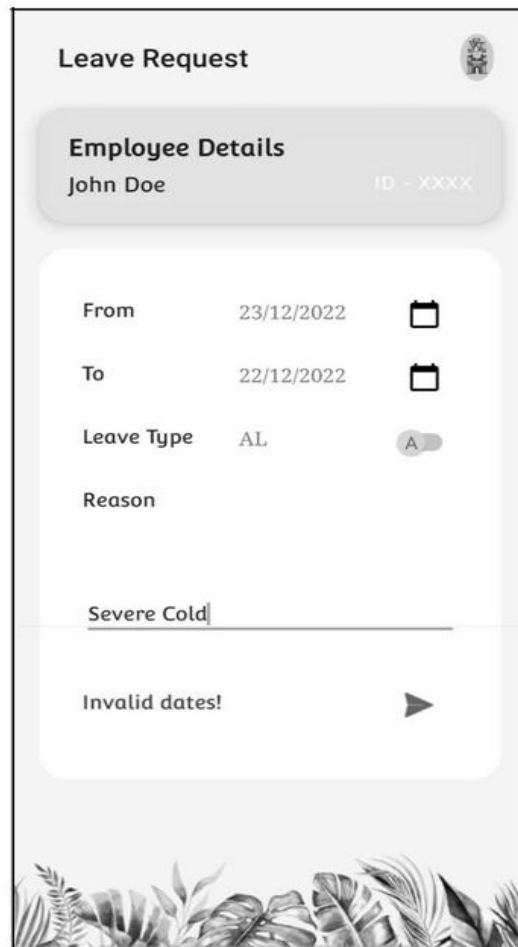
- Start Date should not be older than current date.
- Start and End Date should exist in the same financial year.
- Start date should always be older than end date.
- Reason for the leave must be included.
- Total number of days for the leave must not exceed the available number of leaves.

3. For Sick Leave:

- Total 8 Sick leaves are provided.
- Need a medical report for examining the sickness if crosses the limit of available leaves.
- Date restrictions are similar to annual leaves.
- Reason for leave must be included.

The error pages depicting the fundamental conditions are provided in the next page.

4. Error Pages for the Leave Request Segment



The screenshot displays a mobile application interface for a 'Leave Request' form. At the top, the title 'Leave Request' is visible. Below it, the 'Employee Details' section shows 'John Doe' and 'ID - XXXX'. The form fields include 'From' (23/12/2022), 'To' (22/12/2022), 'Leave Type' (AL), and 'Reason' (Severe Cold). A red error message 'Invalid dates!' is displayed at the bottom of the form, indicating that the start date is later than the end date. The form also features a right-pointing arrow button.

Figure 11: Invalid Date Selection Error

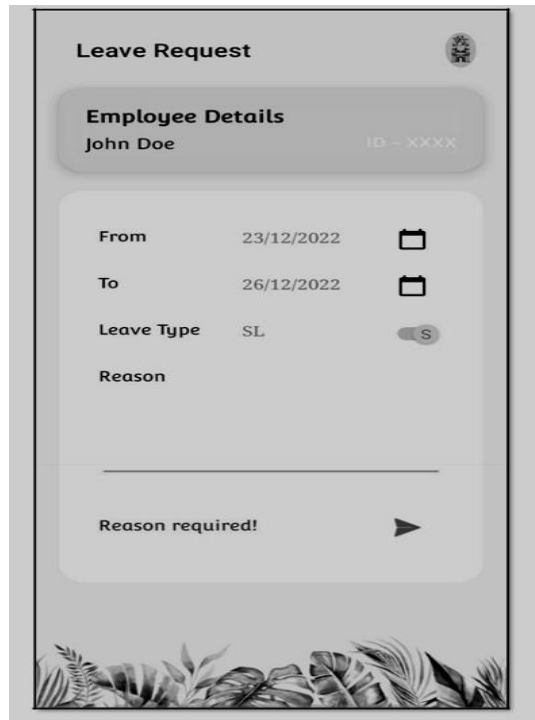


Figure 12: Reason Not Provided Error

VI. LEAVE APPROVAL SEGMENT

Leave Approval Interface displays the list of all leaves submitted to the employee from their subordinates. It is a recycler view component, which optimizes the memory consumed by the interface by destroying the components which are not active, it provides a scrollable interface (see Fig 13).

