Industrial Internship Report on

Banking Information System

Prepared by

Chinmayi Anand Vaidya

Executive Summary
This report provides details of the Industrial Internship provided by upskill Campus and The IoT Academy in collaboration with Industrial Partner UniConverge Technologies Pvt Ltd (UCT). This internship was focused on a project/problem statement provided by UCT. We had to finish the project including the report in 6 weeks' time. My project was (Tell about ur Project) This internship gave me a very good opportunity to get exposure to Industrial problems and design/implement solution for that. It was an overall great experience to have this internship.

TABLE OF CONTENTS

1 Preface

a synopsis of the entire six-week project. Set out to design, construct, and refine a comprehensive banking management system over the course of six weeks. This project gave us the chance to put my technological expertise to use while also learning more about the complex interactions between technology and the banking industry. Below is a synopsis of our weekly successes and important lessons learned:

a synopsis of the entire six-week project. i set out to design, create, and refine an extensive Banking Management System in a span of six weeks. This project offered the chance to apply technological expertise to use while also learning more about the complex interactions between technology and the banking industry. In the initial week,

Week 1: Project Initiation and Planning

During the first week, I kicked on the project by goals, objectives, and project scope. I conducted in-depth research into the requirements of a modern banking system, identified potential challenges, and outlined our development plan. This phase laid the foundation for the weeks ahead

Week 2: Architecture and Design

In the second week, I focused on designing the architecture of My project Banking Management System. I carefully considered the system's components, interactions, and the technologies to be utilized. This phase involved designing the database schema, user interfaces, and backend services. A modular architecture was chosen to ensure scalability and maintainability

Week 3 : Database Integration and User Interface

During the third week, we delved into the implementation of the database integration and user interfaces. We utilized Java Database Connectivity(JDBC) to establish a connection to the relational database. Simultaneously, we began designing the graphical user interface (GUI) using Java's Swing framework. This phase allowed us to witness our project taking shape visually.

Week 4 : Security and Authentication

Week four was dedicated to strengthening the security aspects of our system. We implemented user authentication mechanisms to ensure secure access to the system. Role-based access control was established to different between administrators, bank employees, and customers. We also explored encryption techniques to protect sensitive user data.

Week 5 : Transaction Processing and Testing Banking Information System

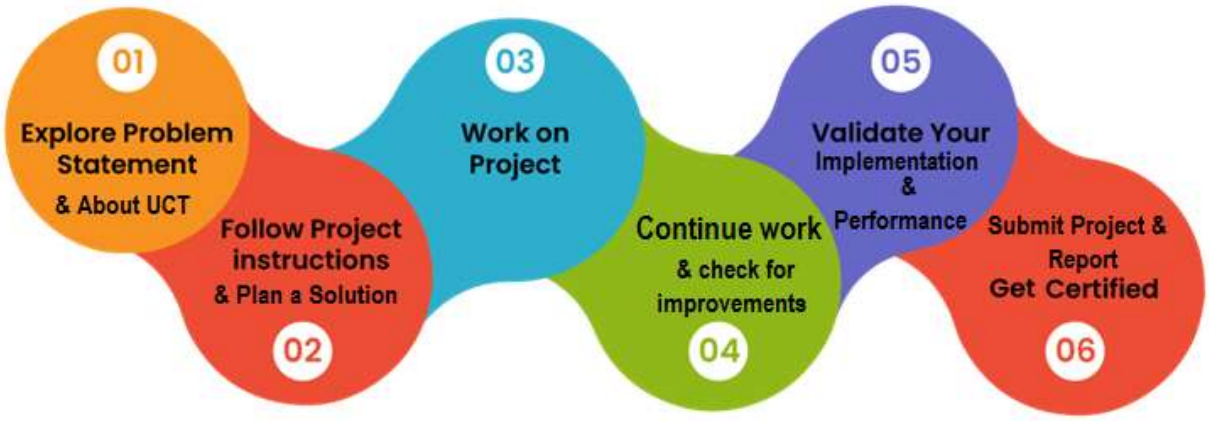
In the this week, we focused on developing the core functionality of our system: transaction processing. We built the logic for various transactions such as deposits, withdrawals, and fund transfers. Rigorous

testing took center stage, with a strong emphasis on unit testing, integration testing, and user acceptance testing. This phase revealed both the strengths and areas for improvement in our system

Week 6: Optimization, Documentation, and Report Preparation

In the final week, we dedicated my efforts to optimizing the system's performance and addressing any issues identified during testing .We improved response, minimized resource utilization, and ensured smooth transaction processing even under high loads.

