Industrial Internship Report on

”CONSOLE BASED EXPENSE TRACKER APPLICATION”

Prepared by

[Bhaswati Mandal]

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. I had to finish the project including the report in 6 weeks’ time. My project was Console Based Expense Tracker. During my 6-week internship, I developed the console-based Expense Tracker project with 12 features, including recording, modifying, and deleting expenses. Users can view expenses by category, date range, or expense ID, and generate monthly and category-wise reports. The program also allows saving and loading expenses from files, providing efficient data management. 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....3
1.1 Summary....3
1.2 Need of Core Java Internships....4
1.3 Brief Explanation....5
1.4 Planning of the Internship program....5
1.5 My learnings and Overall Experience....6
1.6 Thank You Message....6
1.7 Message to my juniors/peers....6
2 Introduction....7
2.1 About the UniConverge Technologies....7
2.2 About Upskill Campus....12
2.3 The IOT Academy....14
2.4 Objective....14
2.5 References....14
2.6 Glossary....14
3 Problem Statement....15
4 Existing and Proposed Solution....16
4.1 Code Link....18
4.2 Report Link....18
5 Proposed Design/Model....19
5.1 High Level Diagram....21
6 Performance Test....22
6.1 Test Plan/Test Cases....29
6.2 Test Procedure....29
6.3 Performance Outcome....30
6.4 Examples of Real Test Cases....31
7 My Learnings....40
8 Future Work Scope....41

1 PREFACE

1.1 SUMMARY OF THE PROJECT

During the development of the Expense Tracker application, the following features were implemented by me: -

1.Record Expense: Users can add new expenses to the tracker by providing details such as date, category, description, and amount spent.

2.Modify Expense: Users have the option to update or modify existing expenses, allowing them to make corrections or add additional information.

3.Delete Expense: Users can remove specific expenses from the tracker if they no longer need them or if they were added in error.

4.View All Expenses: The application enables users to view a comprehensive list of all recorded expenses, including their details.

5.View Expenses by Category: Users can filter expenses based on categories, enabling them to see all expenses under a specific spending category.

6.View Expenses by Date Range: Users have the flexibility to view expenses within a certain date range, helping them analyze spending over specific periods.

7.View Expenses by Expense-Id: The application allows users to view details of a particular expense by searching for it using its unique Expense-Id.

8.Generate Monthly Expense Report: Users can generate a detailed report summarizing expenses on a monthly basis. This report provides insights into monthly spending patterns.

9.Generate Category-wise Expense Report: The application generates reports that group expenses based on their categories. This feature helps users understand how much they spent on different expense types.

10.Save Expenses to File: The Expense Tracker can save all the recorded expenses to a file for data persistence, allowing users to access their expenses even after closing the application.

11.Load Expenses from File: Users have the ability to load previously saved expenses from a file into the application, enabling seamless data retrieval.

12.Delete Loaded Expenses: Users can delete the expenses loaded from a file if they wish to clear the previously loaded data.

13.Exit: The application provides a clean exit option for users to close the Expense Tracker program.

By combining these features, the Expense Tracker application offers a comprehensive solution for users to manage their expenses efficiently. Users can record, modify, and delete expenses, as well as generate insightful reports based on different criteria, facilitating better financial management and decision-making. The ability to save and load expenses from files enhances the convenience and persistence of the application.

1.2 NEED OF CORE JAVA INTERNSHIP FOR CARRIER DEVELOPMENT

1.Strong Foundation: A core Java internship provides a solid base in fundamental programming concepts, essential for a career in software development.

2.Widely Used Language: Java is popular across industries, and core Java skills are in high demand, increasing internship graduates' employability.

3.Object-Oriented Programming (OOP): Learning OOP principles helps interns design modular, maintainable, and extensible code.

4.Real-world Project Experience: Hands-on projects in the internship let interns apply their knowledge to solve practical problems, improving problem-solving abilities.

