REIMBURSEMENT APP (VEE TECH)

**A PROJECT REPORT**

***Submitted by***

ABHINDRA G (Reg.No.1920106005)

KAAMAL V N (Reg.No.1920106031) MAHISHKANNA M (Reg.No.1920106042) MEGUL RAJ S (Reg.No.1920106047)

NAGARAJ K A (Reg.No.1920106052)

NEHA K (Reg.No.1920106057) HARIKRISHNA R (Reg.No.1920106704)

***in partial fulfillment for the award of the degree of***

BACHELOR OF TECHNOLOGY

**IN INFORMATION TECHNOLOGY**

**SONA COLLEGE OF TECHNOLOGY**

ANNA UNIVERSITY: CHENNAI 600 025

**NOVEMBER 2023**

**BONAFIDE CERTIFICATE**

Certified that this project report **“REIMBURSEMENT APP (VEE TECH)”** is the bonafide work of **“ABHINDRA G(1920106005), KAAMAL V N (1920106031), MAHISHKANNA M(1920106042), MEGUL RAJ S (1920106047), NAGARAJ K A(1920106052), NEHA K(1920106057),**

**HARIKRISHNA R (1920106704)”** who carried out the project work under my supervision.

|  |  |
| --- | --- |
| **SIGNATURE** | **SIGNATURE** |
| (Dr.J.Akilandeswari) | (Dr.J.Akilandeswari) |
| Professor | Professor |
| **HEAD OF THE DEPARTMENT** | **SUPERVISOR** |
| Department of Information Technology | Department of Information Technology |
| Sona College of Technology | Sona College of Technology |
| Salem-636 005 | Salem-636 005 |

Submitted for Mini Project viva voice examination held on

……………………………..

|  |  |
| --- | --- |
| **INTERNAL EXAMINER** | **EXTERNAL EXAMINER** |

**ABSTRACT**

The objective of this project is to introduce the "Reimbursement App," a comprehensive web application developed to revolutionize the reimbursement process within organizations. The primary goal of this application, built using React.js, Node.js, and SQL, is to simplify and streamline the often complex and time-consuming process of reimbursing employees for expenses, including travel and broadband costs. The Reimbursement App addresses the challenges associated with traditional reimbursement systems by providing a user-friendly and efficient platform. It automates and digitizes the workflow, reducing the administrative burden on companies and ensuring that employees receive their reimbursements promptly. Key features of the Reimbursement App include an intuitive user interface that facilitates easy submission of reimbursement requests, real-time tracking to keep employees and administrators informed throughout the process, and the ability to customize the application to align with specific company policies. It leverages the power of SQL for data integrity and security, enabling comprehensive reporting and auditing capabilities. By implementing the Reimbursement App, organizations can optimize their reimbursement processes, freeing up valuable time and resources for HR and finance teams to focus on strategic tasks. Employees benefit from a modernized and efficient system that enhances job satisfaction and productivity.

### ACKNOWLEDGEMENT

First and foremost, we thank to **power of almighty** for showing us inner peace and for all blessings. Special gratitude to our parents, for showing their support and love always.

We like to acknowledge the constant support provided by **Sri.C.Valliappa,** Chairman, for his consistent motivation in pursuing my project.

We express our sincere thanks to the Head of Information Technology, **Dr.J.Akilandeswari,M.E,Ph.D,** for providing adequate laboratory facilities to complete this thesis.

We feel elated to keep on record our heartfelt thanks and gratitude to our project guide **Dr.J.Akilandeswari,M.E,Ph.D** our steadfast inspiration, for her valuable guidance, untiring patience and diligent encouragement during the entire span of this project.

We feel proud in sharing this success with my staff members, non- teaching staffs and friends who helped directly or indirectly in completing this project successfully.

**TABLE OF CONTENTS**

**CHAPTER NO. TITLE PAGE NO.**

1. INTRODUCTION 7
   1. [About the project 7](#_bookmark0)
   2. [Problem description 8](#_bookmark1)
   3. [Scope for Future Development 8](#_bookmark2)
2. **LITERATURE SURVEY 9**
3. HARDWARE AND SOFTWARE REQUIREMENTS 11
4. PROJECT DESCRIPTION 12
   1. [Scope of the project 1](#_bookmark3)2
   2. [REACT JS 1](#_bookmark4)3
   3. [NODE JS AND SQL 1](#_bookmark5)3
   4. [System Architecture and Design 1](#_bookmark6)5
5. RESULTS 16
6. SOURCE CODE 21
7. CONCLUSIONS AND

FUTURE WORK 40

* 1. [Conclusion](#_bookmark7) **40**
  2. Future Work **40**

## LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **TITLE** | **PG NO.** |
| **4.4.1** | **ARCHITECTURAL DESIGN** | **15** |
| **5.1.1** | **LOGIN PAGE** | **16** |
| **5.1.2** | **TRAVEL HISTORY** | **16** |
| **5.1.3** | **TRAVEL REQUEST** | **17** |
| **5.1.4** | **TRAVEL DETAILS** | **17** |
| **5.1.5** | **TRAVEL MODE** | **18** |
| **5.1.6** | **ADVANCE** | **18** |
| **5.1.7** | **PREVIEW** | **19** |
| **5.2.1** | **APPROVAL PAGE** | **20** |

### LIST OF ABBREVIATIONS

**1.1 SQL - Structured Query Language**

# CHAPTER 1

# INTRODUCTION

## ABOUT THE PROJECT

The Reimbursement App introduces a dynamic and user-friendly platform, incorporating a robust set of features to address the intricacies of organizational reimbursement processes. Administered by a Product Admin, the application allows for the creation of multiple companies, each governed by a designated Company Admin. This hierarchical structure enables efficient management and customization of reimbursement workflows tailored to the specific needs of individual companies.

Under the guidance of the Company Admin, employees and company departments can be seamlessly added or modified. The flexibility of the application extends to the assignment of designations within the company structure. This comprehensive administrative control ensures that the Reimbursement App is adaptable to the evolving dynamics of the organization, providing a centralized hub for streamlined management. Employees, empowered by the intuitive user interface, can submit travel reimbursement requests both before and after any travel. This feature is designed to enhance user convenience and accommodate varying reimbursement scenarios. The requests undergo a systematic approval process, wherein assigned approvers evaluate and authorize reimbursements. This approval mechanism adds a layer of accountability and ensures that reimbursements align with company policies. The Reimbursement App leverages its robust architecture, incorporating React.js, Node.js, and SQL, to guarantee data integrity and security. The SQL database facilitates comprehensive reporting, offering insights into reimbursement trends and ensuring compliance with established company guidelines. By automating and digitizing the reimbursement workflow, the application significantly reduces the administrative burden on companies, allowing for a more efficient and prompt reimbursement process.

In essence, the Reimbursement App emerges as a sophisticated solution that not only optimizes reimbursement procedures but also empowers administrators to effectively manage company structures, employee details, and approval workflows. This comprehensive approach positions the application as a valuable asset in modernizing and simplifying the complex landscape of employee reimbursement within organizations.

## PROBLEM DESCRIPTION

Conventional reimbursement systems present persistent challenges with their intricate and time-consuming processes, creating a significant operational burden for organizations. The Reimbursement App is a response to these issues, offering an innovative platform that automates and digitizes the reimbursement workflow. This transformative solution eliminates manual errors, alleviating the administrative load on companies and allowing HR and finance teams to allocate their time more strategically. The application not only streamlines reimbursement processes but also tackles tracking inefficiencies and delays by providing real-time monitoring. This ensures continuous communication between employees and administrators, enhancing transparency throughout the reimbursement journey. By modernizing these procedures, the Reimbursement App strives to improve overall efficiency, accuracy, and transparency in reimbursement systems, offering a user-friendly and contemporary alternative that effectively addresses the limitations of traditional methods.

## SCOPE FOR FUTURE DEVELOPMENT

The Reimbursement App lays the foundation for future development and enhancement. One avenue for expansion is the integration of additional features, such as machine learning algorithms for expense categorization or mobile applications for on-the-go reimbursement submissions. Collaboration with financial institutions for seamless fund transfers and integration with other HR systems for a holistic organizational approach are potential future directions. Continuous updates to align with evolving compliance standards and technological advancements will ensure the Reimbursement App remains at the forefront of efficiency in reimbursement processes. The adaptable architecture allows for scalability, enabling the incorporation of emerging technologies to meet the evolving needs of organizations.