Simultaneously, we emulously documented our development journey highlighting design decisions, obstacles surmounted, and solutions put into practice. This paper served as the foundation for our extensive report, which summarized our project from inception to completion. Over the course of these six weeks, we improved our technical proficiency in Core Java, database administration, and user interface design. We also honed our problem-solving, project management, and teamwork abilities. We gained the ability to prioritize activities, adjust to obstacles, and continuously refine our work. Our dedication, cooperation, and commitment to developing a significant solution that connects finance and technology are demonstrated by our Banking Management System. We are enthusiastic about the potential of our technology and the information we have gathered as we wrap up this project. We are eager to contribute our perspectives to the paper and illustrate how technology might revolutionize the banking industry for the bearer. About My Entire Project: A functioning prototype of the main functions of a local banking system is provided by the Core Java Banking Management System. The prototype ought to exhibit the fundamental characteristics and functionality of the system, highlighting its usability and functionality. The program was designed with a correct work division based on weeks, which ensures that the project is completed in an efficient and right manner without any hesitation or tension.



Learnings and overall experience:

Learning and overall experience was excellent and will help in future to grow myself as a good java . Thank to Upskill Campus for providing me the best help in the journey of my internship .

Message for Juniors and peers :-

Accept the Chance to Learn:

Creating a Banking Management System involves more than just writing code; it's an opportunity to comprehend the practical complexity of the banking sector and how technology is essential to its operations. Seize the chance to get knowledge about software development and the financial industry.

Group Cooperation:

The foundation of any successful project is teamwork. To produce a well-rounded solution, work together with your teammates, share ideas, and combine your collective abilities. Effective communication and cooperation are essential for conquering

obstacles. Arrange & Plan:

Take some time to design and organize your idea before you start coding.Clearly define objectives, divide work into doable portions, and create a timeline. A well-thought-out plan will help you stay on course and reduce unpleasant shocks Utilize .

Best Practices:

When developing software, follow industry best practices. Create code that is modular and maintainable, follow coding guidelines, and use safe coding techniques. You'll benefit from these practices for the duration of your career.User-focused design Consider the end users as you develop your system. Offering clients and bank personnel a positive experience requires an intuitive design and intuitive interactions.

Constant Enhancement:

As you advance, you may run against obstacles or opportunities for improvement. Accept that software development is an iterative process, and don't be afraid to make changes in response to feedback from users or fresh perspectives.
Acquire Knowledge from Errors Making mistakes is a normal aspect of learning. Instead of letting them depress you, see them as worthwhile educational opportunities that will help you become better developers in the long term.

Record Your Travels:

Preserve meticulous documentation of your choices, obstacles encountered, resolved issues, and the reasoning behind them. This document will be useful for your report, but it can also be used as a reference for other projects in the future.

Ask for Help:

Don't be afraid to ask for advice from your lecturers, mentors, or online communities if you run into challenges that don't seem to be solved. Seeking assistance is a show of strength rather than weakness.
Honor accomplishments:

Honor each accomplishment, no matter how tiny. A job this size is difficult to complete, therefore it's good to acknowledge your progress as it can inspire confidence and morale.

Think and Develop:

Once your project and report are finished, pause to reflect on the experience. Think back on your

experiences, your personal development, and the ways you might use these lessons in your future undertakings. Recall that this project is

2 Introduction

2.1 About UniConverge Technologies Pvt Ltd

A company established in 2013 and working in Digital Transformation domain and providing Industrial solutions with prime focus on sustainability and RoI.

For developing its products and solutions it is leveraging various **Cutting Edge Technologies e.g. Internet of Things (IoT), Cyber Security, Cloud computing (AWS, Azure), Machine Learning, Communication Technologies (4G/5G/LoRaWAN), Java Full Stack, Python, Front end** etc.

