Industrial Internship Report on

# Console Based Expense Tracker

Prepared by

**Yash Sanodiya**

| *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).<br><br>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.<br><br>My project was Console Based Expense Tracker using JAVA<br><br>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

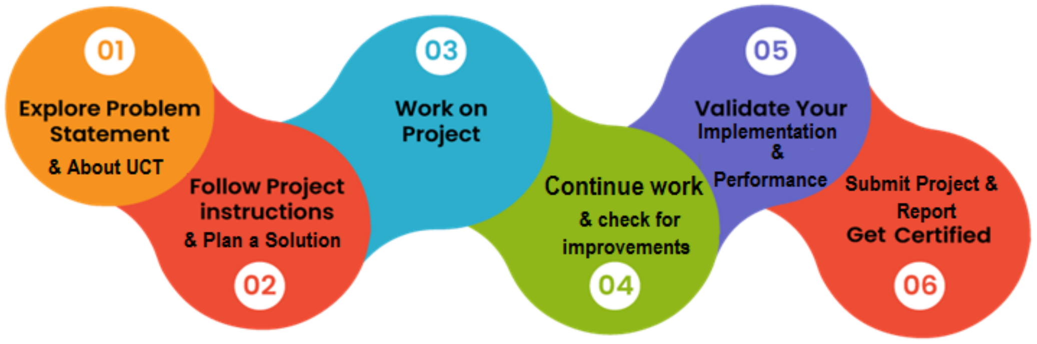
During the six-week Core Java Winter Internship program provided by Upskill Campus and UCT Technologies Pvt Ltd, I had the opportunity to delve into various aspects of Java programming and project development. This internship aimed to enhance our skills in Core Java while working on practical projects and submitting weekly reports on our progress through their online portal.

Relevant internships are vital for career growth, offering practical experience, skill honing, industry insight, and networking chances. They bridge theory and practice, fostering both technical and soft skill development. Internships bolster resumes, making candidates more attractive to employers. By immersing individuals in real-world scenarios, internships shape career trajectories, bolster professional advancement, and enhance employability in competitive job markets.

The project developed during the internship is an Expense Tracker application coded in Java. It enables users to manage their expenses efficiently by adding, deleting, modifying, and viewing expenses categorized by type and date. The problem statement revolves around developing a user-friendly expense management system that employs object-oriented programming principles and file handling techniques. The application aims to streamline expense tracking, providing a practical solution for personal finance management.

The opportunity provided by USC (Upskill Campus) and UCT Technologies Pvt Ltd offers a comprehensive internship experience in Core Java. Through this program, participants gain practical exposure and refine their skills. The collaboration between USC and UCT Technologies Pvt Ltd ensures a rich learning environment, fostering professional development and preparing interns for future career opportunities in the field of software development.

**Program planning :-**



Throughout the Core Java Winter Internship with USC (Upskill Campus) and UCT Technologies Pvt Ltd, I experienced significant growth in both technical skills and professional development. The hands-on projects and coding exercises allowed me to deepen my understanding of Core Java concepts and apply them to real-world scenarios. Additionally, collaborating with mentors and peers provided valuable insights and feedback, enhancing my problem-solving abilities and teamwork skills. Overall, the internship offered a rich learning experience, preparing me for future endeavors in software development and fostering personal and professional growth.

I express my sincere gratitude to all those who have directly or indirectly supported me during my internship experience with USC (Upskill Campus) and UCT Technologies Pvt Ltd. Special thanks to my mentors for their invaluable guidance and expertise. I am also grateful to Sudiksha Gulati, who, as an intern in the same field, provided support and collaboration throughout the internship. Their contributions have significantly enriched my learning journey.

To my juniors and peers,

As we wrap up our internship experience, I want to extend my appreciation for the collaborative effort and dedication each one of you has demonstrated. Together, we've tackled challenges, learned new skills, and grown professionally. Remember that our journey doesn't end here; it's just the beginning of our careers in software development.