**CHAPTER 2**

**LITERATURE SURVEY**

[1] **D. E. Yurochkin, A. A. Horoshiy and S. A. Karpukhin, "Development of an Application for Expense Accounting," 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus), St. Petersburg, Moscow, Russia, 2021, pp. 753-757, doi: 10.1109/ElConRus51938.2021.9396201.**

The article is devoted to the description of creating a mobile application that helps to keep tracking your expenses and control your budget. Its works as follows: a receipt is scanned by a phone camera, preprocessing image. Then the text on the image is recognized, the needed data is selected and saved to database. The application records statistics on expenses received from checks. In this study, the main tasks were implementing a correctly working text recognition algorithm, determining the required data and their distribution. Despite tasks complexity, they have been successfully resolved. Further, the preservation and further storage of this data was implemented. After that, it was necessary to create a mobile application, with the help of which the cost accounting will be carried out and in which the possibility of splitting purchases will be realized, but the development process is still ongoing.

[2]  **P. Bhatele, D. Mahajan, B. Mahajan, D. Mahajan, N. Mahajan and P. Mahajan, "TrackEZ Expense Tracker," 2023 4th International Conference for Emerging Technology (INCET), Belgaum,India,2023,pp.1-5, doi: 10.1109/INCET57972.2023.10170735.**

Expense tracker is an expense management system designed for day-to-day life. The application capably tracks the daily expenses of the user. Such applications allow the users to easily manage their expenditure and hence, eliminates the need of manual paper tasks. Such trackers are computerized diaries used to keep a record of the transactions made by the user. This paper explains about an expense tracker web application that inputs the salary from the user, source of this income and the date of earning that salary and creates a transaction entry as an income. It sums the entries to the total amount of income and makes real time changes. Similarly, it will also input the expenses and make entries for the same. The entries can be deleted after creation. The distribution of income and the expenditure can be visualized in the form of charts and graphs that will keep updating as per user’s transaction.

[3] **S. A. Sabab, S. S. Islam, M. J. Rana and M. Hossain, "eExpense: A Smart Approach to Track Everyday Expense," 2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT), Dhaka, Bangladesh, 2018, pp. 136-141, doi: 10.1109/CEEICT.2018.8628070.**

Tracking regular expense is a key factor to maintain a budget. People often track expense using pen and paper method or take notes in a mobile phone or a computer. These processes of storing expense require further computations and processing for these data to be used as a trackable record. In this work, we are proposing an automated system named as eExpense to store and calculate these data. eExpnese is an application that runs on Android smartphones. By using this application, users can save their expense by simply scanning the bills or receipt copies. This application extracts the textual information from the receipts and saves the amount and description for further processing. It also monitors user's income by tracking the received SMS's from the user's saving accounts. By calculating income and expense it produces the user's balance in monthly and yearly basis. Overall, this is a smart automated solution for tracking expense.

## CHAPTER 3

## HARDWARE AND SOFTWARE REQUIREMENTS

This chapter provides brief description about the requirements essential for our project.

## HARDWARE REQUIREMENT

|  |  |
| --- | --- |
| CPU | Any CPU which can run latest  version of browsers. |
| RAM | 2GB minimum, 4GB or more recommended |
| Operating system | Any operating system which can run browsers. |
| Internet connection | 5mbps minimum |

**SOFTWARE REQUIREMENT**

Latest version of any browser. Recommended: (Chrome 119.,

Safari 17.1., Edge 119., Firefox

119.)

Browser

## CHAPTER 4

## PROJECT DESCRIPTION



## SCOPE OF THE PROJECT:

The Reimbursement App boasts a comprehensive scope, aspiring to reshape the reimbursement landscape within organizational frameworks. At its core, the project is dedicated to the automation and digitization of the reimbursement workflow, introducing a platform that is both user-friendly and highly efficient. The application's versatility encompasses various reimbursement categories, prominently featuring travel and broadband expenses. This inclusivity ensures that the Reimbursement App caters to the diverse needs of organizations, providing a flexible solution that adapts to specific company policies. A key highlight of the project's scope lies in its customization capabilities, enabling organizations to tailor the app to their unique requirements. This adaptability ensures that the Reimbursement App aligns seamlessly with the intricacies of individual company policies, offering a level of flexibility that is crucial in today's dynamic business environment. The strategic integration of SQL not only fortifies data security but also introduces a robust foundation for advanced reporting and auditing functionalities. This incorporation elevates the Reimbursement App beyond mere efficiency gains, facilitating comprehensive insights into reimbursement trends and ensuring adherence to established compliance standards.

Beyond operational Work, the project aspires to make a meaningful impact on employee satisfaction. By providing a modernized and streamlined reimbursement experience, the Reimbursement App becomes a catalyst for positive employee engagement. Its user-centric design and intuitive functionalities aim to enhance the overall satisfaction of employees, contributing to a more positive and productive workplace environment. In essence, the Reimbursement App's scope transcends the boundaries of traditional reimbursement systems. It emerges as a versatile and valuable asset for organizations seeking not only to optimize processes but also to strategically allocate resources, ultimately fostering a more efficient, satisfied, and forward-thinking workplace.

## REACT JS

React.js serves as the frontend framework for the Reimbursement App, contributing to its cutting-edge user interface and seamless user experience. Developed and maintained by Facebook, React.js is a powerful JavaScript library that enables the creation of interactive and dynamic user interfaces. With its component-based architecture, React allows developers to build modular UI elements, facilitating code reusability and maintainability. The Reimbursement App leverages React's virtual DOM (Document Object Model) for efficient updates and rendering, ensuring a responsive and performant interface. Through the use of React.js, users can enjoy a smooth and intuitive reimbursement process, enhancing overall user satisfaction and engagement.

## NODE JS AND SQL

On the backend, the Reimbursement App relies on Node.js to power its server-side logic and handle data communication between the frontend and the database. Node.js, built on the V8 JavaScript runtime, is renowned for its non-blocking, event-driven architecture, making it well- suited for scalable and real-time applications. With Node.js, the Reimbursement App can efficiently handle concurrent user requests, providing a responsive and agile system. Additionally, Node.js facilitates seamless integration with other technologies, ensuring smooth data flow between the client and server components. This backend technology enhances the overall performance and reliability of the Reimbursement App, contributing to its ability to streamline reimbursement workflows.

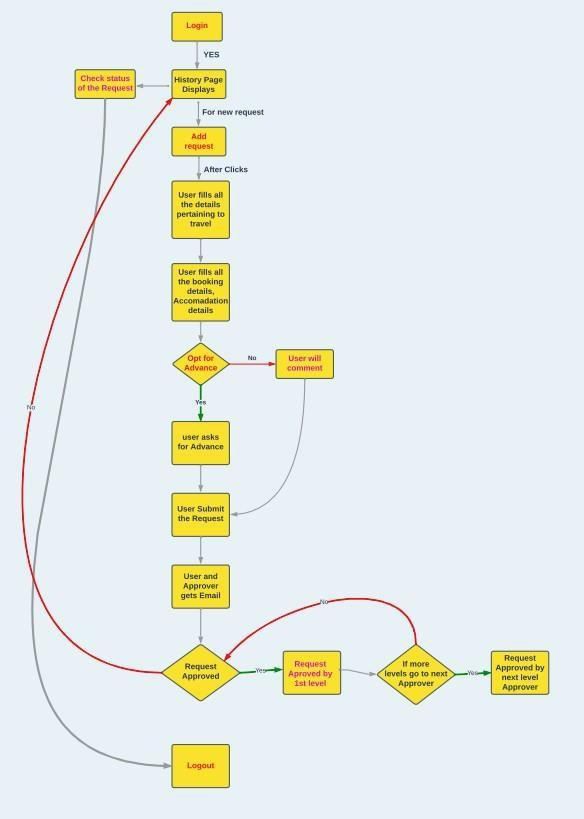
The Reimbursement App relies on SQL (Structured Query Language) for its database management, leveraging incremental ID-based primary keys for enhanced performance, data integrity, and security. This strategic choice in database design ensures that the application operates with optimal efficiency, minimizing process time during data retrieval and manipulation.

Utilizing SQL as the standard language for managing relational databases, the Reimbursement App establishes a robust foundation for storing and retrieving reimbursement-related information. The incorporation of incremental ID-based primary keys adds a layer of efficiency to data management. This approach streamlines the process of accessing specific records, reducing search times and enhancing the overall responsiveness of the application.