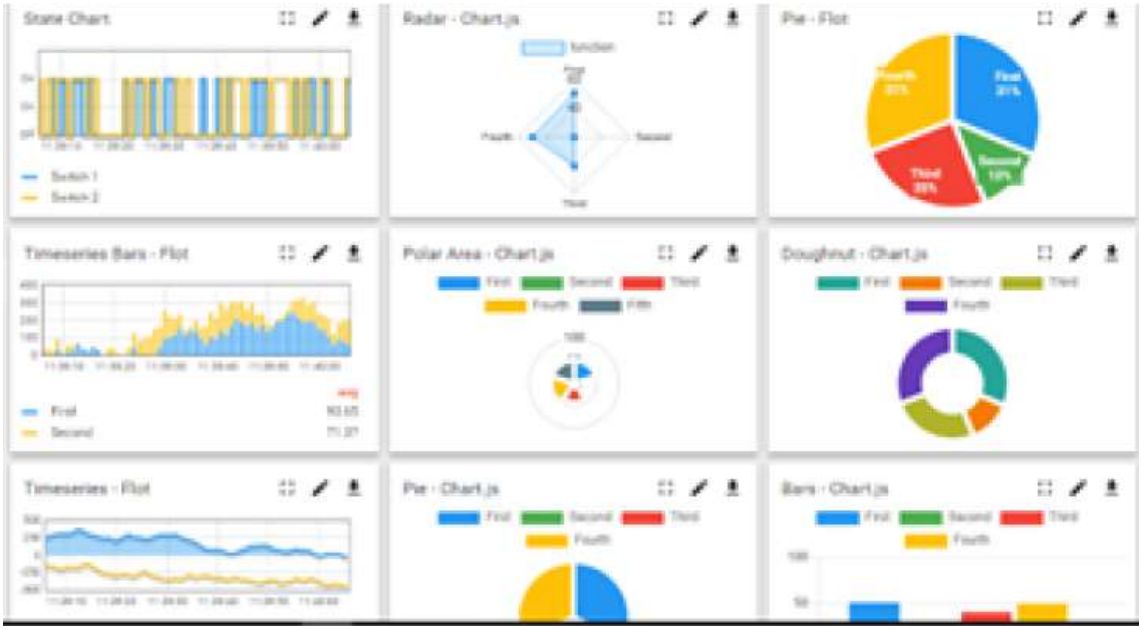
i. UCT IoT Platform (UCT Insight)

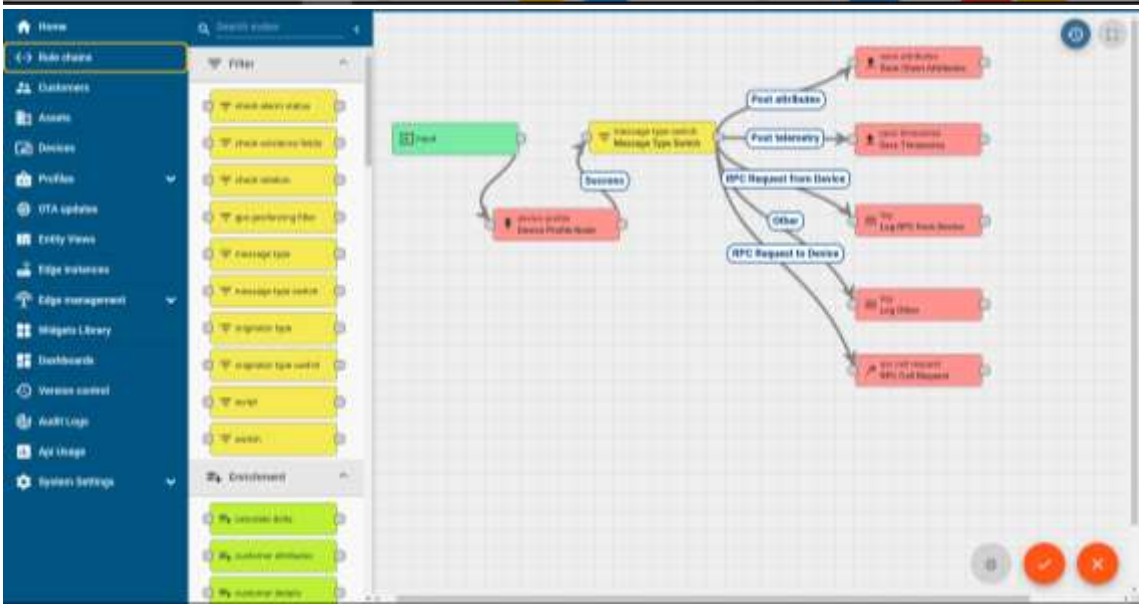
UCT Insight is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable "insight" for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