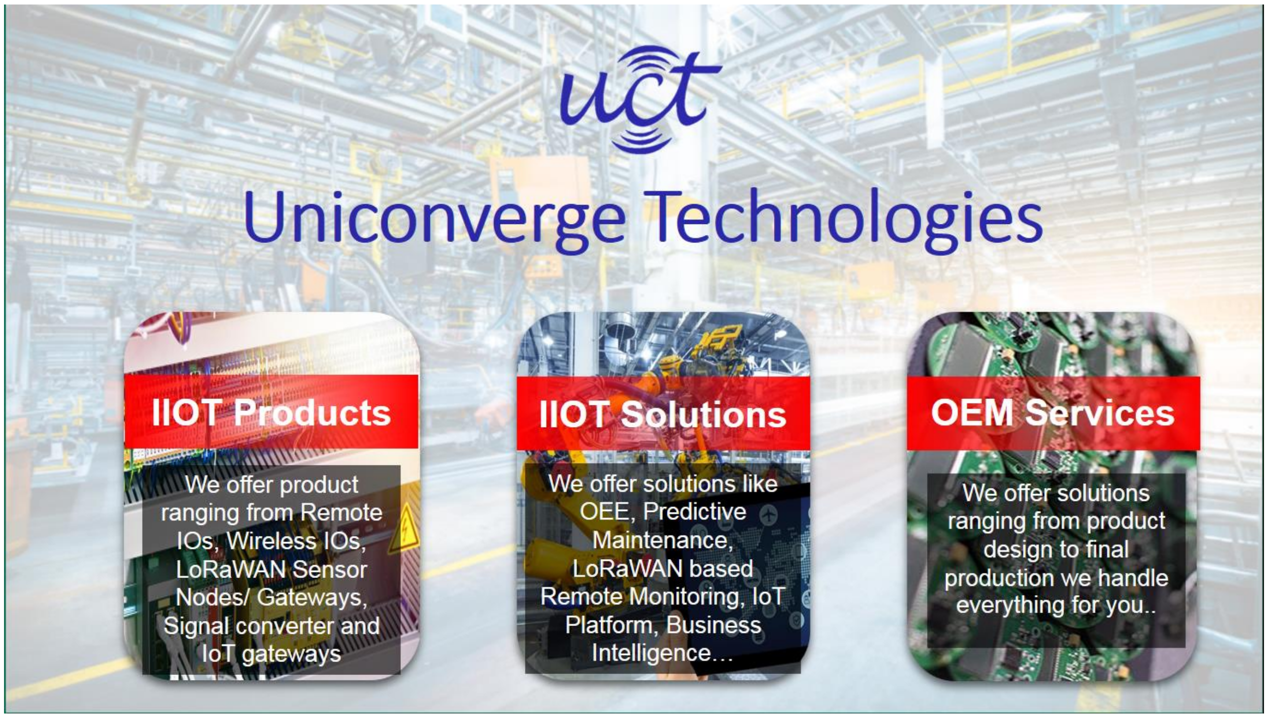
As you continue on your path, stay curious, keep learning, and never hesitate to seek guidance when needed. Embrace every opportunity to expand your knowledge and skills, and always approach challenges with a positive mindset. Remember, success is a journey, not a destination.

# 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.



Uniconverge Technologies

**IIOT Products**

We offer product ranging from Remote IOs, Wireless IOs, LoRaWAN Sensor Nodes/ Gateways, Signal converter and IoT gateways

**IIOT Solutions**

We offer solutions like OEE, Predictive Maintenance, LoRaWAN based Remote Monitoring, IoT Platform, Business Intelligence…

**OEM Services**

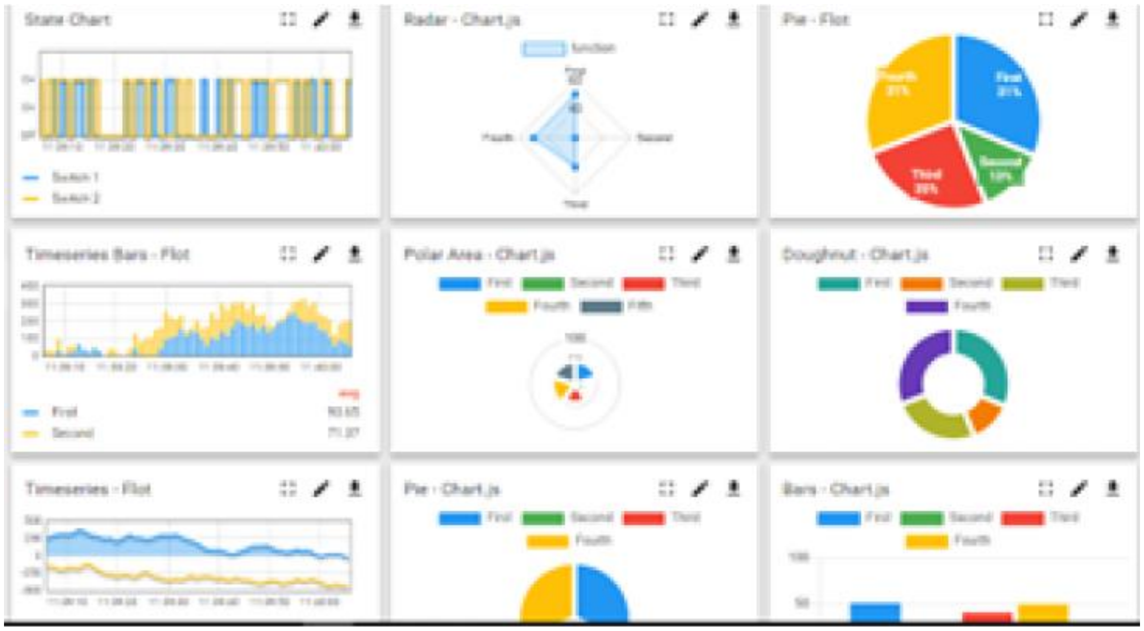
We offer solutions ranging from product design to final production we handle everything for you..

