Financial and Housekeeping Outsourcing Management System

Main Project Report

Submitted by

George Clifford

RegNo: FIT22MCA-2060

Submitted in partial fulfillment of the requirements for the award of the degree of

Master of Computer Applications

of

A P J Abdul Kalam Technological University

FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)®

ANGAMALY - 683577, ERNAKULAM (DIST.)

MAY 2024

DECLARATION

I George Clifford, hereby declare that the report of this project work, submitted to the Department of Computer Applications, Federal Institute of Science and Technology (FISAT), Angamaly in partial fulfillment of the award of the degree of Master of Computer Applications is an authentic record of my original work.

The report has not been submitted for the award of any degree of this university or any other university.

Date:

Place: George Clifford

FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)®

ANGAMALY, ERNAKULAM-683577

DEPARTMENT OF COMPUTER APPLICATIONS

CERTIFICATE

This is to certify that the project report titled ***'Financial and Housekeeping Outsourcing Management System'*** *submitted by George Clifford (Reg No:FIT22MCA-2060) towards partial fulfillment of the requirements for the award of the degree of Master of ComputerApplications is a record of bonafide work carried out by her during the year 2024.*

Project Guide

Head of the Department

Submitted for the viva-voice held on ..... at .....

Examiner :

ACKNOWLEDGEMENT

I am extremely glad to present my main project which I did as a part of our curriculum. I take this opportunity to express my sincere thanks to those who helped me in bringing out the report of my project.

I am deeply grateful to **Dr. Jacob Thomas V**, Principal, FISAT, Angamaly and **Dr Mini P R**, Vice Principal, FISAT, Angamaly.

My sincere thanks to **Dr. Deepa Mary Mathews**, Head of the department of MCA, FISAT, who had been a source of inspiration. During the period of my project work, I have received generous help from **Dr. Deepa Mary Mathews**, my project guide, which I like to put on record here with deep gratitude and great pleasure.

My special gratitude to **Dr. Rakhi Venugopal**, for her encouragement and suggestion. I express my heartfelt thanks to all the faculty members in our department for their constant encouragement and never-ending support throughout the project.

I would like to pay gratitude and special thanks to my mentors Mr. Ashis D Palliyan, **Federal Bank** and Mr Jageeth Mohan for their constant support during the peroid of my internship work.

Finally, I am grateful to all my friends who gave me a lot of suggestion for the successful completion of this project.

George Clifford

ABSTRACT

Outsourcing operational activities to cut costs is a phenomenon that has been around for a bank. Banks have realized that they could partner with service providers to save money and get expert help from these outsiders. Banks leverage the advantages of outsourcing by focusing on strategic priorities, reducing operational costs, staying up to date with the latest industry trends, and strengthening their competitive edge in an ever-changing market landscape.

The project aims to develop an integrated Financial and Housekeeping Outsourcing Management System tailored for banks, intending to revolutionize how banks manage their outsourcing operations. The goal is to enable banks to achieve higher efficiency, cost savings, and competitiveness in the dynamic financial services landscape, leveraging the power of the MERN (MongoDB, Express.js, React.js, Node.js) stack. In today's competitive banking landscape, efficiency and cost-effectiveness are paramount for sustainable growth and customer satisfaction.

The traditional approach to managing financial and housekeeping tasks often involves cumbersome manual processes that are prone to errors and inefficiencies. The proposed system offers a comprehensive solution to this challenge by centralizing and automating various aspects of outsourcing management. By utilizing the MERN stack, the system ensures scalability, flexibility, and responsiveness, allowing seamless access across devices and platforms. MongoDB serves as the backbone for robust and scalable data storage, handling the diverse data needs of banking operations efficiently.

CONTENTS

CHAPTER 1
INTRODUCTION

1.1 COMPANY PROFILE

Federal Bank Limited is an Indian private sector bank headquartered in Aluva, Kochi, Kerala. The Bank has 1480+ banking outlets and, 1935+ ATMs/ CDMS spread across different states in India and overseas representative offices at Abu Dhabi and Dubai.

With a customer base of over 16.6 million, and a large network of remittance partners around the world, Federal Bank handles more than one fifth of India's total inward remittances. The bank has remittance arrangements with more than 110 Banks/Exchange Companies around the world. The bank is also listed in the Bombay Stock Exchange, National Stock Exchange of India and London Stock Exchange and has a branch in India's first International Financial Services Centre (IFSC) at the GIFT City.

Federal Operations and Services Limited (Fedserv) is a subsidiary company of The Federal Bank Ltd, established under the Companies Act, 2013 on 26 October 2018. Fedserv operates from two locations, namely Kakkanad in Kerala and Vishakhapatnam in Andhra Pradesh. It serves as a strategic extension of The Federal Bank Ltd, facilitating various operational and service-related functions. Fedbank Financial Services Ltd (Fedfina) is the non-banking financial arm of Federal Bank. It operates independently from the bank and specializes in offering financial products and services outside traditional banking activities. Fedfina plays a crucial role in diversifying the financial offerings of The Federal Bank Ltd, catering to a broader range of customer needs in the financial sector.

1.2 PROBLEM STATEMENT

In the ever-evolving landscape of the banking sector, the efficient management of both financial and housekeeping operations is paramount. These operations are not only essential for the day-to-day functioning of a bank but also play a significant role in ensuring regulatory compliance, cost-effectiveness, and ultimately, customer satisfaction. However, despite the clear benefits of outsourcing certain functions to third-party service providers, many banks encounter challenges in doing so while upholding control, transparency, and security.