- It enables device connectivity via industry standard IoT protocols - MQTT, CoAP, HTTP, Modbus TCP, OPC UA
- It supports both cloud and on-premises deployments.

It has features to

- Build Your own dashboard
- Analytics and Reporting
- Alert and Notification
- Integration with third party application(Power BI, SAP, ERP)
- Rule Engine





ii. Smart Factory Platform (FACTORY WATCH)

Factory watch is a platform for smart factory needs.

It provides Users/ Factory

- with a scalable solution for their Production and asset monitoring
- OEE and predictive maintenance solution scaling up to digital twin for your assets.
- to unleased the true potential of the data that their machines are generating and helps to identify the KPIs and also improve them.
- A modular architecture that allows users to choose the service that they what to start and then can scale to more complex solutions as per their demands.

Its unique SaaS model helps users to save time, cost and money.


Demo Org
Demo
CNC_S7_81
CNC_S7_82
CNC_S7_84
Utilization
Utilization
Utilization

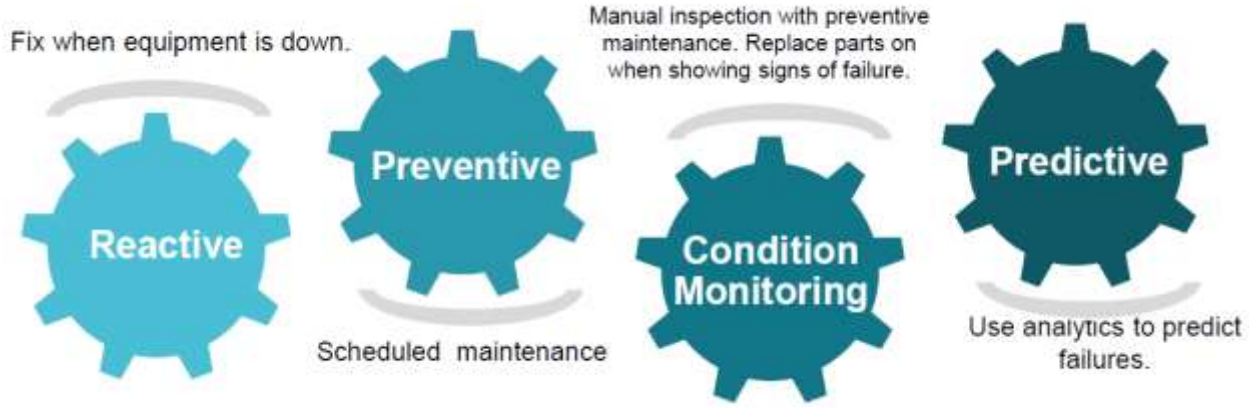

3D View
12:30
Search...
Stopped
Workplace Type
194724_27462-100-A3
Availability
Performance
Quality
OEE
Monitor
42.23%
400.00v
99.21%
99.21%
0.00%
0.00v
24.53%
24.53%
0.0%
200.00v
62.35%
62.35%

iii. LoRaWAN based Solution

UCT is one of the early adopters of LoRAWAN teschnology and providing solution in Agritech, Smart cities, Industrial Monitoring, Smart Street Light, Smart Water/ Gas/ Electricity metering solutions etc.