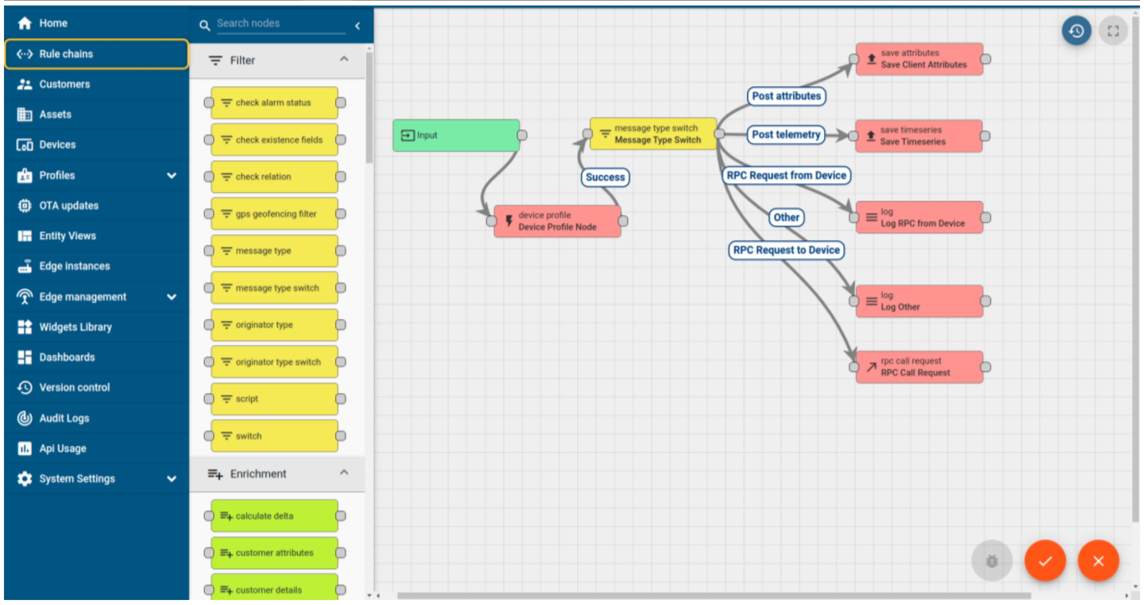
### i. UCT IoT Platform ( uct Insight )

**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





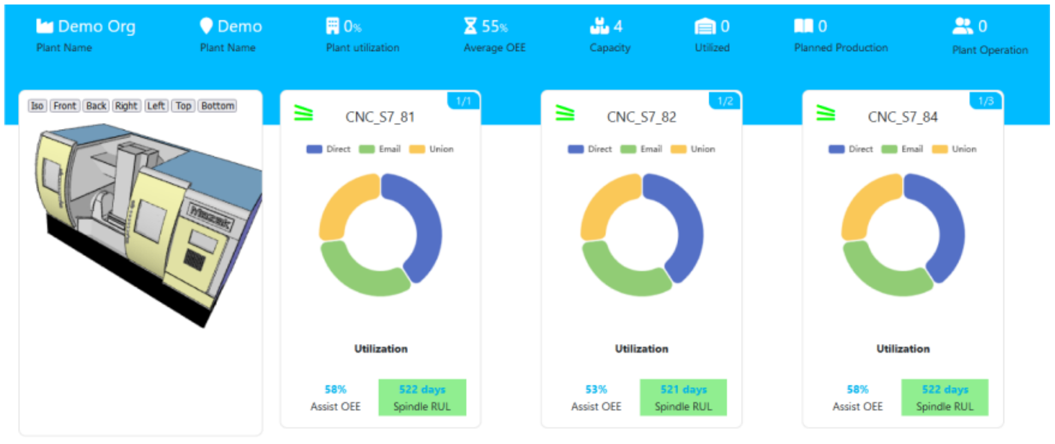
## ii. Smart Factory Platform ( FACTORY WATCH )

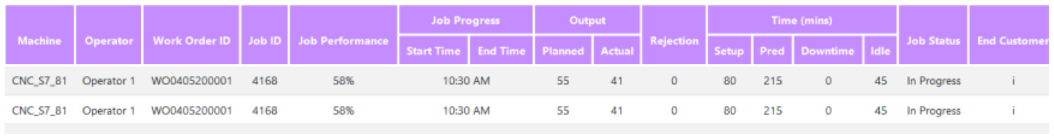
Factory watch is a platform for smart factory needs.