The Reimbursement App's use of SQL extends beyond basic data storage; it empowers the application to execute complex queries. This capability is crucial for comprehensive reporting and auditing functionalities, allowing organizations to gain valuable insights into reimbursement trends and ensuring adherence to compliance standards. The incremental ID-based primary keys contribute to the expeditious execution of queries, further optimizing the reporting process. Furthermore, the adoption of incremental ID-based primary keys enhances the application's ability to organize and manage large datasets. This is particularly essential for tracking and analyzing reimbursement transactions efficiently. By implementing this approach, the Reimbursement App not only ensures the security of sensitive financial data but also lays the groundwork for a more responsive and streamlined reimbursement process.

In summary, the Reimbursement App's strategic use of SQL, with incremental ID-based primary keys, reflects a commitment to efficiency, data integrity, and security in managing reimbursement-related information. This approach not only minimizes process time but also contributes to the overall enhancement of reimbursement processes within organizations.

## SYSTEM ARCHITECTURE AND DESIGN

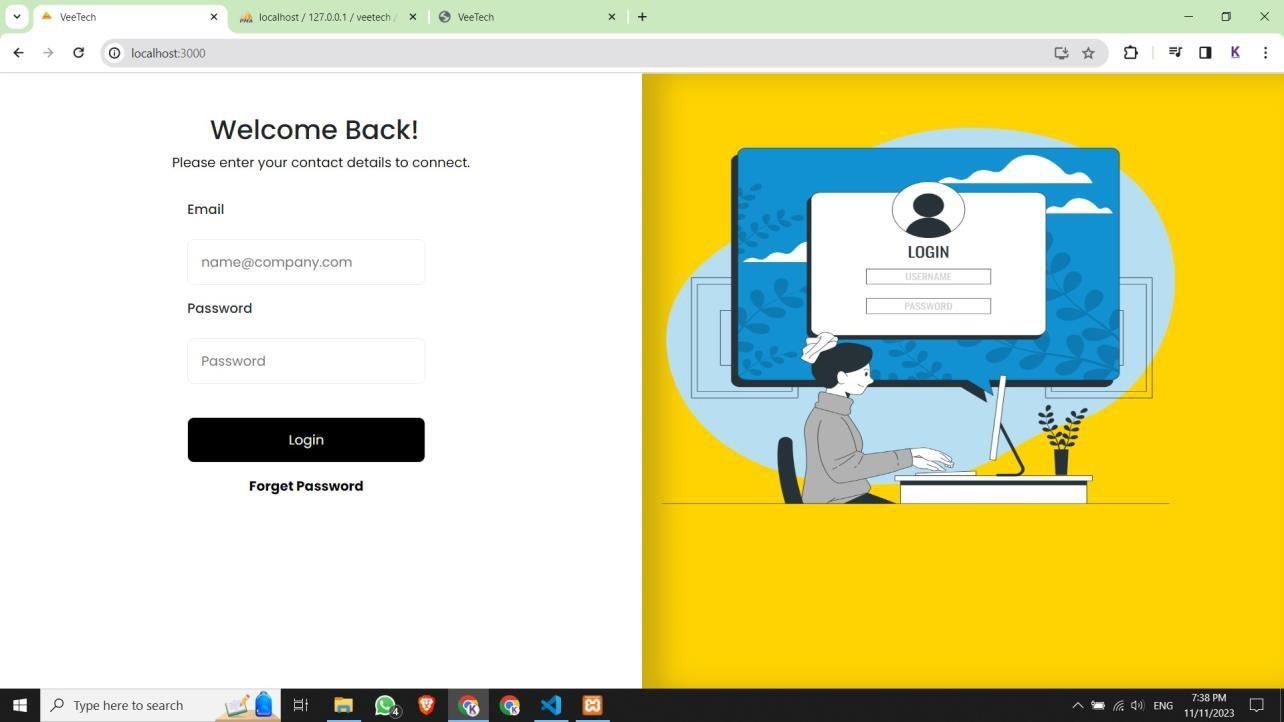


**Fig 4.4.1. ARCHITECTURAL DESIGN**

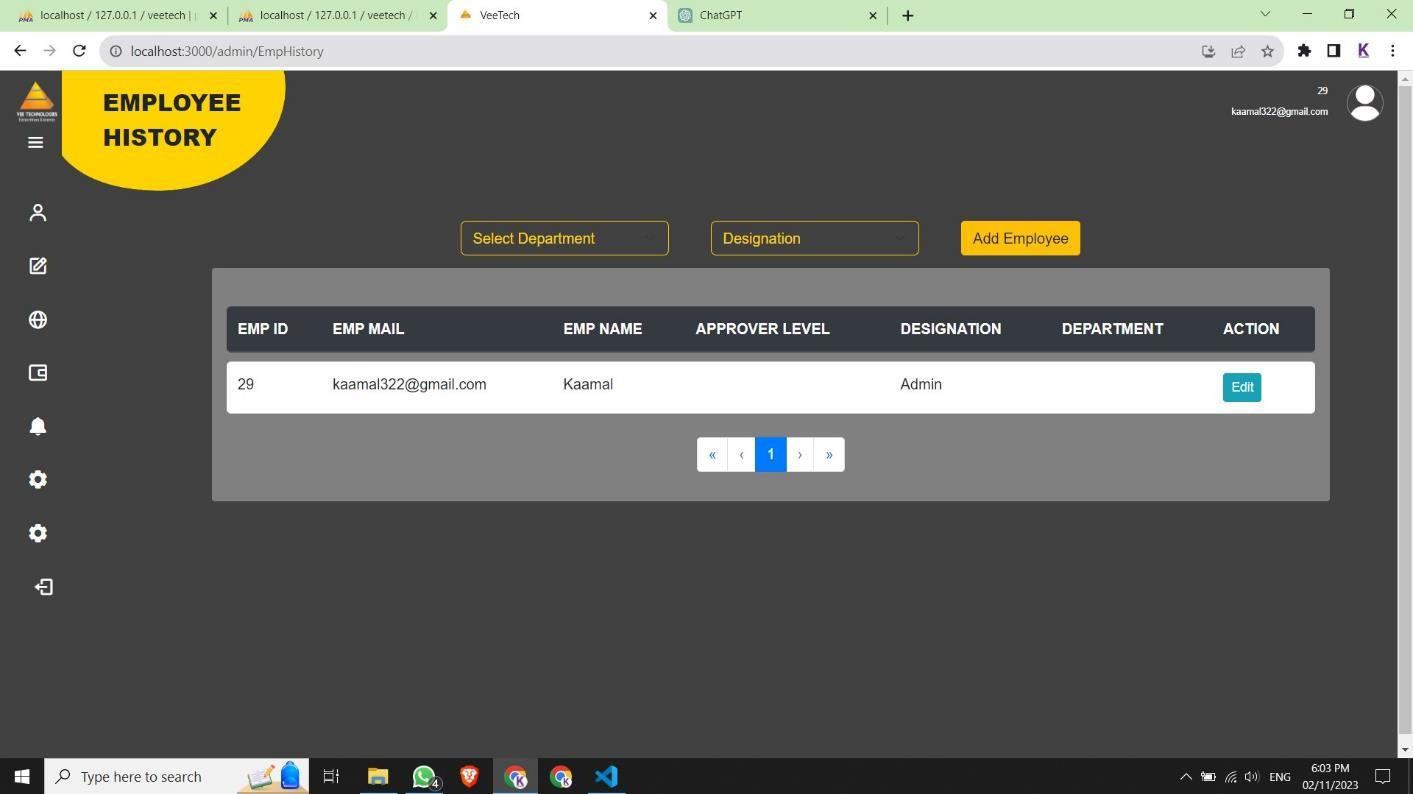
## CHAPTER 5

## RESULTS

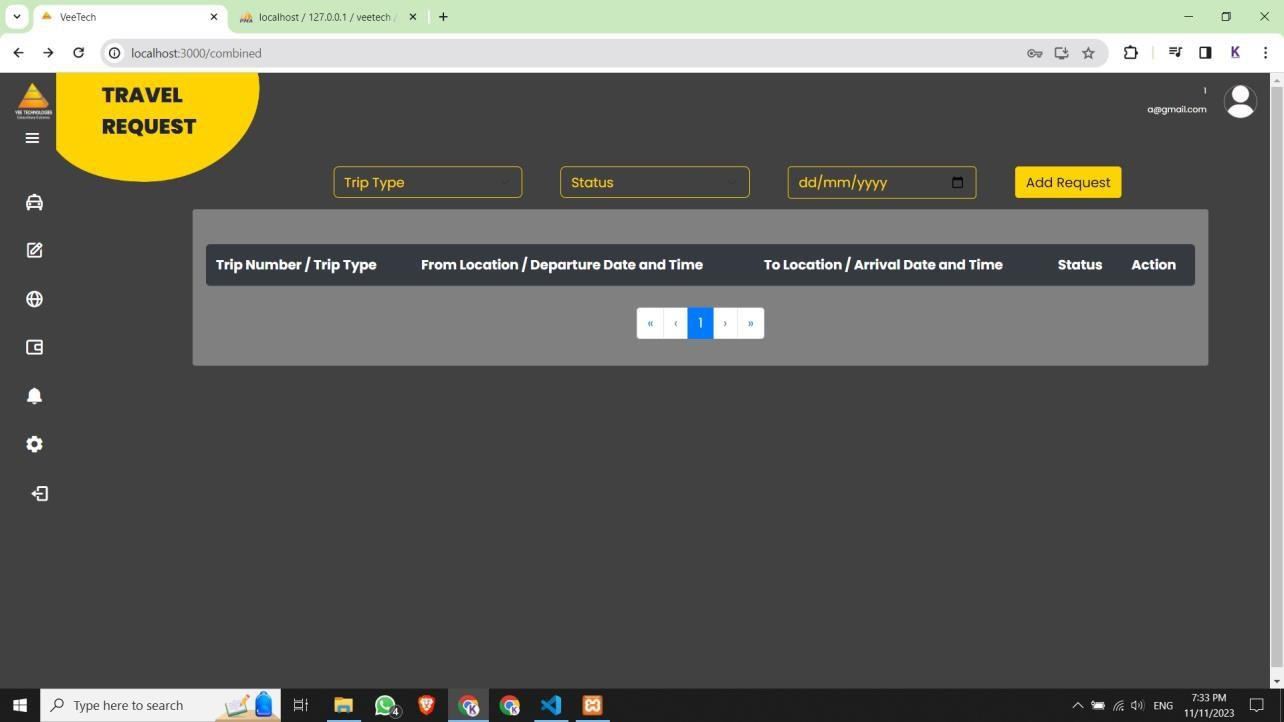
* 1. **User Module :**



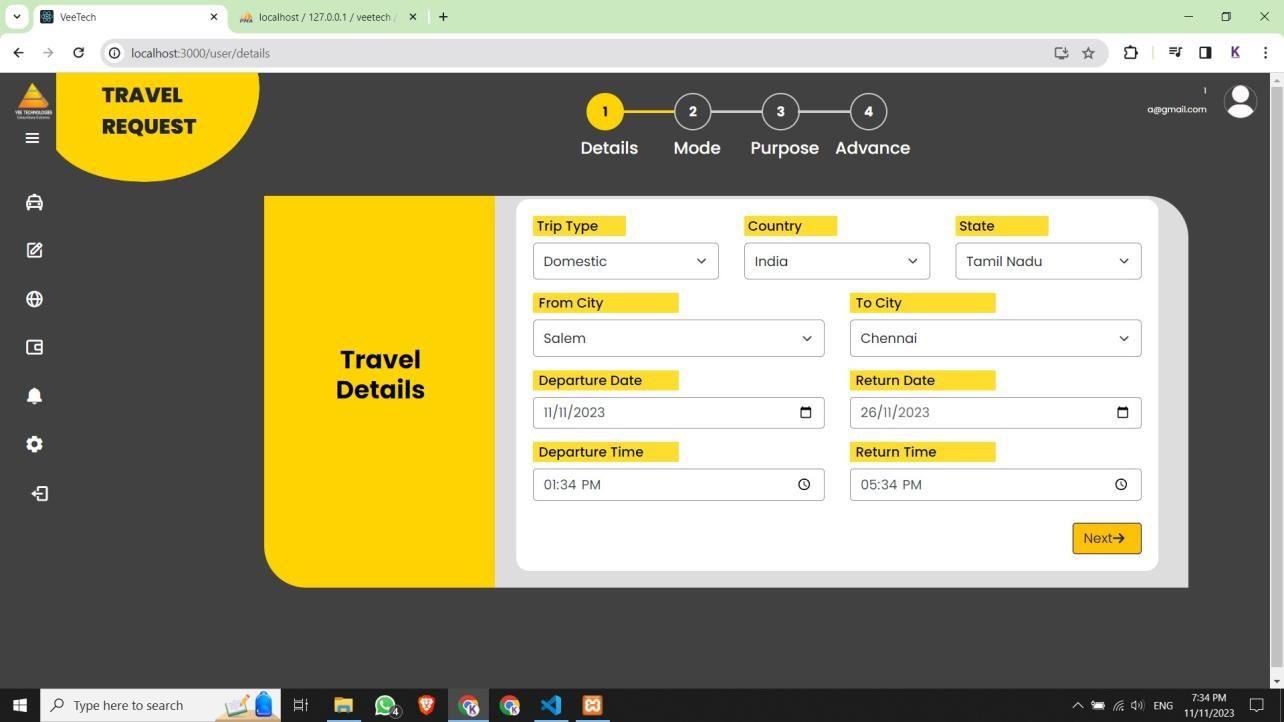
## Fig 4.1.1.Login Page



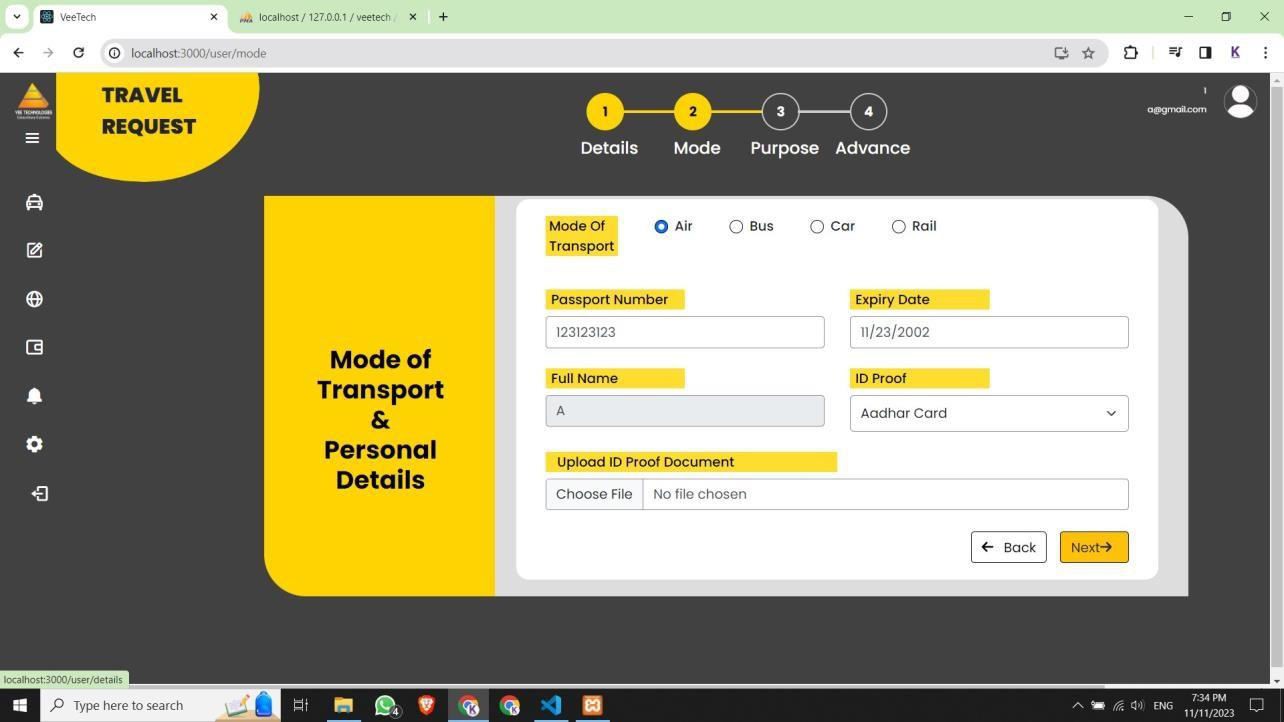
**Fig 5.1.2. Travel History**



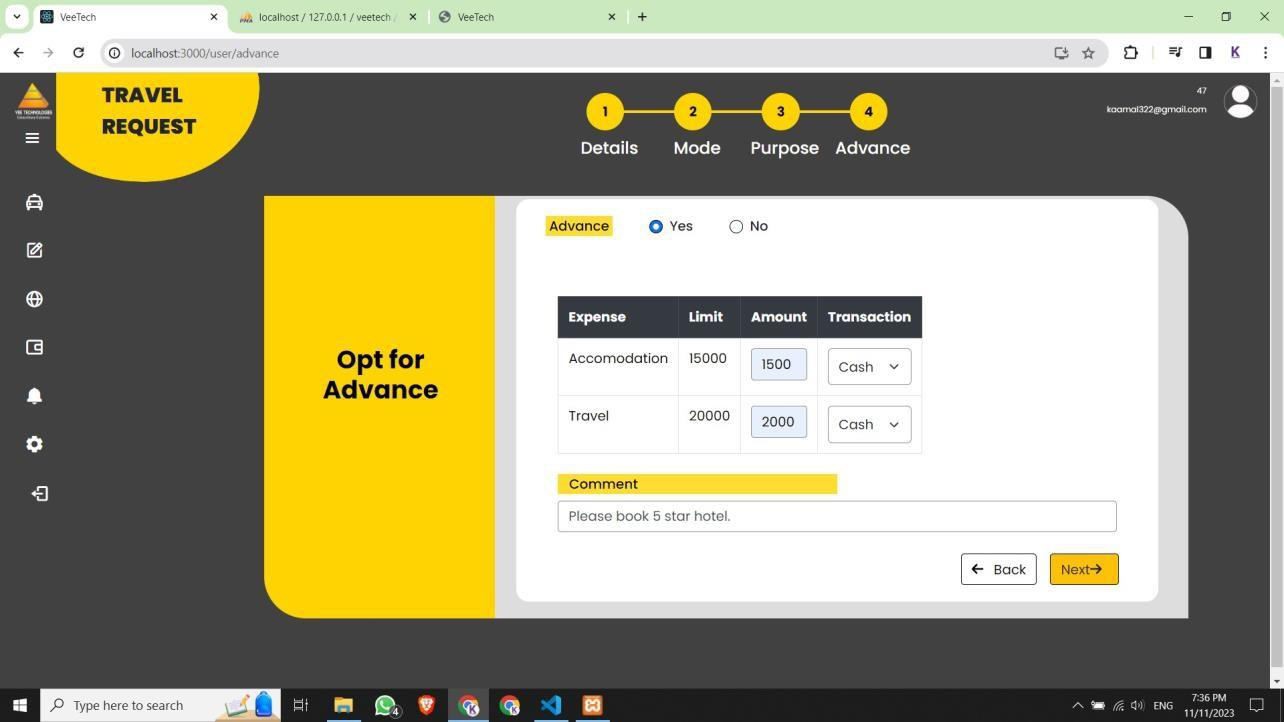
## Fig 5.1.3. Travel Request



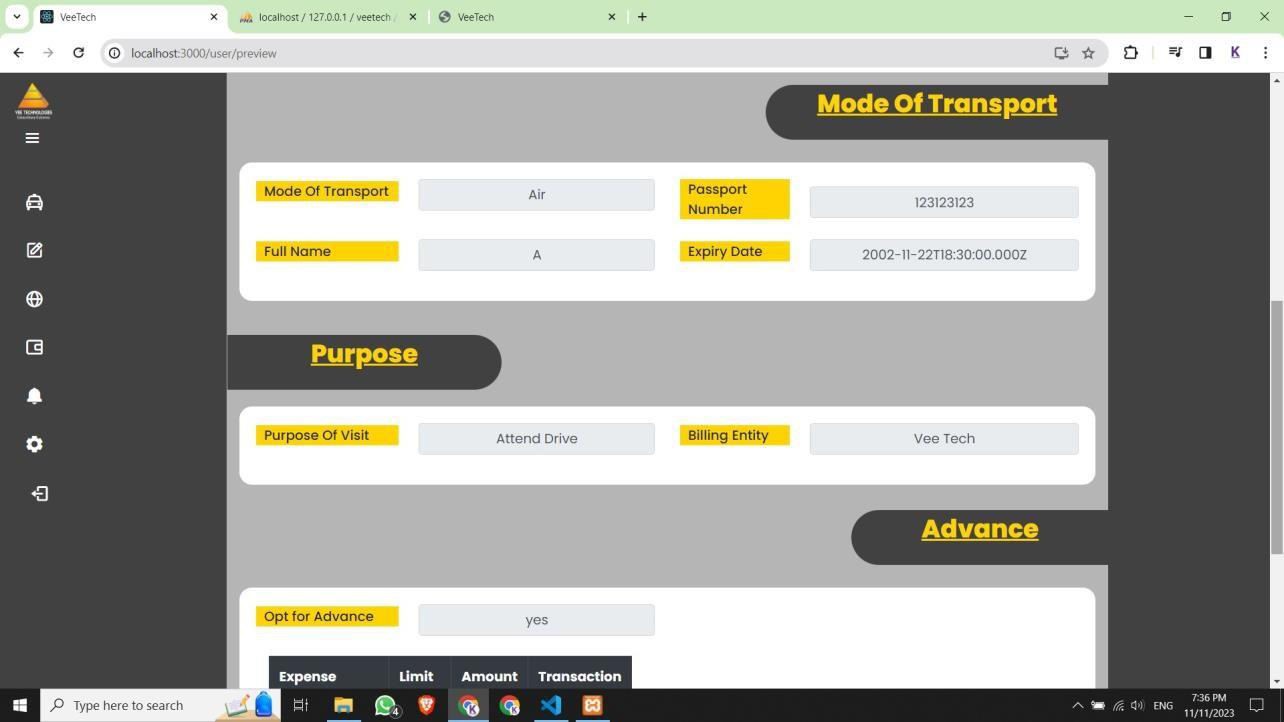