One of the primary challenges faced by banks is the need to balance the advantages of outsourcing with the inherent risks involved. Outsourcing can offer cost savings, access to specialized expertise, and increased operational flexibility. However, it also introduces complexities related to risk management, data protection, and maintaining service quality standards.
The primary issues faced by banks in outsourcing financial and housekeeping services include:

- Risk Management: Ensuring that outsourced services comply with regulatory standards and internal policies to mitigate financial, operational, and reputational risks.
- Cost Optimization: Maximizing cost efficiencies through strategic outsourcing decisions, vendor selection, and contract negotiation while maintaining service quality.
- Operational Oversight: Establishing robust mechanisms for monitoring and evaluating outsourced services to ensure adherence to service level agreements (SLAs) and performance metrics.
- Service Quality Assurance: Implementing measures to assess and improve the quality of outsourced financial and housekeeping services to meet the bank's standards and customer expectations.

- Vendor Relationship Management: Developing and nurturing strong partnerships with outsourcing vendors to foster collaboration, communication, and continuous improvement initiatives.

To address these challenges, there is a critical need for a comprehensive Financial and Housekeeping Outsourcing Management System tailored specifically for the banking sector. This system should integrate robust risk assessment tools, performance monitoring dashboards, data security protocols, vendor management modules, and automation capabilities to streamline outsourcing processes and enhance overall operational efficiency.

CHAPTER 2

SYSTEM STUDY

In the dynamic landscape of the banking sector, efficient management of financial and housekeeping operations is paramount for ensuring regulatory compliance, cost-effectiveness, and customer satisfaction. However, many banks face significant challenges in effectively outsourcing these critical functions to third-party service providers while maintaining control, transparency, and security.

This project aims to address these challenges by developing a comprehensive Financial and Housekeeping Outsourcing Management System tailored specifically for the banking sector. This system will integrate advanced technologies, robust risk assessment tools, performance monitoring dashboards, data security protocols, vendor management modules, and automation capabilities to streamline outsourcing processes and enhance overall operational efficiency.

2.1 EXISTING SYSTEM

The existing system of financial and housekeeping outsourcing management in the banking sector can vary widely depending on the specific practices and technologies adopted by different banks. However, there are some common elements and challenges that are typically found in these systems.

The current state of financial and housekeeping outsourcing management in the banking sector is marked by several challenges. Banks heavily rely on manual processes for tasks like vendor selection, contract negotiation, performance monitoring, and compliance checks, leading to time-consuming and error-prone operations. Moreover, disparate systems are often used, causing data silos and inefficiencies due to fragmented oversight. Automation is limited, particularly in areas like report generation, SLA compliance tracking, and risk assessments, resulting in delays and operational inefficiencies. Risk management remains complex, with gaps in comprehensive risk assessment tools and strategies, especially concerning regulatory compliance, data security, and vendor stability. Data security concerns

persist due to inadequate protection measures, posing vulnerabilities in data handling and transmission. Additionally, banks face challenges in maintaining transparency and visibility into outsourced services' performance, costs, and compliance status, impacting decision-making and risk management efforts. Managing vendor relationships is also challenging, with fragmented oversight and limited centralized vendor management capabilities leading to inefficiencies in onboarding, contract management, performance evaluation, and dispute resolution. Overall, there is a clear need for integrated, automated, and robust outsourcing management systems to address these multifaceted challenges and enhance operational efficiency in the banking sector.

2.2 PROPOSED SYSTEM

The proposed Financial and Housekeeping Outsourcing Management System represents a comprehensive solution designed to transform how banks handle their outsourcing relationships. At its core, the system offers a centralized platform that consolidates all outsourcing-related activities, from vendor selection and contract management to performance monitoring and risk assessment. This centralized approach aims to eliminate data silos, enhance communication channels, and provide greater transparency and visibility into every aspect of outsourcing operations.

Key to the system is its vendor management module, which streamlines the onboarding process for new vendors, facilitates contract negotiations, tracks vendor performance against predefined metrics, and enables swift resolution of any disputes that may arise. This module empowers banks to proactively manage vendor relationships, ensuring adherence to service level agreements (SLAs) and regulatory requirements while fostering collaboration and efficiency.

To mitigate risks inherent in outsourcing, the system integrates advanced risk assessment tools. These tools evaluate vendor stability, regulatory compliance, data security practices, and service continuity, allowing banks to identify, assess, and mitigate potential risks effectively. Real-time performance monitoring dashboards complement these tools by providing actionable insights into service uptime, response times, compliance status, and customer satisfaction ratings, enabling banks to uphold service quality standards and meet customer expectations.

2.3 OBJECTIVE

The objectives of the Financial and Housekeeping Outsourcing Management System project are multifaceted and aim to address key challenges faced by banks in managing outsourcing relationships effectively.

The project aims to streamline the process of requesting new engagements, starting from the business requestor's initial request, which is then reviewed and processed by the business controller. This request undergoes vendor selection approval by the appropriate approver and reviewer committee, ensuring that only suitable vendors are considered for engagement. Upon vendor selection approval, the business requestor proceeds with the new vendor registration, completing a series of questionnaires that include risk assessment components. This step is crucial for evaluating potential risks associated with the new vendor partnership. The project emphasizes the importance of thorough risk assessment to mitigate financial, operational, and reputational risks that may arise from outsourcing engagements.