The fundamental block of the leave approval interface is the leave component (see Fig 14). The leave component displays the details of the leave such as

- Employee ID
- Employee Name
- Total Days of Leave
- Type of Leave

```
public LeaveAppViewHolder onCreateViewHolder(@NonNull @NotNull ViewGroup parent, int viewType) {  
    LayoutInflater inflater=LayoutInflater.from(parent.getContext());  
    View view=inflater.inflate(R.layout.list_approval_viewitem,parent, attachToRoot: false);  
    return new LeaveAppViewHolder(view);  
}  
  
@Override  
public void onBindViewHolder(@NonNull @NotNull LeaveAppAdapter.LeaveAppViewHolder holder, int position) {  
    int employee_code=data[position].employeecode;  
    String employee_name=data[position].employeeName;  
    String leavereason=data[position].leave_reason;  
    int days=data[position].leave_days;  
    String Leavetype=data[position].leave_type;  
    String apply_date=data[position].applied_date;  
    String from_date=data[position].from_date;  
    String to_date=data[position].to_date;  
    holder.empcode.setText(employee_code+"");  
    holder.empname.setText(employee_name);  
    holder.reason.setText(leavereason);  
    holder.days.setText(days+"");  
    holder.leavetype.setText(Leavetype);  
    holder.Fromdate.setText(from_date);  
    holder.Applydate.setText(apply_date);  
    holder.Todate.setText(to_date);  
    ///  
    final boolean isExpanded = position==mExpandedPosition;  
    holder.details.setVisibility(isExpanded?View.VISIBLE:View.GONE);  
    holder.itemView.setActivated(isExpanded);  
    if (isExpanded)  
        previousExpandedPosition = position;  
    holder.itemView.setOnClickListener(new View.OnClickListener() {  
        @Override  
        public void onClick(View v) {  
            mExpandedPosition = isExpanded ? -1:position;  
            notifyItemChanged(previousExpandedPosition);  
        }  
    });  
}
```

Figure 13: Backend Code for RecyclerView which contains implementation of onCreate View Holder and onBind View Holder for the custom defined adapter

Employee ID	Employee Name	Reason	Approval Status
1000	Sri Chakra Raj Pyaraka	Sick	SL
1012	Tanmay	Holiday	AL
1009	Rajendra	Family Issues	AL
1023	Anudeep	Travel	AL
1030	Sravan	Operation	SL
1000	Sri Chakra Raj Pyaraka	Sick	SL
1012	Tanmay	Holiday	AL
1009	Rajendra	Family Issues	AL
1023	Anudeep	Travel	AL
1030	Sravan	Operation	SL
1000	Sri Chakra Raj Pyaraka	Sick	SL

Figure 14: Leave Approval Section

Upon selecting a leave request component, it expands to provide further information such as

- Date of Request
- Start Date of Leave
- End Date of Leave

The expanded view also provides two more options Approve and Cancel Approve which describe the actions that can be taken on the leave request.

- **Approve** – Approves the leave request
- **Cancel Approve** – Cancels the leave request

Each action triggers an email to the leave requestor notifying the status of the leave request or the response for the leave request.