**Fig 5.1.4. Travel Details**

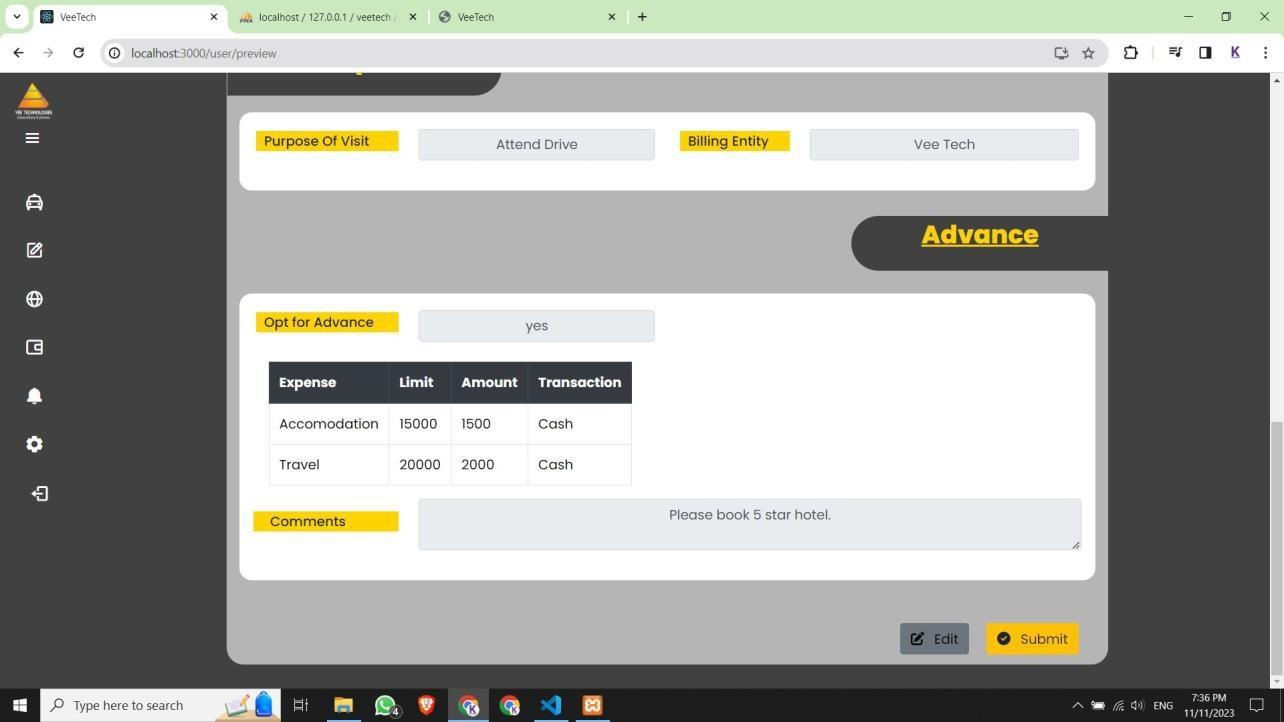


## Fig 5.1.5. Travel Mode



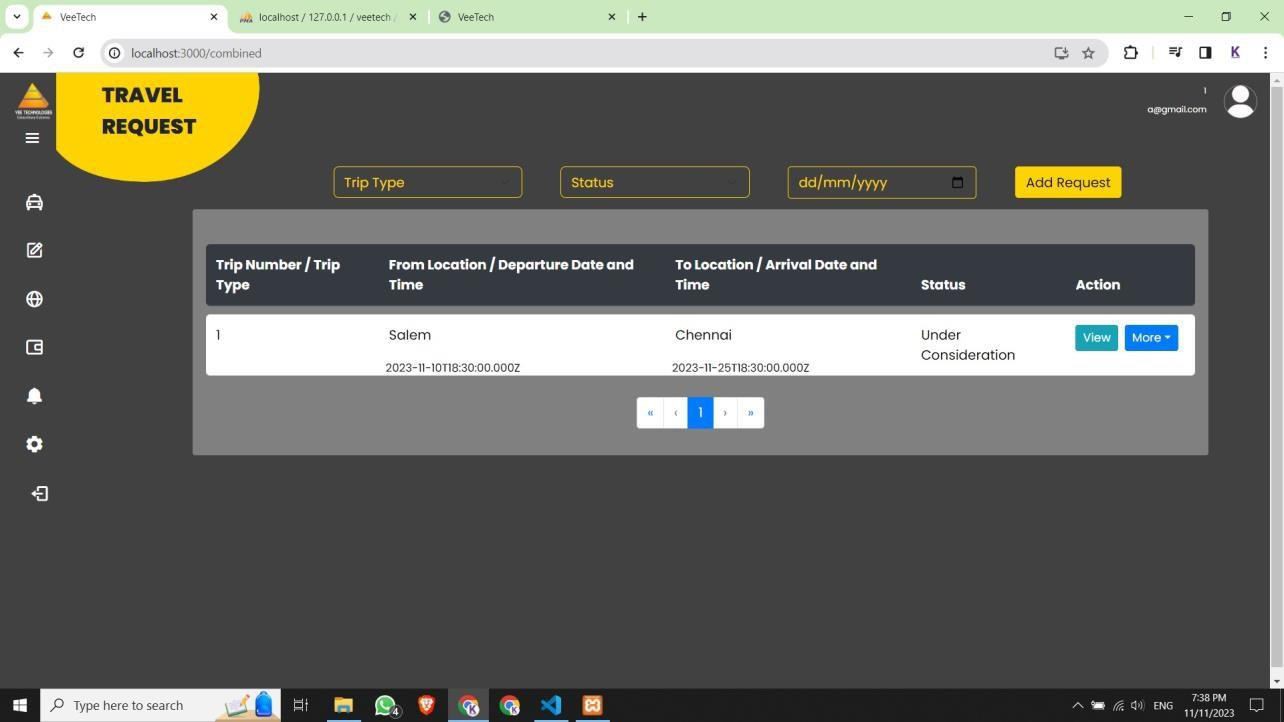
**Fig 5.1.6.Advance**





## Fig 5.1.7. Preview

* 1. **Admin Module :**



## FIG 5.2.1. Approval Page

**SAMPLE CODE**

## Travel History:

import React, { useState, useEffect } from 'react'; import axios from 'axios';

import { Pagination } from 'react-bootstrap';

import { Link, useNavigate } from 'react-router-dom'; import 'bootstrap/dist/css/bootstrap.min.css';

function TravelHistory() {

const [currentPage, setCurrentPage] = useState(1);

const [selectedTripType, setSelectedTripType] = useState(''); const [selectedStatus, setSelectedStatus] = useState(''); const [selectedDate, setSelectedDate] = useState('');

const [data, setData] = useState([]);

const [teamdata, setteamData] = useState([]);

const [expandedRows, setExpandedRows] = useState({});

const [showFirstDiv, setShowFirstDiv] = useState(true); // Added state to toggle first and second div

const navigate = useNavigate();

const approverLevelId = localStorage.getItem('approverLevelId');

useEffect(() => { axios

.post('http://localhost:3002/travel\_details', { uid: localStorage.getItem('userid'),

})

.then((response) => {

setData(response.data); console.log(response.data);

})

.catch((error) => {

console.error('Error fetching travel data:', error);

});

}, []); // Empty dependency array to run once on component mount

useEffect(() => {

// Fetch travel request data based on the current user's data and approver level axios.post('http://localhost:3002/team\_travel\_request\_details', { uid

:localStorage.getItem('userid'), approverLevelId,})

.then((response) => { setteamData(response.data);

})

.catch((error) => {