Following the risk assessment, the approver reviews the results and decides whether to approve or reject the placement of the purchase order with the selected vendor. Once approved, the project incorporates mechanisms for ongoing vendor auditing, periodic risk assessment, and regulatory reporting to ensure continued compliance and risk mitigation throughout the engagement.

Moreover, the system facilitates performance monitoring of vendors, tracking delivery standards, service quality, and adherence to service level agreements (SLAs). This focus on performance monitoring enables banks to assess vendor performance objectively and make data-driven decisions regarding ongoing engagements.

CHAPTER 3

SYSTEM SPECIFICATON

The Financial and Housekeeping Outsourcing Management System is built on a robust technological stack to ensure scalability, reliability, and efficiency in managing outsourcing engagements within the banking sector. The system's specifications encompass hardware and software components, leveraging modern technologies to deliver a seamless user experience and comprehensive functionality.

3.1 HARDWARE SPECIFICATIONS

The selection of hardware is very important in the existence and proper working of any software. when selecting the hardware, the size and capacity requirements are also important. Below is some of the hardware that is required by the system.

Processor	Intel Core i3-3220 (3.3 GHz) or above
RAM	4 GB or above
Storage	512 GB or above
Other	Keyboard and Mouse

3.2 SOFTWARE SPECIFICATION

3.2.1 FRONTEND DEVELOPMENT

The frontend of the system is developed using **React.js**, a popular JavaScript library for building interactive user interfaces. React.js has gained widespread popularity for building dynamic and interactive user interfaces (UIs) in web applications. One of the key features that sets React.js apart is its component-based architecture. In React, the UI is broken down into reusable components, each representing a specific part of the interface. These components encapsulate their own logic, state, and rendering, making

them modular and easy to maintain. Developers can create complex UIs by composing these reusable components, leading to a more organized and scalable codebase.

Another strength of React.js is its use of JSX (JavaScript XML) syntax. JSX allows developers to write HTML-like code directly within JavaScript, which simplifies the process of combining UI markup with JavaScript logic. This declarative approach enhances readability and makes it easier to visualize and understand the structure of the UI. Additionally, JSX enables developers to embed JavaScript expressions within HTML, facilitating dynamic content rendering and data binding in React components. The React.js ecosystem is vast and vibrant, with a wide range of tools, libraries, and extensions that enhance development productivity and extend React's capabilities. Popular tools like React Router for client-side routing, Redux for state management, and Material-UI for pre-designed UI components complement.

3.2.2 BACKEND DEVELOPMENT

Node.js, as a server-side JavaScript runtime environment, is renowned for its asynchronous and event-driven architecture, which fundamentally alters how applications handle I/O operations. Traditional server-side environments can be synchronous, meaning they process requests one at a time, waiting for each operation to complete before moving to the next. In contrast, Node.js leverages an event loop that allows it to manage multiple connections concurrently without blocking the execution of other tasks. This non-blocking I/O model is especially beneficial for applications requiring real-time interactions or handling numerous concurrent users.

Express.js, built on top of Node.js, is a minimalist web framework that simplifies the development of web applications and APIs. It provides a structured approach to handling HTTP requests, defining routes, and integrating middleware for various tasks like request validation, authentication, and error handling. Express.js promotes a modular and organized codebase by separating concerns such as routing logic, middleware functions, and route handlers.

The combination of Node.js and Express.js is particularly powerful for building scalable and efficient backend systems. Node.js excels in handling asynchronous tasks and managing I/O operations efficiently, while Express.js streamlines the process of

creating RESTful APIs and managing HTTP routes. Together, they enable developers to build high-performance web applications with robust backend functionalities, such as data retrieval from databases, processing user requests, and delivering dynamic content to clients. This stack is well-suited for projects like the Financial and Housekeeping Outsourcing Management System, where responsiveness, scalability, and seamless API integration are critical requirements.

3.2.3 DATABASE DEVELOPMENT

MongoDB is a popular NoSQL database management system known for its flexibility, scalability, and performance in handling unstructured or semi-structured data. Unlike traditional relational databases, MongoDB follows a document-oriented data model, where data is stored in flexible, JSON-like documents. This schema-less approach allows for dynamic and agile data modeling, making MongoDB well-suited for modern applications with evolving data requirements. One of the key features of MongoDB is its ability to store and manage large volumes of data efficiently. It uses a distributed architecture and horizontal scaling techniques to distribute data across multiple servers or clusters, enabling high availability and fault tolerance. MongoDB's sharding capability further enhances scalability by partitioning data into shards based on a chosen shard key, allowing for linear scalability as data volumes grow.

MongoDB also provides robust data consistency and durability through features like replica sets and write concern options. Replica sets ensure data redundancy and failover protection by replicating data across multiple nodes, while write concerns allow developers to control the level of acknowledgment required for write operations, balancing performance and data safety.

CHAPTER 4

USER CHARACTERISTICS AND MODULES

4.1 USER CHARACTERISTICS

The Financial and Housekeeping Outsourcing Management System has five users to work collaboratively within the system to facilitate the management, oversight, decision-making, and execution of outsourcing activities, ensuring efficiency, compliance, risk mitigation, and optimal outcomes for the organization.