iv. Predictive Maintenance

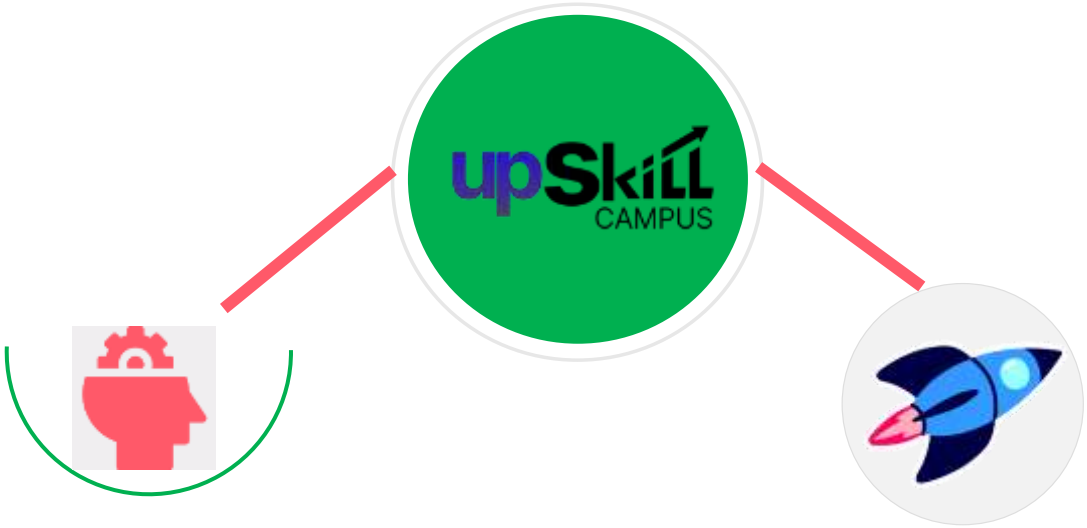
UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.



2.2 About upskill Campus (USC)

upskill Campus along with The IoT Academy and in association with Uniconverge technologies has facilitated the smooth execution of the complete internship process.

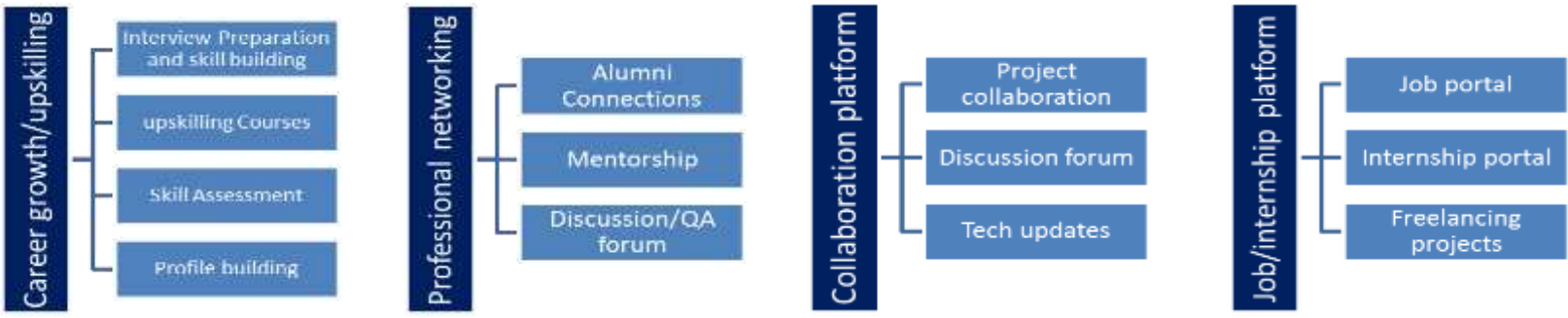
USC is a career development platform that delivers **personalized executive coaching** in a more affordable, scalable and measurable way.



Seeing need of upskilling in self paced manner along-with additional support services e.g. Internship, projects, interaction with Industry experts, Career growth Services

upSkill Campus aiming to upskill 1 million learners in next 5 year

https://www.upskillcampus.com/



2.3 The IoT Academy

The IoT academy is EdTech Division of UCT that is running long executive certification programs in collaboration with EICT Academy, IITK, IITR and IITG in multiple domains.