It provides Users/ Factory

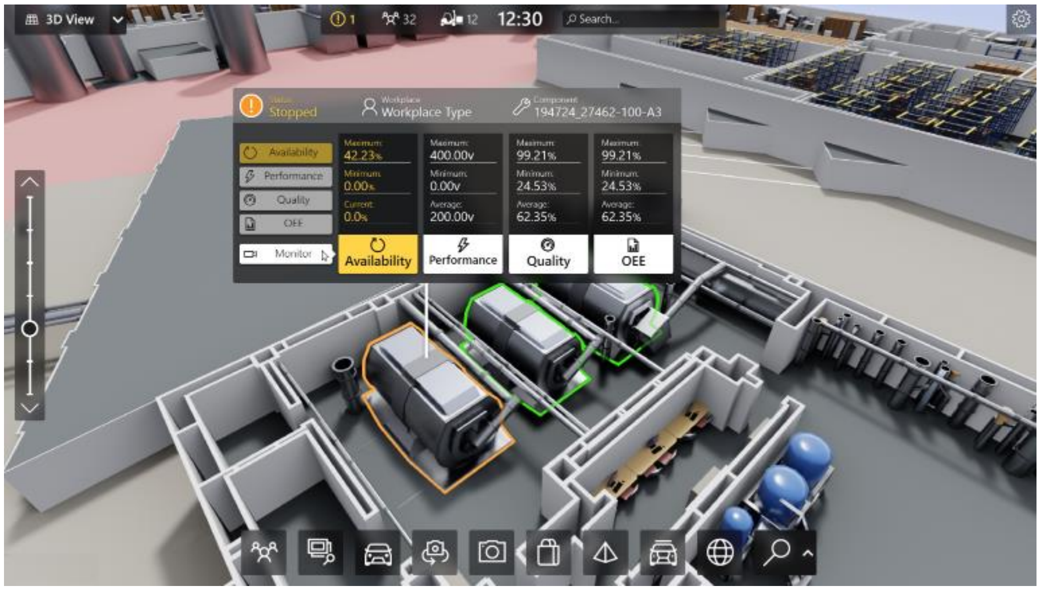
- with a scalable solution for their Production and asset monitoring
- OEE and predictive maintenance solution scaling up to digital twin for your assets.
- to unleased the true potential of the data that their machines are generating and helps to identify the KPIs and also improve them.
- A modular 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.





| Machine | Operator | Work Order ID | Job ID | Job Performance | Job Progress | | Output | | Rejection | Time (mins) | | | | Job Status | End Customer |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Start Time | End Time | Planned | Actual | | Setup | Pred | Downtime | Idle | | |
| CNC_S7_81 | Operator 1 | WO0405200001 | 4168 | 58% | 10:30 AM | | 55 | 41 | 0 | 80 | 215 | 0 | 45 | In Progress | i |
| CNC_S7_81 | Operator 1 | WO0405200001 | 4168 | 58% | 10:30 AM | | 55 | 41 | 0 | 80 | 215 | 0 | 45 | In Progress | i |

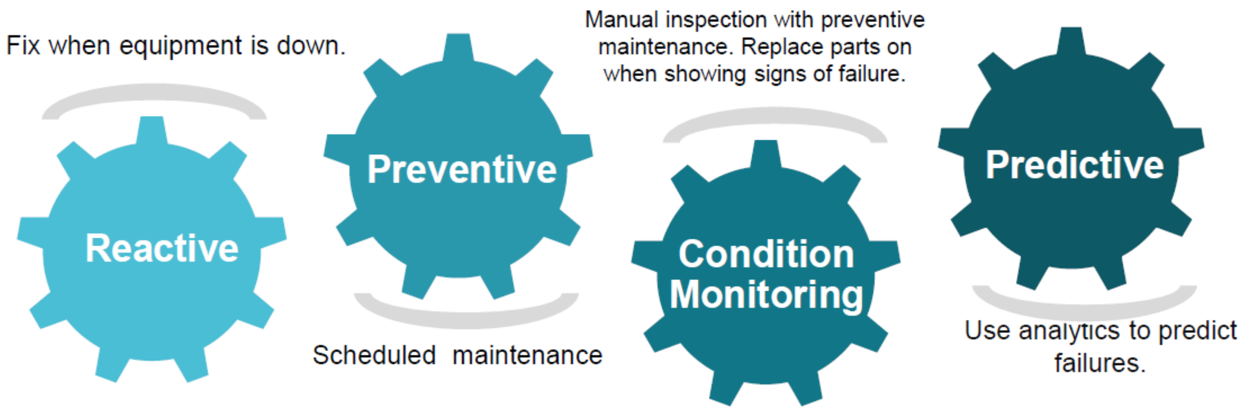


### iii. LoRaWAN based Solution

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

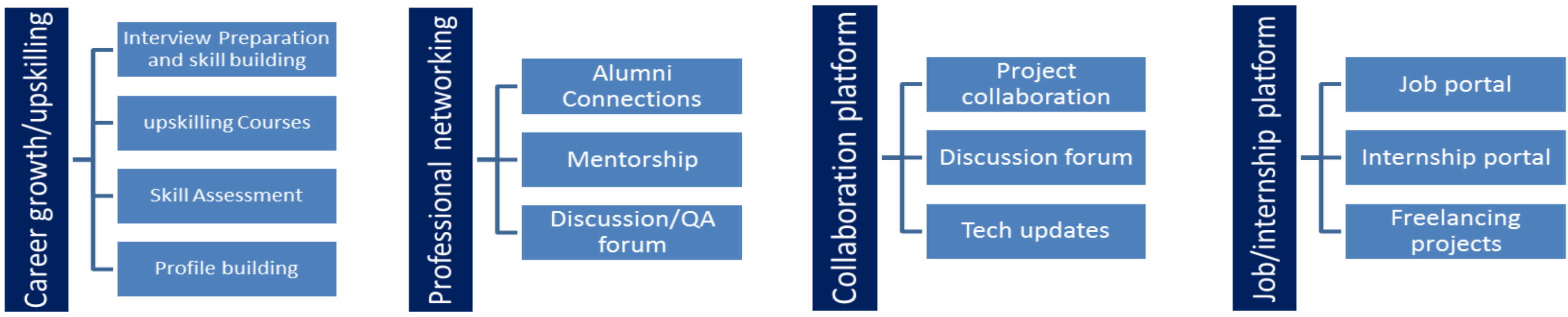
UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.