These user roles are:

- Administrator
- Business Requestor
- Business Controller
- Reviewer
- Approver

Administrator

Administrator or Admin is the super user and main controller of this system. Administrator controls all the activities of the system. Admin does the access management ie, the list of employees who access the system. Admin can activate or deactivate the employee access to the system. Admin can also view all the onboarding requests.

Business Requestor

The Business Requestor initiates requests for new engagements or services within the system. They gather requirements, submit requests, upload SLAs, monitor performance, ensure regulatory compliance, vendor audit and manage tasks related to outsourcing engagements.

Business Controller

The Business Controller controls the incoming requests made by the business requestor. They manage the overall activities of outsourcing engagements.

Reviewer

They review SLAs, performance metrics, compliance status, risk assessments, feasibility studies, and provide feedback, recommendations, or approvals based on their expertise and area of focus.

Approver

Approvers review and approve requests, proposals, contracts, and documents related to outsourcing engagements. They assess alignment with business objectives, regulatory compliance, financial implications, risk management strategies, and make informed decisions based on predefined criteria and guidelines.

4.2 MODULE DESCRIPTION

4.2.1 THIRD PARTY ONBOARDING

- This module manages the onboarding process for third-party vendors, suppliers, or service providers.
- It includes functionalities for new vendor engagement, service approvals and query management.
- The Business requestor requests for new engagement and Business Controller approves the requests

4.2.2 THIRD PARTY MANAGEMENT

- The Third-Party Management Module focuses on ongoing management and registration of new third-party relationships.
- It includes vendor audit also.
- Business Requestor can track vendor activities, assess performance against contractual obligations, and address issues or disputes as they arise.

4.2.3 SERVICE LEVEL AGREEMENT

- The SLA Management Module manages SLAs between the organization and third-party vendors/service providers.
- It includes SLA creation, negotiation, review and upload SLA.
- Downloadable Draft SLA is also provided.

4.2.4 PERIODIC RISK ASSESSMENT

- This module conducts regular risk assessments to identify, evaluate, and mitigate risks associated with outsourcing engagements.
- It includes risk identification, risk analysis, risk rating, risk mitigation strategies, and risk monitoring.
- Business Requestor can assess potential risks, implement controls, monitor risk exposure, and update risk profiles as needed to ensure risk management effectiveness.

4.2.5 PERFORMANCE MONITORING

- This module tracks and evaluates the performance of outsourcing engagements, vendors, and service providers.
- It includes monitoring key performance level indicators like rating level and their area of performance.
- Business requestor can generate performance reports and add new monitored performance.

4.2.6 REGULATORY REPORTING

- The Regulatory Reporting Module is responsible for generating reports to demonstrate compliance with regulatory requirements.
- The business requestor can make the purchase order.
- It shows the amount details and filters can be applied.

4.2.7 TASK MANAGEMENT

- Admin can assign task to business requestor and these assigned tasks can be viewed in a table

4.2.8 ACCESS MANAGEMENT

- This module controls and manages user access to system resources, data, and functionalities.
- It includes user authentication, authorization, role-based access control, permissions management.
- Admin can assign roles, define access levels, activate or revoke permissions to different employees.

CHAPTER 5

SYSTEM DESIGN

5.1 INPUT DESIGN

Input design is the process of converting a user-oriented description of the inputs to a computer-based system into a programmer-oriented specification. The quality of system input determines the quality of system output. Input specification describes the manner in which data enter the system for processing. Input design features can ensure the reliability of the system and produce result from accurate data or they can result in the production of errors. The input design also determines whether the user can interact efficiently with the system.

Input design requires consideration of the needs of the data entry operator. Three data entry considerations are:

- The field length must be documented
- The sequence of fields must match the sequence of the fields on the source document.
- The data format must be identified to the data entry operator.

In our system almost all inputs are being taken from the databases. To provide adequate inputs we have to select necessary values from the databases and arrange it to the appropriate controls.

Inaccurate input data are the most common cause of errors in data processing. Errors entered by data entry can be controlled by input design. Input design is the process of converting user-oriented inputs to a computer-based format. There are three major approaches for entering data into the computer. They are menus, formatted forms and prompts.

5.2 OUTPUT DESIGN

One of the important features of an information system for users is the output it produces. Output is the information delivered to users through the information system. Without quality output, the entire system appears to be unnecessary that users will avoid using it. Uses generally merit the system solely by its output. In order to create the most useful output possible. One works closely with the user through an interactive process, until the result is considered to be satisfactory. Output design has been an ongoing activity almost from the beginning of the project. In the study phase, outputs were identified and described general in the project directive. A tentative output medium was then selected and sketches made for each output. In the feasibility analysis, a “best” new system was selected; its description identified the input and output media. In the design phase the system has included an evaluation and selection of specific equipment for the system.

Outputs from computer systems are required primarily to communicate the results of processing to the user. They are also used to provide a permanent copy of these results for later consultation.

5.3 DATABASE DESIGN

The system database is stored in MongoDB. In MongoDB, schemas and collections are fundamental components of its flexible and dynamic data model. MongoDB's approach to schema design is dynamic, allowing for documents within a collection to have varying structures without a predefined schema enforcement. This flexibility extends to the field level, enabling developers to add, modify, or remove fields from documents as needed, without affecting the overall collection. Unlike traditional relational databases with rigid schemas, MongoDB embraces a schema less approach where schemas evolve naturally as data is inserted or updated, making it well-suited for agile development and handling diverse data models efficiently. Collections in MongoDB serve as containers for grouping related documents, similar to tables in relational databases. However, MongoDB collections do not enforce a schema at the collection level, allowing documents within the same collection to have different structures or schemas.