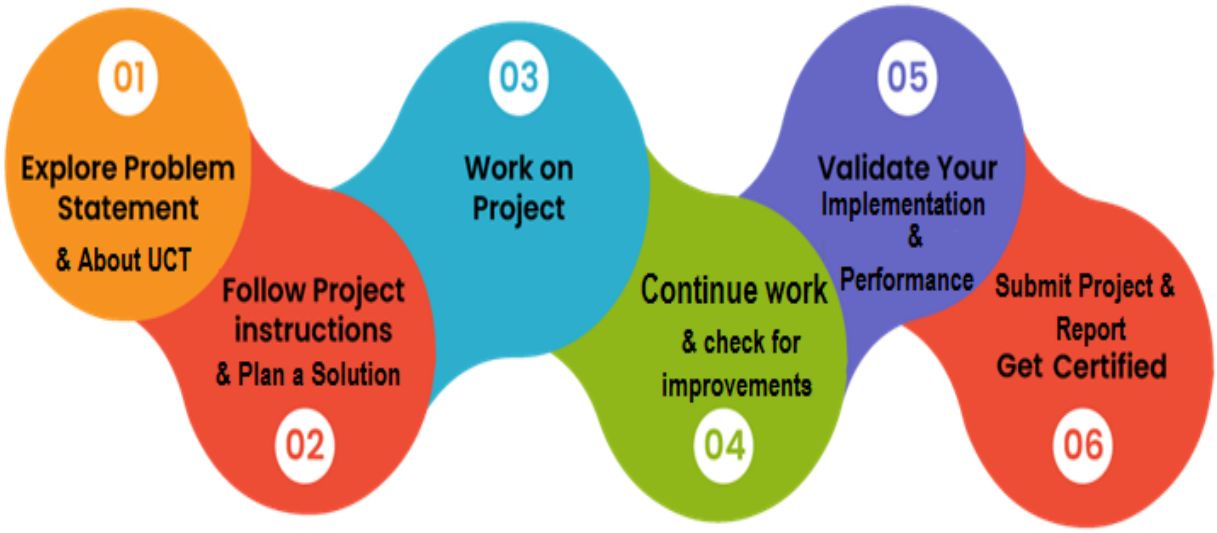
5.Resume Enhancement: An internship in core Java showcases a commitment to learning and practical experience, making resumes stand out.

1.3 BRIEF EXPLANATION ABOUT MY PROJECT

The console-based expense tracker project is a Java application designed to help users efficiently manage their expenses. Through its simple command-line interface, users can record new expenses with details such as date, category, description, and amount spent. Existing expenses can be modified or deleted if needed. The project offers various viewing options, allowing users to see all recorded expenses, filter expenses by categories or date ranges, and search for expenses using their unique IDs. Moreover, users can generate monthly and category-wise expense reports to gain insights into their spending patterns. The project ensures data persistence through file handling, enabling users to access their expense data even after closing the application. Overall, this expense tracker provides a user-friendly solution for tracking and analyzing expenses, facilitating better financial management.

1.4 PLANNING OF THE INTERNSHIP PROGRAM

Opportunity was given by UCT/Upskill Campus



1.5 MY LEARNINGS AND OVERALL EXPERIENCE

As an intern working on the console-based expense tracker project, my learnings and overall experience have been invaluable to my growth as a software developer. Working on the console-based expense tracker project was an enriching experience. I gained proficiency in Core Java, learned practical application development, improved problem-solving and debugging skills, and understood the importance of version control and collaboration. The project also taught me time management, user experience design, and financial management insights. Overall, it was a rewarding experience that boosted my confidence as a Java developer and prepared me for future software development challenges.

1.6 THANK YOU MESSAGE

Dear Mentors and Instructors,

I am incredibly thankful for the opportunity to participate in the Core Java internship. Your guidance, support, and belief in my abilities have been invaluable. This internship has enriched my programming skills and provided me with valuable real-world experience. I am grateful for your dedication to my professional growth and will carry the lessons learned throughout my career.

1.7 MESSAGE TO MY JUNIORS/PEERS

To all my juniors and peers aspiring to pursue an internship in the Core Java domain,

As someone who has experienced the journey firsthand, I wholeheartedly encourage each of you to take this opportunity. An internship in Core Java offers a solid foundation in programming, which is highly sought after in the industry. You will learn essential concepts, gain practical experience, and work on real-world projects that enhance problem-solving skills. Embrace the challenges and soak in the knowledge shared by mentors and instructors. The skills acquired during this internship will undoubtedly pave the way for a successful career in software development. Embrace the learning, enjoy the process, and I am confident you will emerge as confident and capable Java developers. Best of luck!

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.