2.4 Objectives of this Internship program

The objective for this internship program was to

- ☛ get practical experience of working in the industry.
- ☛ to solve real world problems.
- ☛ to have improved job prospects.
- ☛ to have Improved understanding of our field and its applications.
- ☛ to have Personal growth like better communication and problem solving.

2.5 Reference

[1] Google.com

[2] Book – Spring in Action (For Spring Framework)

3 Problem Statement

The importance of technology to the banking industry cannot be overstated in the quickly changing financial landscape of today. Financial institutions must establish strong and sophisticated banking management systems in order to offer their clients smooth and efficient services. This study explores the use of the Core Java programming language in the design, development, and implementation of a banking management system.

The report's Banking Management System serves as evidence of Core Java's potential to provide more safe and versatile software solutions for the banking sector. Utilizing the built-in object-oriented capabilities, platform independence, and strong libraries of Core Java, a comprehensive system including customer account administration, transaction processing, security, and reporting has been developed. As stated in this study, We examine the basic ideas that guide the architecture of the Banking Management System. Emphasis is placed on the importance of inheritance, polymorphism, and encapsulation in facilitating modular and maintainable programming. Additionally, the integration of database management strategies for effective data retrieval and storage is covered, emphasizing the function of Java Database Connectivity (JDBC) in connecting to a relational database.

Any banking system must have security, and this paper describes the security features included into the Banking Management System. Strict security procedures are implemented to guarantee the confidentiality, integrity, and availability of sensitive financial data, from user authentication to data encryption .

Furthermore, the paper offers insights into the Banking Management System's user interface design. Account creation, transaction processing, and report generation are just a few of the duties that bank staff and administrators may easily complete with the help of an intuitive and user-friendly graphical interface. Java's Swing framework is used to construct the graphical user interface (GUI), which is an example of the combination of functionality and aesthetics.

A review of the system's potential improvements and future prospects comes to a close in this study. Opportunities to integrate cutting-edge technologies like artificial intelligence, machine learning, and blockchain into the banking industry are being investigated as technology advances.

These advances in the industrial possess the ability to completely transform banking operations, improving both client satisfaction and operational efficiency. It is important to note that the goal of this paper is to give a thorough overview of the Banking Management System created with Core Java.

Though the field of technology is ever-evolving, it's possible that new developments have happened since this research was finished. However, the approaches and ideas covered here provide a solid foundation for comprehending the complexities involved in creating sophisticated banking management systems with Core Java.

The Banking Management System, which is the culmination of technology and finance, is showcased in this paper. It shows how Core Java can be used to construct secure, efficient, and user-centric solutions for the modern banking industry.

4 Existing and Proposed solution

Current state and suggested remedy
4.1 Current ApproachIn the field of banking management systems, the current solution typically combines manual procedures with old software. These systems may be out-of-date, unable to grow, and vulnerable to security flaws. They may be limited in terms of real-time data updates, transaction processing speed, and user interfaces. Within the limitations of current systems, the tasks of integrating new technology, adapting to changing regulations, and meeting consumer expectations can be daunting.
4.2 Suggested RemediesThe suggested solution is to create a cutting-edge Banking Management System using Core Java in order to solve the flaws in the current systems and offer a reliable, safe, and easy-to-use framework for efficient banking operations. This solution makes use of the object-oriented programming, multithreading, and database connectivity features of Core Java to produce an all-encompassing system.

Modular design:

The suggested solution uses the encapsulation and inheritance concepts of Core Java to adopt a modular design. This makes it possible for various system components to be developed, tested, and maintained independently, which results in a more structured and manageable codebase.

Database Integration:

Java Database Connectivity (JDBC) is used to integrate the Banking Management System with a relational database. This makes it possible to store, retrieve, and manipulate customer data, account information, transaction history, and other data efficiently.

Graphical User Interface (GUI): The system makes advantage of Java's Swing framework to provide an intuitive GUI. With the help of this GUI, bank workers and administrators can easily communicate with the system, making it easier to complete duties like opening new accounts, handling transactions, and producing reports. In order to improve user experience, simplicity and intuitiveness are prioritized in GUI design.