DATABASE SCHEMA

1. Collection: users

Field	Datatype	Constraint	Description
_id	ObjectID	Primary key	User id
username	String	Not null	User login email
password	String	Not null	User password
user_role	String	Not null	User role
name	String	Not null	User name
department	String	Not null	User department
status	String	Not null	Login status

2. Collection: thirdparties

Field	Datatype	Constraint	Description
_id	ObjectID	Primary key	Document ID
thirdPartyId	ObjectID	Foreign Key	Third party ID
email	String	Not null	User login email
addressDetail	String	Not null	Address of vendor
typeOfServiceProvider	String	Not null	Service provider type
accessToCriticalData	String	Not null	Data access level
status	String	Not null	Request status

3. Collection: engagementrequests

Field	Datatype	Constraint	Description
_id	ObjectID	Primary key	Document ID
thirdPartyName	String	Not null	Third party name
typeOfWork	String	Not null	Third party work type
fromDate	String	Not null	Start date
toDate	String	Not null	End date
outsourcingType	String	Not null	Type of outsourcing
status	String	Not null	Request status
attachment	String	Not null	SLA

4. Collection: performances

Field	Datatype	Constraint	Description
_id	ObjectID	Primary key	Document ID
thirdPartyId	ObjectID	Foreign Key	Third party ID
performanceArea	String	Not null	Performance Area
date	String	Not null	Date of monitoring
rating	String	Not null	Rating given
comment	String	Not null	Comments given

5. Collection: regulatories

Field	Datatype	Constraint	Description
_id	ObjectID	Primary key	Document ID
thirdPartyId	ObjectID	Foreign Key	Third party ID
descrption	String	Not null	Description
cardNumber	String	Not null	ATM card number
nameOnCard	String	Not null	Card Holder
expirationDate	String	Not null	Expiry of card
cvv	String	Not null	Cvv of card
amount	String	Not null	Amout paid
date	String	Not null	Daye of payment

6. Collection: slas

Field	Datatype	Constraint	Description
_id	ObjectID	Primary key	Document ID
thirdPartyId	ObjectID	Foreign Key	Third party ID
status	String	Not null	Status of request
sla	String	Not null	Uploaded SLA

7. Collection: tasks

Field	Datatype	Constraint	Description
_id	ObjectID	Primary key	Document ID
thirdPartyId	ObjectID	Foreign Key	Third party ID
descrption	String	Not null	Description
assignTo	String	Not null	Department
priority	String	Not null	Priority level
dueDate	String	Not null	Task due date
status	String	Not null	Status of task

5.4 ENTITY RELATIONSHIP DIAGRAM

CHAPTER 6
SYSTEM TESTING

Testing is a set of activity that can be planned in advance and conducted systematically. Testing begins at the module level and work towards the integration of entire computers-based system. Nothing is completed without testing, as it is vital to the success of the system. System testing makes a logical assumption that if all parts of the system are corrected, the goal will not appear until months later. The process of evaluation a system by manual or automated means to verify that it satisfies specified requirements or to identify differences between expected and actual result.

System testing is the first Stage of implementation, which is aimed at ensuring that the system works accurately and efficiently before live operations commences. Testing is vital to the success of the system. System testing makes a logical assumption that if all the parts of the system are correct and the goal will be successfully achieved. A series of testing are performed for the proposed system before the proposed system is ready for user acceptance testing.

Levels of Testing:
1. Unit testing
2. Integration testing
3. Validation testing
4. Output testing

6.1 UNIT TESTING

In this each module is tested individually before integrating it to the final system. Unit test focuses verification in the smallest unit of software design in each module. This is also known as module testing as here each module is tested to check whether it is producing the desired output and to see if any error occurs.

6.2 INTEGRATION TESTING

Integration testing is a systematic technique for constructing the program structure while at the same time conducting test to uncover errors associated with interfacing. The objective is to take unit tested modules and to combine them and test it as a whole. In this step all errors are encountered are corrected for next testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing. The purpose of integration testing is to verify functional, performance, and reliability requirements placed on major design items.

6.3 OUTPUT TESTING

No system could be useful if it does not produce the required output in the specific format. Output testing is performed to ensure the correctness of the output and its format. The output generated or displayed by the system is tested asking the users about the format required by them.

6.4 VALIDATION TESTING

Here the inputs are given by the user validated. That is password validation, format of data are correct, textbox validation. Changes are need to be done after result of this testing. Verification testing runs the system in a simulated data. Validation refers to the process of using software in order to find errors. The feedback from the validation phase generally produces changes in software to deal with errors and failures that are uncovered. Validation may continue for several months. During the course of validating the system, failure may occur and the software will be changed. Continued use may produce additional failures and need for still more changes. Proper validation checks redone in case if insertion and updating of tables, in order to see that no duplication of data has occurred.

CHAPTER 7
SYSTEM IMPLEMENTATION AND MAINTENANCE

7.1 SYSTEM IMPLEMENTATION

Implementation is the stage in the project where the theoretical design is turned into a working system and is giving confidence on the new system for the user that will work efficiently and effectively. It involves careful planning, investigation of the current system and its constraints on implementation, design of methods, to achieve the change over an evaluation of changes over methods. Apart from planning major implementation process begins with preparing a plan for the implementation of the system. According to these plans the activities are to be carried out, discussions made regarding the equipment and resources and the additional equipment has to be acquired to implement the new system.