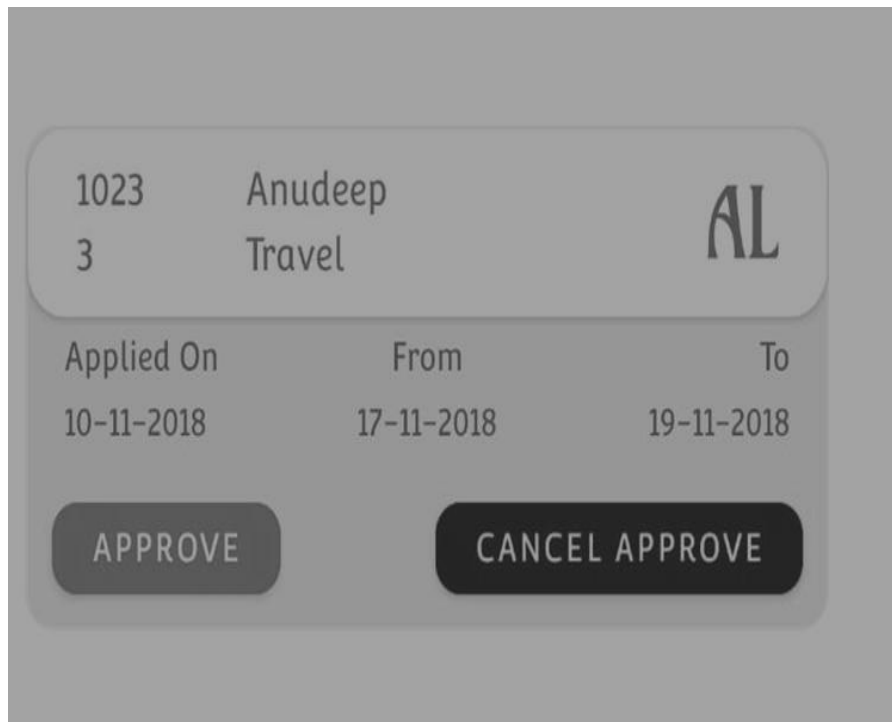


Figure 15: Leave Request Component

VII. AUTO EMAIL NOTIFICATION SEGMENT

When a leave request is sent to the higher authority, (manager here), it will notify the manager through mail with the leave details such as Employee information and Leave Type, Total days, and whether or not doctor consultation is required. (See Fig 16)

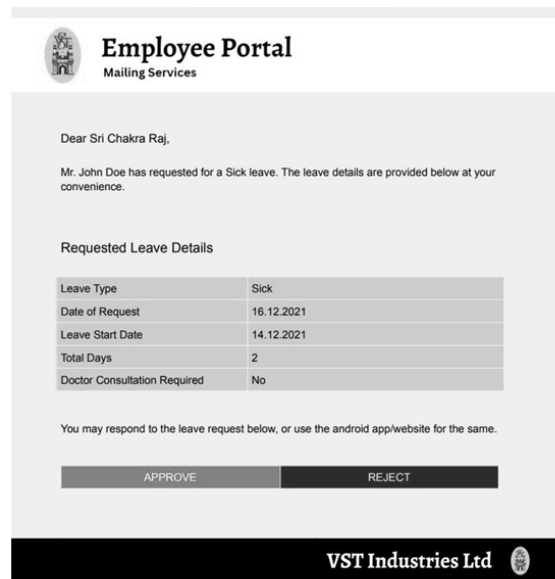


Figure 16: Automated Email sent to Authority for Approval of a leave request submitted by a subordinate

Password change/reset form triggers an email for verification of the user. The mail is sent to the registered email of the employee with a 5-digit one time password. (See Fig 17)