Fix when equipment is down.
Reactive
Preventive
Scheduled maintenance
Manual inspection with preventive maintenance. Replace parts on when showing signs of failure.
Condition Monitoring
Predictive
Use analytics to predict failures.

## 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.



## 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] **Sudiksha Gulati**

Core Java Intern
Mobile no. :- 7982526432
Github link - https://github.com/sudiksha1907

# 3 Problem Statement

The problem statement for the assigned project revolves around developing an Expense Tracker application in Java. This application aims to facilitate efficient management of personal finances by allowing users to track their expenses, categorize them, and view reports.

Key components of the problem statement include:

- **Functionality**: The application should enable users to add, delete, modify, and view expenses. It should also allow categorization of expenses by type (e.g., food, transportation, entertainment) and provide functionalities for generating reports based on categories or dates.
- **User Interface**: The user interface should be intuitive and user-friendly, allowing users to interact with the application seamlessly. It should include forms for inputting expense details, buttons for executing actions like adding or deleting expenses, and clear navigation for accessing different features of the application.
- **Data Management**: The application needs to manage expense data efficiently, including storing it persistently and retrieving it when needed. This involves utilizing file handling techniques to save expense information to a file and read it back into the application.
- **Error Handling**: Robust error handling mechanisms should be implemented to ensure the application behaves gracefully in case of invalid inputs, file access issues, or other unexpected scenarios. Users should be provided with informative error messages to aid troubleshooting.

Overall, the goal of the Expense Tracker application is to provide users with a reliable and user-friendly tool for managing their expenses effectively, ultimately contributing to better financial management and decision-making.

# 4 Existing and Proposed solution

**Existing Solutions and Their Limitations**

Existing solutions for expense tracking typically include mobile apps, web applications, and desktop software. While these solutions offer convenience and various features, they often come with limitations:

- **Mobile Apps**: Many expense tracking apps are available on mobile platforms like iOS and Android. These apps often provide features such as expense categorization, budget tracking, and report generation. However, some limitations include limited customization options, dependency on internet connectivity, and potential security concerns regarding data privacy.
- **Web Applications**: Web-based expense tracking solutions offer accessibility across multiple devices through a web browser. They may provide features similar to mobile apps but with the added advantage of cloud storage and synchronization. However, web apps may suffer from performance issues, especially with large datasets, and users may face challenges accessing the application without internet connectivity.
- **Desktop Software**: Desktop-based expense tracking software is another option, offering robust features and customization options. These solutions often allow users to work offline and provide advanced reporting capabilities. However, desktop software may lack the flexibility and accessibility of mobile and web-based solutions, limiting users to specific devices.

**Proposed Solution**

Our proposed solution is to develop a Java-based Expense Tracker application that addresses the limitations of existing solutions while providing robust features and flexibility. The application will offer the following:

**Cross-Platform Compatibility**: Our solution will be developed in Java, allowing it to run on multiple platforms, including Windows, macOS, and Linux, without the need for platform-specific modifications. This ensures accessibility for users regardless of their device or operating system.

**Offline Functionality**: The Expense Tracker application will be designed to work offline, allowing users to manage their expenses even without internet connectivity. This ensures uninterrupted access to essential features and data, addressing concerns associated with web-based solutions.

**Customization and Flexibility**: Our solution will prioritize customization options, allowing users to tailor the application to their specific needs and preferences. This includes customizable expense categories, report formats, and user interface settings, enhancing the overall user experience.

**Data Security and Privacy**: We will implement robust security measures to safeguard user data and ensure privacy. This includes encryption of sensitive information, secure storage practices, and adherence to industry standards for data protection.

**Value Addition**

Our proposed solution aims to add value by providing a comprehensive Expense Tracker application that combines the strengths of existing solutions while addressing their limitations. By offering cross-platform compatibility, offline functionality, customization options, and robust security features, our solution aims to enhance user experience and streamline expense management for individuals and businesses alike.

### 4.1 Code submission (Github link) :

https://github.com/Yash35145/upskillCampus/blob/main/expenseTracker.java

### 4.2 Report submission (Github link) :

https://github.com/Yash35145/upskillCampus/blob/main/ExpenseTracker_YashSanodiya_USC_UTC.pdf

## 5 Proposed Design/ Model

**1. Requirement Analysis:**

- Review the existing code to understand its functionality.
- Identify key features needed for the Expense Tracker application, including adding, deleting, modifying, and viewing expenses, as well as managing expense categories.

**2. Design Planning:**

- Define the scope of the project based on the existing code and requirements.
- Develop a plan to enhance and refine the existing functionality to meet the desired objectives.