Implementation is the final and important phase. The most critical stage in achieving a new successful system and giving users confidences that the new system will work effectively. The system can be implemented only after through testing is done and if it found to be working according to the specification. This method also offers the greatest security since the old system can takes if the error is found or inability to handle certain type of transaction while using the new system.

7.2 SYSTEM MAINTENANCE

There are four types of maintenance, namely, corrective, adaptive, perfective, and preventive. Corrective maintenance is concerned with fixing errors that are observed when the software is in use. Adaptive maintenance is concerned with the change in the software that takes place to make the software adaptable to new environment such as to run the software on a new operating system. Perfective maintenance is concerned with the change in the software that occurs while adding new functionalities in the software. Preventive maintenance involves implementation changes to prevent the occurrence of errors. The distribution of types of maintenance by type and by

percentage of time consumed. Corrective maintenance deals with the repair of faults or defects found in day-today system functions. A defect can result due to errors in software design, logic and coding. Design errors occur when changes made to the software are incorrect, incomplete, wrongly communicated, or the change request is misunderstood. Logical errors result from invalid tests and conclusions, incorrect implementation of design specifications, faulty logic flow, or incomplete test of data. All these errors, referred to as residual errors, prevent the software from conforming to its agreed specifications. Note that the need for corrective maintenance is usually initiated by bug reports drawn by the users.

In the event of a system failure due to an error, actions are taken to restore the operation of the software system. The approach in corrective maintenance is to locate the original specifications in order to determine what the system was originally designed to do. However, due to pressure from management, the maintenance team sometimes resorts to emergency fixes known as patching. Corrective maintenance accounts for 20% of all the maintenance activities.

Adaptive maintenance

Adaptive maintenance is the implementation of changes in a part of the system, which has been affected by a change that occurred in some other part of the system. Adaptive maintenance consists of adapting software to changes in the environment such as the hardware or the operating system. The term environment in this context refers to the conditions and the influences which act (from outside) on the system. For example, business rules, work patterns, and government policies have a significant impact on the software system. For instance, a government policy to use a single 'European currency' will have a significant effect on the software system. An acceptance of this change will require banks in various member countries to make significant changes in their software systems to accommodate this currency. Adaptive maintenance accounts for 25% of all the maintenance activities.

Perfective maintenance

Perfective maintenance mainly deals with implementing new or changed user requirements. Perfective maintenance involves making functional enhancements to the system in addition to the activities to increase the system's performance even when the changes have not been suggested by faults. This includes enhancing both the function

and efficiency of the code and changing the functionalities of the system as per the users' changing needs. Examples of perfective maintenance includes modifying the payroll program to incorporate a new union settlement and adding a new report in the sales analysis system. Perfective maintenance accounts for 50%, that is, the largest of all the maintenance activities.

Preventive maintenance

Preventive maintenance involves performing activities to prevent the occurrence of errors. It tends to reduce the software complexity thereby improving program understand ability and increasing software maintainability. It comprises documentation updating, code optimization, and code restructuring. Documentation updating involves modifying the documents affectedby the changes in order to correspond to the present state of the system. Code optimization involves modifying the programs for faster execution or efficient use of storage space. Code restructuring involves transforming the program structure for reducing the complexity in source code and making it easier to understand. Preventive maintenance is limited to the maintenance organization only and no external requests are acquired for this type of maintenance. Preventive maintenance accounts for only 5% of all the maintenance activities.

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

8.1 CONCLUSION

In conclusion, the Financial and Housekeeping Outsourcing Management System presents a comprehensive and integrated solution for banks and financial institutions to efficiently manage outsourcing operations. By leveraging modern technologies such as React.js, Node.js, MongoDB, and Express, the system addresses key challenges faced in outsourcing management, including manual processes, disparate systems, limited automation, risk management complexities, data security concerns, and transparency issues.

The system's modular architecture allows for flexible customization and scalability, accommodating diverse user roles and functionalities tailored to specific business requirements. From user management to engagement monitoring, regulatory reporting, third-party onboarding, access management, task management, periodic risk assessment, and service level agreement management, each module plays a crucial role in streamlining processes, ensuring compliance, mitigating risks, optimizing performance, and enhancing transparency.

In a dynamic banking sector landscape, where regulatory requirements, cost pressures, and customer expectations continue to evolve, the Financial and Housekeeping Outsourcing Management System stands as a reliable and innovative solution to navigate the complexities of outsourcing management, drive operational excellence, and deliver value to stakeholders and customers alike.

8.2 FUTURE ENHANCEMENT

In future iterations, the Financial and Housekeeping Outsourcing Management System can be enhanced with advanced features and technologies to stay at the forefront of outsourcing management solutions. One potential avenue for improvement is the implementation of advanced reporting and analytics capabilities, including interactive dashboards and predictive analytics, to provide deeper insights into vendor performance, risk prediction, and cost optimization. Integrating blockchain technology could enhance security, transparency, and automation in outsourcing transactions and contract management, leveraging smart contracts for automated execution and dispute resolution. Additionally, AI-powered automation, such as document processing and chatbot support, could streamline tasks and improve user experience. Enhancing collaboration with vendors through collaborative tools and real-time feedback mechanisms can foster better relationships and performance evaluation.