Security Procedures:

Strong security features, such as role-based access control, data encryption, and user authentication, are implemented by the suggested solution. central Java's security libraries and procedures are utilized to protect sensitive consumer information and guarantee adherence to privacy regulations.
Core Transaction Processing The multi-threading capabilities of Java are utilized to enhance transaction processing. Real-time updates to account balances and transaction histories are made possible by concurrent execution of transactions, which guarantees quick and efficient operations.
A reporting module in the system produces thorough financial analyses and reports. Bank managers can make better decisions by learning more about account activity, transaction trends, and other important information.
Flexibility and Scalability:

Because of the scalable nature of the suggested solution, the system can handle expanding customer bases and rising transaction volumes. Furthermore, the flexibility of Core Java makes it simple to adjust to changing legal requirements and technology developments.
Prospect-Making:
The architecture of the solution is planned with the future in mind. In order to ensure the system's long-term relevance and competitiveness, it can act as a foundation for integrating cutting-edge technologies like artificial intelligence, machine learning, and blockchain into the banking sector.
In summary, the suggested Banking Management System utilizing Core Java provides a comprehensive solution to the issues encountered by conventional banking systems. The system intends to provide improved security, scalability, efficiency, and user experience by utilizing Core Java's capabilities, opening the door for a modern banking environment that satisfies the needs of the digital age.

4.1 Code submission (Github link)

https://github.com/chinmayi0212/upskillcampus.git

4.2 Report submission (Github link) :

https://github.com/chinmayi0212/upskillcampus.git

5 Proposed Design/ Model

Main Dashboard:

Grants access to different functionalities according to user roles. Facilitates the process for bank personnel to create new customer accounts.
Transcript of Transaction:
enables users to carry out transactions such as deposits, withdrawals, and money transfers.
Results and Evaluations Front:

Offers information on account activities, transactions, and trends.
Layer Controller:
Hetcaton au Roller: Oversees user animation and sound synchronization. Accounts Payable Coordinator: Manages account-related tasks like creation and updates.

Transacton Conroller: Account balance updates and orchestration of robotic processing.
Reporting Coordinator: Prepares reports and analyses according to user requests.

The Auhentcaton Service, located at the Service Layer, verifies user credentials and oversees user roles.
Support: Organizes business processes for account creation and updates.

Transaction Service: Updates account balances and does transaction validation.

Reporting Service: Generates reports and analyzes data with Excel and SAS.
Layer of Data Access:
Database Management System, or DBMS:
represents the relational database that contains customer data, account details, and historical records.

DAOs (Data Access Objectives):
For data access operations, such as CRUD (Create, Read, Update, and Delete) operations, provide an abstraction layer.
Layer of the Model:Model of User:
represents user data, such as role, password, and username.
Account Information Model: Describes the elements of a customer account, including account number, balance, and owner details.The Transaction Model describes the various types of defects, such as amoutes, and type of defects.
Libraries and Universities:

Communication between the application and the relational database is facilitated via Java Database Connectivity (JDBC).
The graphical user interface components are created using the Java Swing Framework.

Security libraries offer secure authentication, hashing, and encryption techniques. Future Improvements:
Including Emerging Technologies: Take into consideration the use of blockchain, AI, and ML for safe transactions and enhanced analytics. This design model offers a thorough framework for creating a Core Java Banking Management System. I explains the relationships between various layers and components, helping developers create a robust, secure, and scalable solution that complies with contemporary banking requirements.

6 Performance Test

Performance testing is crucial for evaluating the efficiency, responsiveness, and scalability of a Banking Management System. It ensures that the system can handle varying loads, maintain acceptable response times, and deliver a seamless user experience. Here's how you could conduct performance testing for your system

Identify Performance Metrics: Define key performance metrics such as response time (for different transactions), throughput (transactions per second), resource utilization (CPU, memory), and error rates.

Load Testing: Test the system's performance under expected and peak loads. Gradually increase the number of concurrent users and transactions to assess its breaking point and response times

Stress Testing: Apply extreme loads to the system to evaluate its behavior under stress conditions. This helps identify potential bottlenecks, resource exhaustion, and system crashes.