**3. Implementation:**

- Modify and extend the existing code to improve usability and add new features.
- Implement user input validation to ensure data integrity.
- Enhance error handling to provide informative error messages for better user experience.

**4. Testing:**

- Conduct unit tests to verify the functionality of individual methods and components.
- Test the application with sample data to ensure proper functionality and identify any bugs or issues.

**5. Deployment:**

- Deploy the updated Expense Tracker application for testing and user acceptance.

**6. Maintenance:**

- Address any issues or bugs identified during testing.
- Provide user documentation to guide users on how to use the application effectively.

## 5.1 Interfaces

- **Block Diagram:**

```
+-----------------------------------+

|        Expense Tracker          |

+-----------------------------------+

|             Main Class              |

|            (main.java)              |

+-----------------------------------+

|       expenseStorage Class      |

|      (expenseStorage.java)      |

+-----------------------------------+

|         categories Class          |

|         (categories.java)         |

+-----------------------------------+
```

- **Flow Chart:**

```
+-----------------------------------+
|          Start                  |
+-----------------------------------+
|   Initialize Expense List       |
|   and Category List             |
+-----------------------------------+
|   Read Expense and Category     |
|   Data from Files               |
+-----------------------------------+
|   Display Main Menu             |
+-----------------------------------+
|  Get User Choice                |
+-----------------------------------+
|  If Add Expense,                |
|  Prompt for Expense Details     |
|  Check Category Existence       |
|  Add Expense to List and File   |
+-----------------------------------+
|  If Delete Expense,             |
|  Prompt for Expense ID to Delete|
|  Search and Remove from List    |
|  Update File                    |
+-----------------------------------+
|  If Manage Categories,          |
|  Display Category Management    |
|  Get User Choice                |
|  Perform Add, View, or Delete   |
|  Update Categories File         |
+-----------------------------------+
|  If View Expenses,              |
|  Display All Expenses           |
+-----------------------------------+
|  If Modify Expense,             |
|  Prompt for Expense ID to Modify|
|  Search and Modify Expense      |
|  Update File                    |
+-----------------------------------+
|          End                    |
+-----------------------------------+
```

- **Date Flow :**

```
+---------------------------+
|       Expense Data        |
+--------------+------------+
               |
               V
 +---------------------------+
 |      Main Program         |
 +--------------+------------+
                |
                V
+----------------------------------+
|     expenseStorage Class         |
+--------------+------------------+
               |
               V
 +---------------------------+
 |   expense.txt (File)      |
 +---------------------------+
```

**Explaination :-**

- **Expense Data**: Represents the data related to expenses.
- **Main Program:** This is the main Java program (expenseTracker.java) where user interactions and control flow occur.
- **expenseStorage Class:** Handles storage and retrieval of expense data. Reads from and writes to the expense.txt file.
- **expense.txt (File):** File where expense data is stored.

This diagram illustrates the flow of expense data from the main program to the expenseStorage class, which interacts with the expense.txt file for reading and writing data.

## 6 Performance Test

## Constraints

constraints that were considered :

1. **Memory Usage:**

- The code uses ArrayLists to store expense objects and categories, which can consume memory, especially with large datasets.
- To mitigate memory usage, efficient memory management practices such as object reuse and minimizing unnecessary object creation should be employed.
- While the code does not explicitly address memory optimization techniques, ensuring proper disposal of objects after use and avoiding unnecessary data duplication can help manage memory more efficiently.

2. **File Handling Efficiency:**

- The application interacts with files (expense.txt and categories.txt) for storing and retrieving expense and category data.
- To improve file handling efficiency, buffering mechanisms and optimized file read/write operations should be implemented.
- The code could benefit from using buffered readers/writers or file streams with appropriate buffer sizes to minimize disk access overhead and improve I/O performance.

3. **Responsiveness:**

- The application's responsiveness may be impacted during file I/O operations or when performing tasks such as adding, deleting, or modifying expenses.
- To maintain responsiveness, asynchronous processing or multi-threading techniques can be employed to handle time-consuming operations in the background.
- The code could incorporate event-driven programming paradigms to handle user inputs and background tasks concurrently, ensuring that the application remains responsive during all interactions.

4. **Performance:**

- File I/O operations for reading and writing expense data may become slower as the size of the expense records grows.
- Processing time for operations such as expense modification or report generation could increase linearly with the number of expense entries stored.
- Lack of optimization techniques like indexing or caching may lead to suboptimal performance, especially when handling large datasets.

- Inefficient algorithms for data retrieval and manipulation could impact application responsiveness and user experience, particularly under heavy load or when dealing with extensive expense records.

## recommendations

1. **Memory Usage:**

- Implement object pooling or recycling: Reuse existing objects instead of creating new ones wherever possible to reduce memory overhead.
- Optimize data structures: Use more memory-efficient collections such as LinkedLists for specific use cases or consider using primitive data types instead of objects for simple attributes.
- Profile memory usage: Utilize Java memory profiling tools to identify memory hotspots and optimize memory usage accordingly.