CHAPTER 9

CODING

9.1 SOURCE CODE

UserDetailsTable.jsx

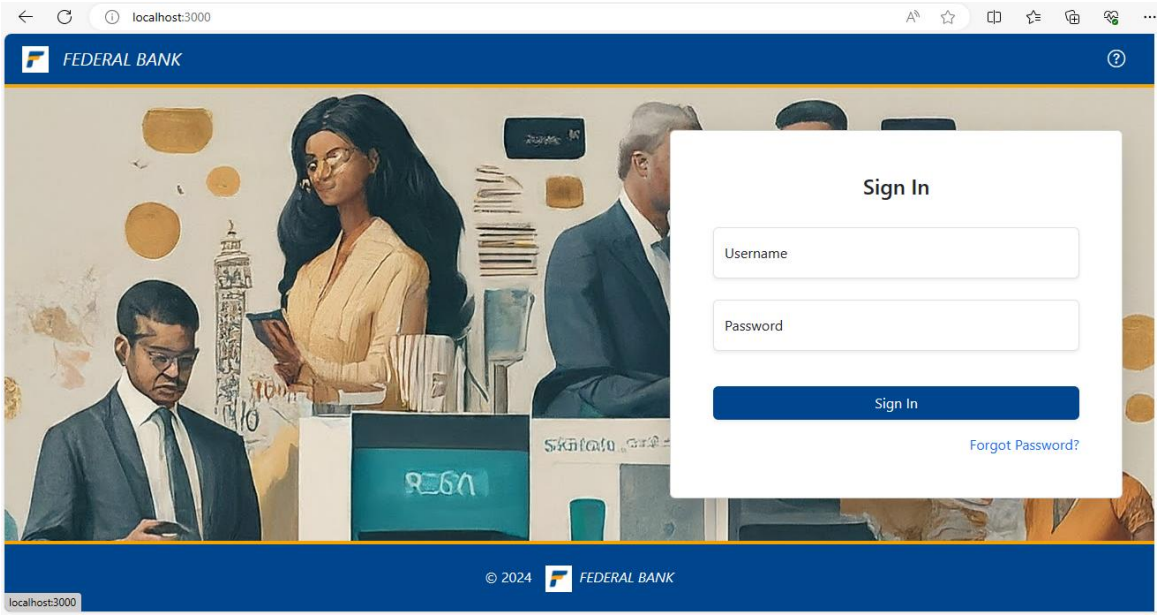
```
import React, { useState, useEffect, useRef } from "react";
import axios from "axios";
import enStrings from "../../localization/en.json";
import EditUserRole from "./EditUserRole";
import { ToastContainer, toast } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";

const UserDetailsTable = ({ searchQuery, departmentFilter, roleFilter }) => {
const [users, setUsers] = useState([]);
const tableRef = useRef(null);

// Function to fetch user data
const fetchUsers = async () => {
try {
const response = await axios.get("http://localhost:5000/api/outsourcing/users");
setUsers(response.data);
} catch (error) {
toast.error(error);
}
};

const handleActivate = async (userId) => {
try {
await axios.put(`http://localhost:5000/api/outsourcing/users/${userId}/activate`);
// Refresh user data after activation
fetchUsers();
toast.success(enStrings.userActivated);
} catch (error) {
console.error("Error activating user:", error);
toast.error(enStrings.activationFailed);
}
};

const handleDeactivate = async (userId) => {
try {
await axios.put(`http://localhost:5000/api/outsourcing/users/${userId}/deactivate`);
```

```
toast.success(enStrings.userDeactivated);
} catch (error) {
console.error("Error deactivating user:", error);
toast.error(enStrings.deactivationFailed);
}
};

const filteredUsers = users.filter((user) => {
const matchesSearch =
user.username.toLowerCase().includes(searchQuery.toLowerCase()) ||
user.name.toLowerCase().includes(searchQuery.toLowerCase());

const matchesDepartment =
departmentFilter === "all" || user.department === departmentFilter;

const matchesRole = roleFilter === "all" || user.user_role === roleFilter;

return matchesSearch && matchesDepartment && matchesRole;
});

const handleUpdateRole = async () => {
fetchUsers();
toast.success(enStrings.roleEdited);
};

// useEffect to fetch user data on component mount
useEffect(() => {
fetchUsers();
}, []);

return (
<div className="row">
<ToastContainer autoClose={2000} />
<div className="col col-12 col-sm-12 col-md-12 col-lg-12 col-xl-12 col-xxl-12
table-responsive">
<table
className="table table-striped table-hover align-middle text-nowrap shadow mb-4"
id="user_details"
ref={tableRef}>
<thead>
<tr>
<th scope="col">{enStrings.slNo}</th>
<th scope="col">{enStrings.username}</th>
<th scope="col">{enStrings.name}</th>
<th scope="col">{enStrings.department}</th>
<th scope="col">{enStrings.role}</th>


<th scope="col">{enStrings.actions}</th>
```