console.error('Error fetching travel data:', error);

});

}, []); // Fetch data when userId or approverLevelId changes

const handlePageChange = (pageNumber) => { setCurrentPage(pageNumber);

};

const handleTripTypeChange = (value) => { setSelectedTripType(value);

};

const handleStatusChange = (value) => {

setSelectedStatus(value); console.log(value);

};

const handleDateChange = (value) => { setSelectedDate(value);

};

const handleViewClick = (tripId,fromFirstDiv) => {

// Store the tripId in local storage

localStorage.setItem('viewSource', fromFirstDiv ? 'firstDiv' : 'secondDiv'); localStorage.setItem('selectedTripId', tripId);

//console.log(tripId);

// Navigate to the /user/view page navigate(`/user/view`);

};

const [statusOptions, setStatusOptions] = useState([]); useEffect(() => {

axios

.post('http://localhost:3002/status\_options', { uid :localStorage.getItem('userid'), }) // Send 'uid' in the request body

.then((response) => {

// Assuming the API returns an array of status objects with 'id' and 'status' properties const statusOptions = response.data.map((status) => ({

value: status.status, label: status.status,

}));

setStatusOptions(statusOptions);

})

.catch((error) => {

console.error('Error fetching status options:', error);

});

}, []);

const [mappingRange, setMappingRange] = useState({ firstApprover: null, lastApprover: null

});

useEffect(() => {

// Make a GET request to fetch the first and last values of the mapping axios

.post('http://localhost:3002/get\_approver\_mapping\_range',{ uid

:localStorage.getItem('userid'), }) // Replace with your actual API endpoint

.then((response) => { setMappingRange(response.data); const data = response.data; console.log(data);

// Extract the first column values