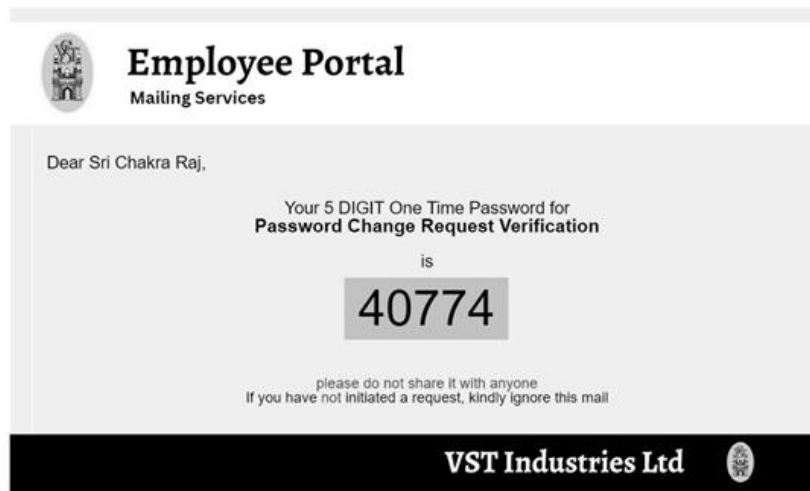


Figure 17: Automated Email of OTP for Employee Verification

VIII. CHALLENGES & SOLUTIONS

Challenges bear an integral part of development process of every application. This internship project had its own strong challenges. Due to the rapid growth in technology, various basic libraries used for integration have been updated with advanced security features. This caused termination of the supporting libraries, making it very hard to research for alternatives.

Few of the Challenges in the internship are mentioned below to support the future projects in android development.

- 1. Database Connectivity:** Database connection is important part of the entire development process. Although there are several serverless applications built on cloud platforms such as Google Firebase, but this problem is faced whenever traditional server-based applications are developed which depend on on-premises resources such as hosting server, database like Oracle XE/MySQL that need programmable adapters for communication.

After several updates from JAVA JDK 7, the traditional method of using JDBC/OJDBC Drivers directly from the android application code has been deprecated due to security reasons. There was a high chance of injection attacks (if user queries are not processed) and database credential data leakage with a little effort of applying patches/rewrite dependency files of the application.

I have found a safe and working alternative for connecting and querying data from a database. This has also increased the performance of the application by reducing the overall active thread processing time.

- 2. User Interface Optimization:** The user interface should be always lightweight and responsive, by responsive I mean, it must always be quick in adapting and user must be able to understand the task being processed behind. This every aspect of the user interface affects the experience largely.

While I was developing the complex User Interfaces for various segments of the project, due to the use of multiple components including few nested components, the application consumed lot of memory to only load the components. A major issue of repetitive loss of frames and also crashing has took place due to this uncontrolled excessive usage of memory and processing power for UI components loading and all the other additional background tasks such as data querying, other code functionalities.

After extensive research and analysis of the above-mentioned issues, I have found multithreading is the best choice for minimizing the resource usage. Main Thread for an android application is used for UI purposes ideally, so isolating all the functionalities, query tasks from the main thread into another thread can reduce the hardware resource utilization of the device.

IX. CONCLUSION

The Leave Management System android app was a successful project that allowed employees to request and track their leave from their mobile devices. The app streamlined the leave request process and made it easier for employees to stay informed about the status of their requests. Overall, the app was well-received by both employees and the HR department, and it helped to improve the efficiency and accuracy of the leave management process. The Leave Management System android app was a valuable addition to the company's HR operations, and it has been an asset in managing leave requests and tracking employee time off.

REFERENCES

- [1] Android Studio Documentation: <https://developer.android.com/studio/intro>
- [2] Volley Library Basics: <https://google.github.io/volley/>
- [3] Stack Overflow Assistance: <https://stackoverflow.com/questions/69494630/java-sql-driveraction-not-found-on-path>
- [4] Post about “Problem with JDBC Driver” on Stack Overflow:
<https://stackoverflow.com/questions/15419415/cannot-load-jdbc-driver-classnotfoundexception>
- [5] Post about “JDBC vs Webservices for Android” on Stack Overflow:
<https://stackoverflow.com/questions/15853367/jdbc-vs-web-service-for-android>
- [6] Android Multithreading: <https://developer.android.com/topic/performance/threads>
- [7] Multithreading In-depth Information - <https://www.toptal.com/android/android-threading-all-you-need-to-know>
- [8] Oracle XE Database Documentation: https://docs.oracle.com/cd/E17781_01/index.html