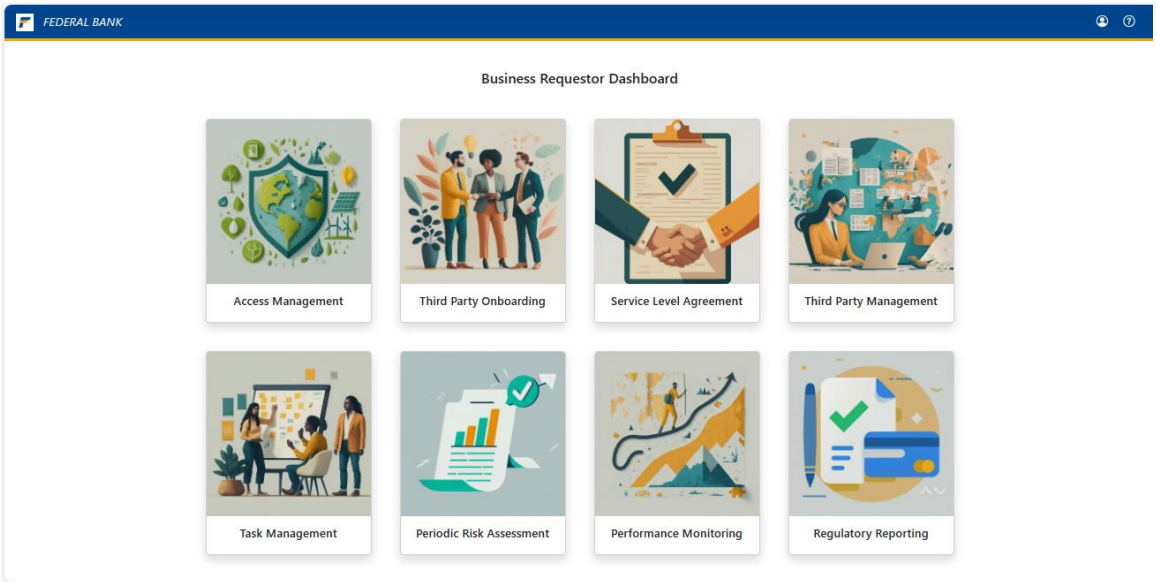
```
</tr>
</thead>
<tbody className="table-group-divider">
{filteredUsers.map((user, index) => (
<tr key={user._id}>
<th scope="row">{index + 1}</th>
<td>{user.username}</td>
<td>{user.name}</td>
<td>{user.department}</td>
<td>{user.user_role}</td>
<td>
<EditUserRole
userId={user._id}
userRole={user.user_role}
onUpdateRole={handleUpdateRole}
/>
{user.status === "Inactive" ? (
<button
type="button"
className="btn btn-success m-1"
title={enStrings.activateEmployee}
onClick={() => handleActivate(user._id)}>
<i className="bi bi-check-circle"></i>
</button>
) : (
<button
type="button"
className="btn btn-danger m-1"
title={enStrings.deactivateEmployee}
onClick={() => handleDeactivate(user._id)}>
<i className="bi bi-x-circle"></i>
</button>
)}
</td>
</tr>
))}
</tbody>
</table>
</div>
</div>
);
};

export default UserDetailsTable;
```

9.2 SCREENSHOTS

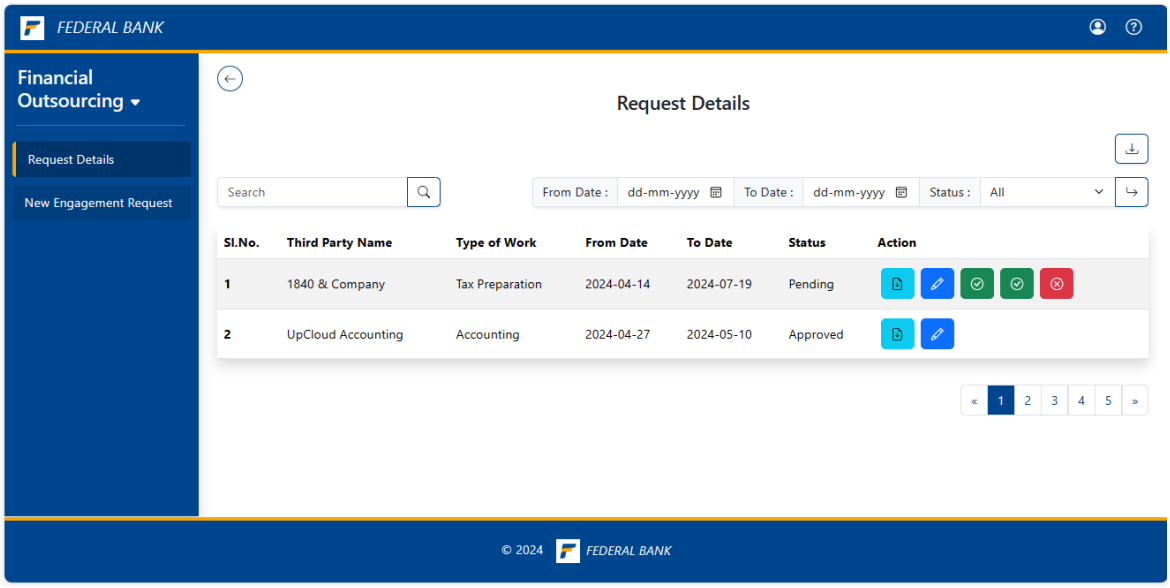
9.2.1 Home Screen



9.2.2 Business Requestor Home Page



9.2.3 Financial Outsourcing Request Details



REFERENCES

- https://getbootstrap.com/
- https://stackoverflow.com/
-