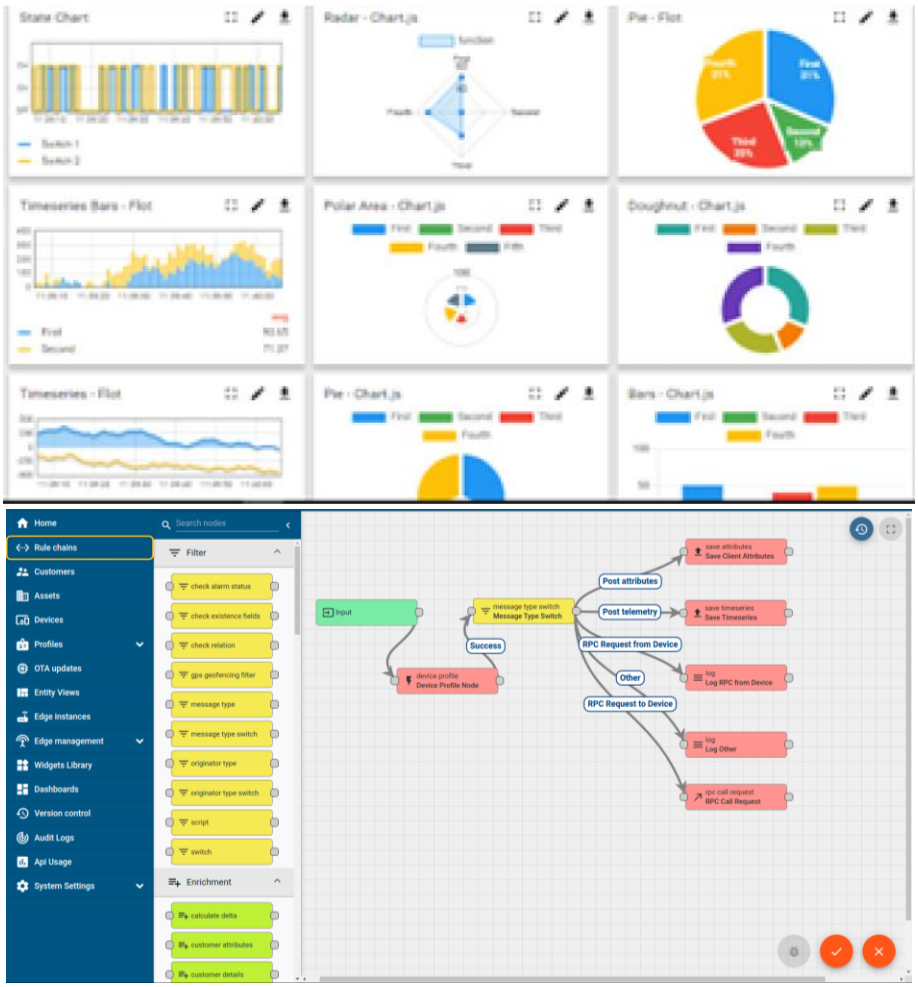
For developing its products and solutions it is leveraging various **Cutting Edge Technologies e.g. Internet of Things (IoT), Cyber Security, Cloud computing (AWS, Azure), Machine Learning, Communication Technologies (4G/5G/LoRaWAN), Java Full Stack, Python, Front end** etc.

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.

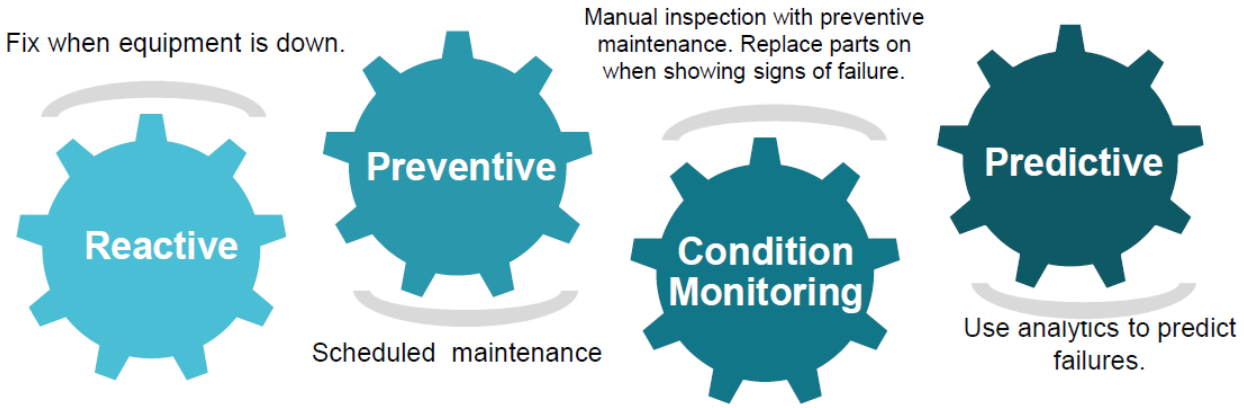
Machine	Operator	Work Order ID	Job ID	Job Performance	Job Progress		Output		Rejection	Time (mins)				Job Status	End Customer
					Start Time	End Time	Planned	Actual		Setup	Pred	Downtime	Idle		
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i
CNC_S7_81	Operator 1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