const firstColumnValues = Object.keys(data).map(key => parseInt(key.split(',')[0]));

// Calculate the minimum and maximum values

const minApprover = Math.min(...firstColumnValues); const maxApprover = Math.max(...firstColumnValues);

// Set the state with the values

setMappingRange({ firstApprover: minApprover, lastApprover: maxApprover });

// Store the values in localStorage localStorage.setItem('firstApprover', minApprover.toString()); localStorage.setItem('lastApprover', maxApprover.toString());

})

.catch((error) => {

console.error('Error fetching approver mapping range:', error);

});

}, []);

const toggleRowExpansion = (tripId) => { setExpandedRows((prevExpandedRows) => ({

...prevExpandedRows,

[tripId]: !prevExpandedRows[tripId],

}));

};

const firstApprover = localStorage.getItem('firstApprover'); const itemsPerPage = 6;

const startIndex = (currentPage - 1) \* itemsPerPage; const endIndex = startIndex + itemsPerPage;

const filteredData = data

.filter((item) => selectedTripType === '' || item.trip\_type === selectedTripType)

.filter((item) => selectedStatus === '' || item.status === selectedStatus)

.filter((item) => {

const departureDate = item.departureDateTime ? item.departureDateTime.split('T')[0] : ''; return departureDate >= selectedDate;

});

const currentData = filteredData && filteredData.length >0 ? filteredData.slice(startIndex, endIndex) : [];

const filteredteamData = teamdata

.filter((item) => selectedTripType === '' || item.trip\_type === selectedTripType)

.filter((item) => selectedStatus === '' || item.status === selectedStatus)

.filter((item) => {

const departureDate = item.departureDateTime ? item.departureDateTime.split('T')[0] : ''; return departureDate >= selectedDate;