Endurance Testing: Run the system under continuous loads for an extended period to identify memory leaks, resource depletion, and degradation in performance over time.

Scalability Testing: Evaluate the system's ability to scale vertically (adding resources to a single machine) and horizontally (distributing load across multiple machines).

Peak Performance Testing: Test the system at the anticipated peak usage times to ensure it can handle high transaction volumes without degradation.

Real-User Simulation: Use realistic user scenarios to simulate user interactions, such as logging in, performing transactions, and generating reports, to replicate real-world usage patterns.

Database Performance Testing: Evaluate the system's interaction with the database by assessing query performance, indexing efficiency, and data retrieval times.

Network Latency Testing: Introduce network delays to mimic real-world network condition sand assess the system's response times and performance under varying latencies.

Third-Party Integration Testing: Assess the system's performance when interacting with third-party services, such as payment gateways, that are essential for banking operations.

Mobile and Web Performance Testing (if applicable): Evaluate the responsiveness and load handling capabilities of mobile and web interfaces across different devices and browsers.

Reporting Performance: Test the generation and delivery of reports, ensuring that the process doesn't adversely affect the system's responsiveness.

Analyzing Results: Analyze collected data to identify performance bottlenecks, resource constraints, and areas for improvement. Compare results against predefined performance metrics.

Optimization and Retesting: Address identified issues by optimizing code, improving database queries, or scaling resources. Retest the system to ensure improvements have been effective.

Scalability Assessment: Determine whether the system can handle increased loads by extrapolating test results to predict its behavior under higher user volumes.

7 My learnings

Constructing a Banking Management System via Core Java offers significant perspectives on a range of software development, project management, and domain-specific issues. The following are some important lessons that your report should emphasize:

Implementing Principles of Object-Oriented Design:

The system's development served as a reminder of the value of object-oriented concepts such as polymorphism, inheritance, and encapsulation. These guidelines encourage code reuse, modular design, and simplicity of maintenance.

DBA Integration Obstacles:

The significance of comprehending data modelling and effective database architecture was brought to light by integrating the system with a relational database. Better query optimization, indexing, and normalization all support data integrity and performance.

Difficulties in Real-World Applications:

Concurrent access, insufficient balances, and transaction failures are examples of real-world circumstances that the development of a banking system exposes the significance of managing.To tackle these obstacles, strong error management and validation systems are needed.

Testing and Quality Assurance:

To identify and recognize problems early in the development lifecycle, rigorous testing, such as unit testing, integration testing, and user acceptability testing, is necessary.Automated testing instruments and procedures lead to improved software quality.

Comments on Documentation and Code:

Keeping thorough and understandable documentation up to date, coupled with insightful code comments, is essential for future maintenance and knowledge transfer. Coding that is well-documented makes collaboration and troubleshooting easier.

In summary, creating a Banking Management System using Core Java provides a wealth of invaluable insights that go beyond technical expertise. It includes effective communication, problem-solving
In summary, creating a Banking Management System using Core Java provides a wealth of invaluable insights that go beyond technical expertise. It includes effective communication, problem-solving, teamwork, and knowledge of the complex interactions between technology and the financial sector.

These insights provide a comprehensive comprehension of software development and its practical applications in real-life situations.

8 Future work scope

The development of a Banking Management System using Core Java is a significant step towards modernizing banking operations. However, to keep up with evolving technologies and industry trends, there are several avenues for future enhancements and improvements. Here are some potential areas for future work :

8.1.1Integration of AI and ML:

Incorporating artificial intelligence and machine learning can enhance the system's capabilities. AI-powered chatbots for customer support, fraud detection algorithms, and predictive analytics for personalized financial recommendations are just a few possibilities.

8.1.2Blockchain Integration:

Implementing blockchain technology can improve transparency, security, and efficiency in areas like digital identity verification, cross-border payments, and secure record keeping.

8.1.3Mobile and Web Applications:

Developing mobile and web applications will allow customers to access their accounts and perform transactions from various devices, expanding convenience and accessibility.

8.1.4Enhanced Security Measures:

Implementing biometric authentication (fingerprint, facial recognition), two-factor authentication, and advanced encryption standards can further bolster the system's security