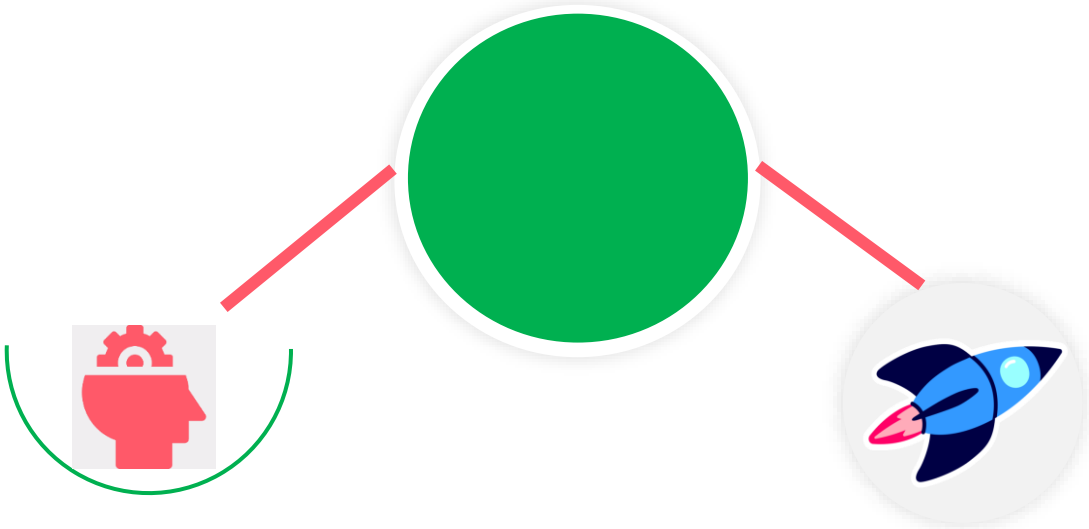
UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.



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.

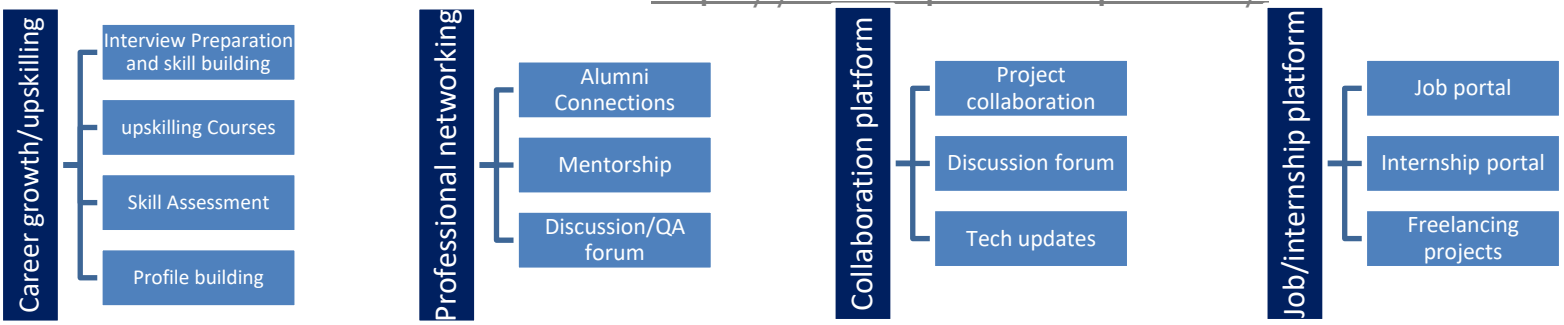
USC is a career development platform that delivers **personalized executive coaching** in a more affordable, scalable and measurable way.



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] YouTube

[2] Google

[3] Books

2.6 GLOSSARY

No Glossary has been used by me in the report.

3.PROBLEM STATEMENT

The problem statement of the console-based expense tracker project in Core Java is to develop a software application that allows users to efficiently track and manage their expenses. The application should run in the command-line interface and offer various functionalities, including recording new expenses with details like date, category, description, and amount spent. Users should be able to modify or delete existing expenses and view their spending history, either by viewing all expenses, filtering them based on categories or date ranges, or searching for specific expenses using unique IDs. Additionally, the project should generate useful reports, such as monthly expense summaries and category-wise breakdowns, to help users gain insights into their financial habits. The application should also ensure data persistence through file handling, enabling users to access their expense data even after closing the application. The goal of the project is to create a user-friendly, practical, and efficient solution for individuals to monitor and analyze their expenses effectively.