});

const currentTeamData = filteredteamData.slice(startIndex, endIndex); console.log(currentData);

return (

<div id="global-background">

<div id="travelContainer">

<div id="travelRow">

<div id="travelrequestFilter" className="col-md-3">

<select

className="form-select mb-2" aria-label="Trip Type" value={selectedTripType}

onChange={(e) => handleTripTypeChange(e.target.value)}

>

<option value="">Trip Type</option>

<option value="Domestic">Domestic</option>

<option value="International">International</option>

</select>

</div>

<div id="travelrequestFilter" className="col-md-3">

<select

className="form-select mb-2" aria-label="Status" value={selectedStatus}

onChange={(e) => handleStatusChange(e.target.value)}

>

<option value="">Status</option>

{statusOptions.map((option) => (

<option key={option.value} value={option.value}>

{option.label}

</option>

))}

</select>

</div>

<div id="travelrequestFilter" className="col-md-3">

<input type="date"

className="form-control mb-2" value={selectedDate}

onChange={(e) => handleDateChange(e.target.value)}

/>

</div>

<div id="travelrequestFilter" className="col-md-3">

<Link to="/user/details">

<button className="btn btn-primary mb-2">Add Request</button>

</Link>

</div>

</div>

{approverLevelId >= firstApprover && (

<div className='d-flex' id="travelrequestToggle" >

<button

className="btn mb-2 me-2" onClick={() => setShowFirstDiv(true)} id="travelrequestToggleButtons"

>

My Requests

</button>

<button

className="btn mb-2 me-2" onClick={() => setShowFirstDiv(false)} id="travelrequestToggleButtons"

>

My Team Requests

</button>

<Link to="/user/travelapprovalrejection">

<button className="btn mb-2"

id="travelrequestToggleButtons"

>

Approval/Rejection History

</button>

</Link>

</div>

)}

{showFirstDiv && (

<div id="travelrequestBackground" style={{ marginTop: approverLevelId >= firstApprover ? '0' : '12%' }}>

<table id="travelrequestTable" className="table table\_border">

<thead className="thead-dark">

<tr>

<th id='tableStart'>Trip Number / Trip Type</th>

<th>From Location / Departure Date and Time</th>

<th>To Location / Arrival Date and Time</th>

<th>Status</th>

<th id='tableEnd'>Action</th>

</tr>

</thead>

<tbody>

{currentData.map((item) => (

<React.Fragment key={item.trip\_id}>

<tr key={item.trip\_id} >

<td id="travelrequestTdFirstChild">

{item.trip\_id}

<sub id="travelrequestSub">{item.trip\_type}</sub>

</td>

<td id="travelrequestTd">

{item.from\_city\_name}

<sub id="travelrequestSub">{item.departure\_date}</sub>

</td>

<td id="travelrequestTd">

{item.to\_city\_name}

<sub id="travelrequestSub">{item.return\_date}</sub>

</td>

<td id="travelrequestTd">{item.status}</td>

<td id="travelrequestTdLastChild">

<div className="d-flex align-items-center action\_buttons">

<Link to='/user/view'>

<button id="travelrequestView" className="btn btn-sm btn-info me-2" onClick={()

=> handleViewClick(item.trip\_id,true)}>View</button>

</Link>

<button className="btn btn-sm btn-primary dropdown-toggle me-2" onClick={() => toggleRowExpansion(item.trip\_id)}>More</button>

</div>

</td>

</tr>

{expandedRows[item.trip\_id] && (

<tr id="more\_details\_row">

<td colSpan="5" className="grid-container">

<div className="grid-item">

<strong>Review: </strong>

{item.rejector\_remark ? "Travel request rejected" : item.rejector\_remark || "Review pending from the approver."}

</div>

<div className="grid-item">

<strong>Accommodation Details: </strong>

{item.rejector\_remark ? "Travel request rejected" : (item.accomodation\_address ? item.accomodation\_address : "Accommodation details are yet to be allotted")}

</div>

<div className="grid-item">

<strong>Ticket Details: </strong>

{item.rejector\_remark ? "Travel request rejected" : (item.ticket\_details ? item.ticket\_details : "Ticket details are yet to be allotted")}

</div>

</td>

</tr>

)}

</React.Fragment>

))}

</tbody>

</table>

<Pagination className="justify-content-center ">

<Pagination.First onClick={() => handlePageChange(1)} />

<Pagination.Prev

onClick={() => handlePageChange(currentPage - 1)} disabled={currentPage === 1}

/>

<Pagination.Item active>{currentPage}</Pagination.Item>

<Pagination.Next

onClick={() => handlePageChange(currentPage + 1)} disabled={currentData.length < itemsPerPage}

/>

<Pagination.Last onClick={() => handlePageChange(Math.ceil(data.length / itemsPerPage))} />

</Pagination>

</div>

)}

{!showFirstDiv && (

<div id="travelrequestBackground" style={{ marginTop: approverLevelId >= firstApprover ? '0' : '12%' }}>

<table id="travelrequestTable" className="table table\_border">

<thead className="thead-dark">

<tr>

<th id='tableStart'>Trip Number / Trip Type</th>

<th>From Location / Departure Date and Time</th>

<th>To Location / Arrival Date and Time</th>

<th>Status</th>

<th id='tableEnd'>Action</th>

</tr>

</thead>

<tbody>

{currentTeamData.map((teamItem) => (

<React.Fragment key={teamItem.trip\_id}>

<tr key={teamItem.trip\_id}>

<td id="travelrequestTdFirstChild">

{teamItem.trip\_id}

<sub id="travelrequestSub">{teamItem.trip\_type}</sub>

</td>

<td id="travelrequestTd">

{teamItem.from\_city\_name}

<sub id="travelrequestSub">{teamItem.departure\_date}</sub>

</td>

<td id="travelrequestTd">

{teamItem.to\_city\_name}

<sub id="travelrequestSub">{teamItem.return\_date}</sub>

</td>

<td id="travelrequestTd">{teamItem.status}</td>

<td id="travelrequestTdLastChild">

<div className="d-flex align-items-center action\_buttons">

<Link to="/user/view">

<button id="travelrequestView"

className="btn btn-sm btn-info me-2"

onClick={() => handleViewClick(teamItem.trip\_id,false)}

>

View

</button>

</Link>

</div>

</td>

</tr>

</React.Fragment>

))}

</tbody>

</table>

<Pagination className="justify-content-center ">

<Pagination.First onClick={() => handlePageChange(1)} />

<Pagination.Prev

onClick={() => handlePageChange(currentPage - 1)} disabled={currentPage === 1}

/>

<Pagination.Item active>{currentPage}</Pagination.Item>

<Pagination.Next

onClick={() => handlePageChange(currentPage + 1)} disabled={currentTeamData.length < itemsPerPage}

/>

<Pagination.Last

onClick={() => handlePageChange(Math.ceil(teamdata.length / itemsPerPage))}

/>

</Pagination> </div>)}<div/></div>);} export default TravelHistory;

## index.js (Backend) :

const express = require('express');

const bodyParser = require('body-parser'); const cors = require('cors');

const nodemailer = require('nodemailer'); const mysql = require('mysql');

const app = express(); app.use(cors());

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

const connection = mysql.createConnection({ host: 'localhost',

user: 'root',

password: '', database: 'VeeTech'

});

var transporter = nodemailer.createTransport({ service: 'Outlook365',

auth: {

user: 'abhindra.20it@sonatech.ac.in', pass: 'abhi123abhI'

},

tls: {

rejectUnauthorized: false

}

});

app.post('/login', (req, res) => { const email = req.body.email;

const password = req.body.password;

const query = `SELECT \* FROM login\_table WHERE email = ? AND password = ?`; connection.query(query, [email, password], (error, results) => {

if (error) {

console.error('Error executing query:', error); res.send('An error occurred');

}

else {

console.log('Query results:', results); console.log(results[0].email);

if (results[0].email == "[admin@gmail.com](mailto:admin@gmail.com)") { console.log("Logged in as Product Admin"); res.send('admin');

} else if (results.length > 0) {

const qu = `SELECT designation\_id, company\_id FROM employee\_table WHERE

mail = ?`;

connection.query(qu, [email], (error1, results1) => { if (error1) {

console.error('Error executing query:', error1); res.send('An error occurred');

} else {

const designationId = results1[0].designation\_id;

// Add a SQL query to fetch the designation\_name from the designation\_table const designationQuery = 'SELECT designation\_name FROM designation\_table

WHERE id = ?';

connection.query(designationQuery, [designationId], (error2, results2) => { if (error2) {

console.error('Error executing designation query:', error2); res.send('An error occurred');

} else {

const designationName = results2[0].designation\_name.toLowerCase();

if (designationName == "admin") { console.log("Login Success as Company Admin"); const companyId = results1[0].company\_id; res.send("admin1/" + companyId);

else {

} else {

res.send(`Login successful`); console.log("Login Successfull");}}});}});}

res.send('Invalid email or password'); console.log("Login Not Successful");

}

}

});