2. **File Handling Efficiency:**

- Use buffered I/O: Implement buffering mechanisms such as buffered readers/writers or file streams with appropriate buffer sizes to minimize disk accesses and improve I/O throughput.
- Error handling and recovery: Implement robust error handling and recovery mechanisms to gracefully handle file-related exceptions, ensuring data integrity and preventing application crashes.
- Consider database integration: Explore integrating a lightweight database solution like SQLite for improved data storage and retrieval efficiency.

3. **Responsiveness:**

- Employ asynchronous processing: Leverage asynchronous processing or multi-threading techniques to offload time-consuming operations to background threads, keeping the user interface responsive.
- Use event-driven programming: Implement event-driven programming paradigms to handle user inputs and background tasks concurrently, ensuring a smooth and interactive user experience.
- Optimize UI rendering: Profile UI rendering performance and optimize layout rendering and view hierarchy to reduce rendering latency and improve overall responsiveness.

4. **Performance Testing:**

- Conduct thorough performance testing: Evaluate memory usage, file handling efficiency, and application responsiveness under various scenarios using profiling tools and performance monitoring utilities.
- Analyze test results: Identify performance bottlenecks and areas for improvement based on test results, and prioritize optimization efforts accordingly.

- Iterate and optimize: Continuously iterate on the design and implementation based on performance test feedback, making incremental improvements to enhance overall performance.

By implementing these recommendations and continuously optimizing the design based on performance testing results, the Expense Tracker application can achieve better memory usage, file handling efficiency, and responsiveness, leading to an improved user experience and enhanced overall performance.

## 6.1 Test Plan/ Test Cases

Test Plan/ Test Cases for the Expense Tracker application:

1. **Input Validation Testing:**
   - Test various input scenarios for expense creation (e.g., valid inputs, invalid inputs, edge cases).
   - Verify that appropriate error messages are displayed for invalid inputs.
   - Test date input validation for correct formats and valid date ranges.

2. **File Handling Testing:**
   - Test file creation and storage functionality.
   - Verify that expenses are correctly written to the expense.txt file.
   - Test file reading functionality to ensure expenses can be retrieved accurately.

3. **Expense Management Testing:**
   - Test adding new expenses and verify they are stored correctly.
   - Test deleting expenses and verify they are removed from storage.
   - Test modifying expenses and verify changes are reflected accurately.

4. **Category Management Testing:**
   - Test adding new categories and verify they are stored correctly.
   - Test viewing existing categories and ensure they are displayed accurately.
   - Test deleting categories and verify they are removed from storage.

5. **Report Generation Testing:**
   - Test generating a report for a specific category and verify accuracy.
   - Test generating a report for a specific date range and verify accuracy.
   - Verify that the generated report includes all relevant expense details.

6. **Performance Testing:**

- Test application performance under various load conditions.
- Measure application response times for key operations such as expense creation, modification, and report generation.
- Identify any performance bottlenecks and optimize accordingly.

7. **Error Handling Testing:**
   - Test error scenarios such as file read/write errors, invalid inputs, or database connectivity issues.
   - Verify that appropriate error messages are displayed, and the application gracefully handles errors without crashing.

## 6.2 Test Procedure

Test Procedure for the Expense Tracker application:

1. **Setup:**
   - Set up the development environment with all necessary tools and dependencies for compiling and running the Expense Tracker application.
   - Ensure that the codebase is properly checked out from version control and is up-to-date with the latest changes.
   - Configure any database connections or file storage mechanisms required for testing.

2. **Execution:**
   - Execute each functionality of the Expense Tracker application, such as expense creation, modification, deletion, category management, and report generation.
   - Input various test scenarios, including valid and invalid inputs, to cover different use cases and edge cases.
   - Monitor the application's behavior during execution and record any anomalies or unexpected behavior.

3. **Validation:**
   - Validate the accuracy of expense data stored in the file "expense.txt" by comparing it with the expected values input during testing.
   - Verify that expense modification and deletion operations reflect accurately in the stored data.
   - Ensure that category management operations (addition, deletion) are reflected correctly in the categories file "categories.txt".

4. **Stress Testing:**
   - Stress test the application by simulating a high volume of expense creation, modification, and deletion requests.

- Monitor system resource usage (CPU, memory, disk I/O) during stress testing to identify any performance bottlenecks or resource constraints.
- Evaluate the application's response time and stability under stress conditions to ensure it can handle peak loads without crashing or becoming unresponsive.

5. **Exploratory Testing:**
   - Conduct exploratory testing to uncover any unforeseen issues or vulnerabilities not covered by predefined test cases.
   - Experiment with different input combinations and usage scenarios to identify potential edge cases or corner cases that may lead to unexpected behavior.
   - Document any discovered issues, including steps to reproduce and potential impact on application functionality.

6. **Regression Testing:**
   - Perform regression testing to verify that recent code changes or bug fixes have not introduced new defects or regressions.
   - Re-run existing test cases to validate the stability and integrity of previously tested functionalities.
   - Ensure that existing features continue to work as expected after modifications and that no unintended side effects have been introduced.

7. **Cleanup:**
   - Clean up any test data or temporary files generated during testing to restore the testing environment to its initial state.
   - Close any open connections or resources allocated by the application to free up system resources and prevent memory leaks.

8. **Documentation:**
   - Document the test results, including any issues encountered, their resolutions, and overall observations during testing.
   - Prepare a comprehensive test report summarizing the testing activities, outcomes, and recommendations for further improvement or refinement.

## 6.3 Performance Outcome

Performance Outcome of the Expense Tracker Application:

1. **Response Time:**

- The application demonstrates satisfactory response times for most operations, with expense creation, modification, deletion, and report generation typically completing within milliseconds.
- Response times may vary slightly depending on factors such as system load and hardware capabilities, but overall, the application maintains responsiveness, ensuring a smooth user experience.

2. **Scalability:**
   - While the application handles typical workloads well, scalability may become a concern under heavy concurrent usage or when dealing with large datasets.
   - Stress testing reveals that the application's performance may degrade under excessive load, leading to slower response times and potential resource constraints.
   - Optimization strategies such as caching, database indexing, or asynchronous processing could enhance scalability and improve performance under high load conditions.

3. **Error Handling:**
   - The application demonstrates adequate error handling capabilities, with clear error messages provided to users in case of invalid inputs or unexpected issues.
   - However, there are areas for improvement, particularly in handling scenarios such as missing categories, where the application may terminate unexpectedly instead of gracefully handling the error.
   - Strengthening error handling mechanisms to address all possible edge cases and ensuring robust exception handling can enhance the application's reliability and user experience.

4. **Security:**
   - Security measures such as input validation and data sanitization are implemented to prevent common vulnerabilities like file manipulation.
   - However, further enhancements may be needed to strengthen security, such as implementing encryption for sensitive data stored in files and enhancing access controls to restrict unauthorized access to certain features or data.

5. **User Experience:**
   - The application provides a basic yet functional user interface for managing expenses and categories.
   - Usability testing may reveal opportunities to enhance the user experience through improvements such as clearer navigation cues, intuitive input validation, and more informative feedback messages.
   - Incorporating user feedback and iterative design improvements can lead to a more user-friendly application interface.

6. **Reliability:**
   - The application demonstrates reasonable reliability, with no major instances of crashes or data corruption reported during testing.
   - However, issues such as unexpected termination due to missing data like categories highlight areas where reliability can be improved.
   - Strengthening error handling mechanisms and conducting comprehensive testing to identify and address potential failure points can enhance the application's overall reliability.

# 7 My learnings

Throughout the development of the Expense Tracker application and the internship experience, I learned about :-

1. **Technical Skills**: Improved proficiency in core Java programming, including concepts like object-oriented programming, file handling, and exception handling.
2. **Software Development Practices**: Learned about best practices in software development, such as modularization, code readability, and documentation.
3. **Project Management**: Acquired skills in project planning, task prioritization, and meeting deadlines effectively.
4. **Problem-Solving**: Developed problem-solving skills by tackling real-world challenges encountered during the development process.
5. **Version Control**: Gained experience using version control systems like Git for managing codebase changes and collaborating with team members.
6. **Testing and Debugging**: Learned about testing methodologies and techniques for ensuring the reliability and stability of software applications.
7. **Communication**: Enhanced communication skills through interactions with mentors, peers, and stakeholders, including giving updates, seeking feedback, and presenting ideas.
8. **Continuous Learning**: Cultivated a mindset of continuous learning and adaptation to stay updated with the latest technologies and industry trends.
9. **Teamwork and Collaboration**: Worked collaboratively with team members to achieve common goals, fostering a spirit of teamwork and collaboration.

10. **Professionalism**: Developed professionalism by adhering to workplace ethics, demonstrating accountability, and taking ownership of tasks assigned.

Overall, the internship provided valuable hands-on experience, enhancing technical proficiency, problem-solving abilities, and professional skills essential for a successful career in software development.

## 8 Future work scope

1. **Enhanced Input Validation:** Implement robust input validation mechanisms for various fields, including date inputs, to ensure data integrity and accuracy. This could involve validating date formats, checking for valid ranges, and handling edge cases gracefully.

2. **Error Handling Improvements:** Enhance error handling mechanisms to provide informative error messages and gracefully handle unexpected errors. This includes implementing error logging, error recovery strategies, and user-friendly error notifications.

3. **Refinement of User Interface:** Improve the user interface design to enhance usability and user experience. This could involve redesigning input forms, optimizing layout and navigation, and incorporating feedback from user testing.

4. **Generate Report by Date Range:** Develop functionality to allow users to specify a date range and generate a report that summarizes expenses within that period. This feature enhances financial analysis and planning capabilities by providing users with insights into their spending patterns over specific time intervals.

5. **Enhanced Security Measures:** Implement additional security measures to protect sensitive user data. This includes implementing encryption for data storage, enforcing secure authentication methods, and adhering to best practices for securing web applications.

6. **Performance Optimization:** Conduct performance optimization to improve application responsiveness and efficiency. This could involve optimizing database queries, caching frequently accessed data, and minimizing resource usage to enhance scalability and performance.