});

app.post("/travel\_details", (req, res) => { const uid = req.body.uid;

const getEmployeeIdQuery = "SELECT id FROM employee\_table WHERE employee\_id = ?"; connection.query(getEmployeeIdQuery, [uid], (err, employeeResults) => {

if (err) {

console.error('Error querying the database for employee ID: ' + err.stack); res.status(500).json({ error: 'Database error' });

return;

}

if (employeeResults.length === 0) { console.error('Employee not found in the database.'); res.status(404).json({ error: 'Employee not found' }); return;

}

const userId = employeeResults[0].id; console.log('userid',userId);

const query = ` SELECT

travel\_request\_table.id, travel\_request\_table.trip\_id, travel\_request\_table.departure\_date, travel\_request\_table.departure\_time, travel\_request\_table.return\_date,

travel\_request\_table.return\_time, travel\_request\_table.accompanying\_count, travel\_request\_table.opt\_for\_advance, travel\_request\_table.adv\_accomodation\_amount, travel\_request\_table.adv\_travel\_amount, travel\_request\_table.employee\_id, purpose\_of\_visit\_table.purpose\_of\_visit, billing\_entity\_table.billing\_entity, travel\_mode\_table.travel\_mode, status\_table.status, employee\_table.employee\_id AS rejector\_id, travel\_request\_table.rejector\_remark, travel\_request\_table.accomodation\_address, travel\_request\_table.ticket\_details, ct\_from.city\_name AS from\_city\_name, ct\_to.city\_name AS to\_city\_name, trip\_type\_table.trip\_type

FROM travel\_request\_table

LEFT JOIN purpose\_of\_visit\_table ON travel\_request\_table.purpose\_of\_visit\_id = purpose\_of\_visit\_table.id

LEFT JOIN billing\_entity\_table ON travel\_request\_table.billing\_entity\_id = billing\_entity\_table.id

LEFT JOIN travel\_mode\_table ON travel\_request\_table.travel\_mode\_id = travel\_mode\_table.id

LEFT JOIN status\_table ON travel\_request\_table.status\_id = status\_table.id

LEFT JOIN employee\_table ON travel\_request\_table.rejector\_id=employee\_table.id LEFT JOIN city\_table ct\_from ON travel\_request\_table.from\_city\_id = ct\_from.city\_id LEFT JOIN city\_table ct\_to ON travel\_request\_table.to\_city\_id = ct\_to.city\_id

LEFT JOIN trip\_type\_table ON travel\_request\_table.trip\_type = trip\_type\_table.trip\_type\_id WHERE travel\_request\_table.employee\_id = ?

connection.query(query, [userId], (err, travelResults) => { if (err) {

console.error('Error querying the database for travel data: ' + err.stack); res.status(500).json({ error: 'Database error' });

return;

}

res.send(travelResults);

});

});

});

app.post("/team\_travel\_request\_details", (req, res) => { const uid = req.body.uid;

const approverLevelId = req.body.approverLevelId;

// Assuming you have an 'employee\_table' with a 'user\_id' column

const getEmployeeIdQuery = "SELECT id FROM employee\_table WHERE employee\_id = ?"; connection.query(getEmployeeIdQuery, [uid], (err, employeeResults) => {

if (err) {

console.error('Error querying the database for employee ID: ' + err.stack); res.status(500).json({ error: 'Database error' });

return;

}

if (employeeResults.length === 0) { console.error('Employee not found in the database.'); res.status(404).json({ error: 'Employee not found' }); return;

}

// Get the `employee\_id` from the request body const employeeId = employeeResults[0].id;

const getCompanyQuery = "SELECT company\_id FROM employee\_table WHERE id = ?"; connection.query(getCompanyQuery, [employeeId], (err, companyResults) => {

if (err) {

console.error('Error querying the database for company ID: ' + err.stack); res.status(500).json({ error: 'Database error' });

return;

}

if (companyResults.length === 0) { console.error('Company ID not found for the employee.');

res.status(404).json({ error: 'Company ID not found' }); return;

}

const companyId = companyResults[0].company\_id;

const statusId = approverToStatusMapping[approverLevelId]; const findEmployeeIdsQuery = `

SELECT employee\_id

FROM approver\_mapping\_table WHERE approver\_employee\_id = ?;

`;

const getTravelRequestDetailsQuery = ` SELECT

travel\_request\_table.id, travel\_request\_table.trip\_id, travel\_request\_table.departure\_date, travel\_request\_table.departure\_time, travel\_request\_table.return\_date, travel\_request\_table.return\_time, travel\_request\_table.accompanying\_count, travel\_request\_table.opt\_for\_advance, travel\_request\_table.adv\_accomodation\_amount, travel\_request\_table.adv\_travel\_amount, travel\_request\_table.travel\_comment, travel\_request\_table.employee\_id, purpose\_of\_visit\_table.purpose\_of\_visit, billing\_entity\_table.billing\_entity, travel\_mode\_table.travel\_mode, status\_table.status, employee\_table.employee\_id AS rejector\_id, travel\_request\_table.rejector\_remark, travel\_request\_table.accomodation\_address, travel\_request\_table.ticket\_details, ct\_from.city\_name AS from\_city\_name, ct\_to.city\_name AS to\_city\_name, trip\_type\_table.trip\_type FROM travel\_request\_table

LEFT JOIN purpose\_of\_visit\_table ON travel\_request\_table.purpose\_of\_visit\_id = purpose\_of\_visit\_table.id

LEFT JOIN billing\_entity\_table ON travel\_request\_table.billing\_entity\_id = billing\_entity\_table.id

LEFT JOIN travel\_mode\_table ON travel\_request\_table.travel\_mode\_id = travel\_mode\_table.id

LEFT JOIN status\_table ON travel\_request\_table.status\_id = status\_table.id

LEFT JOIN employee\_table ON travel\_request\_table.rejector\_id=employee\_table.id LEFT JOIN city\_table ct\_from ON travel\_request\_table.from\_city\_id = ct\_from.city\_id LEFT JOIN city\_table ct\_to ON travel\_request\_table.to\_city\_id = ct\_to.city\_id

LEFT JOIN trip\_type\_table ON travel\_request\_table.trip\_type = trip\_type\_table.trip\_type\_id WHERE travel\_request\_table.status\_id=? AND travel\_request\_table.employee\_id IN (?);

`;

connection.query(findEmployeeIdsQuery, [employeeId], (err, employeeIdsResults) => { if (err) {

console.error('Error querying the database for employee IDs: ' + err.stack); res.status(500).json({ error: 'Database error' });

return;

}

// Extract employee IDs from the results

const employeeIds = employeeIdsResults.map((row) => row.employee\_id); console.log(employeeIds);

if (employeeIds.length === 0) { console.error('No eligible employees found.');

res.status(404).json({ error: 'No eligible employees found' }); return;

}

connection.query(getTravelRequestDetailsQuery, [statusId, employeeIds], (err,

travelResults) => { if (err) {

console.error('Error querying the database for travel request details: '); res.status(500).json({ error: 'Database error' });

return;

}

res.json(travelResults);

});

});

});

});

});

app.listen(3002, () => { console.log('Server is running');

});

## CONCLUSIONS AND FUTURE WORK

## CONCLUSION

In conclusion, the Reimbursement App stands as a pivotal solution designed to revolutionize and simplify the reimbursement processes within organizations. By leveraging the power of React.js, Node.js, and SQL, this application addresses the challenges associated with traditional reimbursement systems. The user-friendly interface, real-time tracking, and customization options align seamlessly with specific company policies, ensuring a streamlined and efficient reimbursement workflow. The adoption of SQL ensures not only data integrity and security but also provides a robust foundation for comprehensive reporting and auditing capabilities. The Reimbursement App is poised to enhance organizational efficiency, reduce administrative burdens, and contribute to employee satisfaction by modernizing the reimbursement experience.

## FUTURE ENHANCEMENT

Looking ahead, the Reimbursement App offers exciting opportunities for future enhancement. One avenue for development is the integration of advanced technologies, such as machine learning algorithms, to further automate expense categorization and improve accuracy. Exploring partnerships with financial institutions could facilitate seamless fund transfers, enhancing the overall reimbursement experience. Additionally, the potential development of mobile applications would allow for on-the-go submission, catering to the evolving needs of a dynamic workforce. Continuous updates to comply with changing regulations and the integration with emerging technologies will ensure the Reimbursement App remains at the forefront of innovation. The project's scalability and adaptability lay the groundwork for ongoing Work that will keep the application relevant and valuable in the ever-evolving landscape of organizational processes.