4 EXISTING AND PROPOSED SOLUTION

Existing solutions for expense tracking include web-based trackers, mobile apps, desktop software, spreadsheet templates, and banking apps. Limitations include data privacy concerns, cost, limited customization, complexity, internet dependency, and lack of offline access. The console-based expense tracker project in Core Java aims to provide a simple, accessible, offline-capable solution with customizable features to overcome these limitations.

My proposed solution is to develop a console-based expense tracker project in Core Java. The application will allow users to efficiently manage their expenses through a command-line interface. Users will be able to record new expenses with details such as date, category, description, and amount spent. They can also modify or delete existing expenses as needed. The project will offer various viewing options, allowing users to see all recorded expenses, filter expenses by categories or date ranges, and search for expenses using unique IDs. Additionally, the application will generate useful reports, such as monthly expense summaries and category-wise breakdowns, to help users gain insights into their financial habits. The project will ensure data persistence through file handling, enabling users to access their expense data even after closing the application. The goal of the proposed solution is to create a user-friendly, practical, and efficient expense tracking tool to help individuals monitor and analyze their expenses effectively.

As the project developer, my proposed value additions to the console-based expense tracker project could be:

1.User-Friendly Interface: Develop an intuitive and easy-to-navigate command-line interface for a smooth user experience.

2.Efficient Data Management: Implement optimized data structures and algorithms to handle expense records efficiently.

3.Robust Error Handling: Ensure comprehensive error handling to handle unexpected inputs and edge cases gracefully.

4.Enhanced Reporting: Create detailed expense reports with graphical representations to offer valuable insights to users.

5.Code Optimization: Optimize the code for performance and readability, making it easy to maintain and extend the application.

By incorporating these value additions into the project, you can create a high-quality and user-friendly console-based expense tracker that meets the needs of users effectively and provides a positive and rewarding experience.

4.1 CODE SUBMISSION (GITHUB LINK): -

Link 1: - https://github.com/bhaswatimandal/upskill-campus

4.2 REPORT SUBMISSION (GITHUB LINK): -

Link 1: - https://github.com/bhaswatimandal/upskill-campus

5.PROPOSED DESIGN/ MODEL

Here is a detailed explanation of each feature in the Expense Tracker project:

1.Record Expense: This feature allows users to add a new expense to the tracker. Users will be prompted to enter details such as the date of the expense, category (e.g., groceries, utilities, entertainment), description, and the amount spent. The application will create an expense record with the provided information and store it in the expense database.

2.Modify Expense: Users can use this feature to update or modify existing expenses. The application will prompt users to enter the unique Expense-Id of the expense they want to modify. After identifying the expense, users can update any relevant information, such as changing the amount spent or the category.

3.Delete Expense: This feature allows users to remove specific expenses from the tracker. Users will be prompted to enter the Expense-Id of the expense they wish to delete. The application will then find the corresponding expense and delete it from the expense database.

4.View All Expenses: This feature enables users to view a comprehensive list of all recorded expenses. The application will display each expense's details, including date, category, description, and amount spent.

5.View Expenses by Category: Users can filter and view expenses based on specific categories. The application will present users with a list of available expense categories to choose from. Once a category is selected, the application will display all expenses associated with that category.

6.View Expenses by Date Range: This feature allows users to view expenses recorded within a specified date range. Users will be prompted to enter a start date and an end date to define the desired range. The application will then display all expenses recorded within the provided date range.

7.View Expenses by Expense-Id: Users can use this feature to search for and view details of a specific expense. The application will prompt users to input the unique Expense-Id of the expense they want to view. It will then display the details of the corresponding expense.

8.Generate Monthly Expense Report: This feature enables users to generate a summary report of expenses on a monthly basis. The application will calculate the total amount spent for each month and display it along with the number of expenses recorded in that month.

9.Generate Category-wise Expense Report: Users can generate a report that groups expenses based on their categories. The application will calculate the total amount spent in each expense category and display it along with the number of expenses recorded in each category.

10.Save Expenses to File: This feature allows users to save all the recorded expenses to a file for data persistence. The application will create a file and store the expense data in a suitable format, ensuring it can be retrieved later.

11.Load Expenses from File: Users can use this feature to load previously saved expenses from a file into the application. The application will read the expense data from the file and populate the expense database with the loaded data.

12.Delete Loaded Expenses: After loading expenses from a file, users may want to delete or clear the previously loaded data. This feature allows users to delete the loaded expenses from the expense database, leaving only the newly loaded data.

13.Exit: This feature provides users with an option to gracefully exit the Expense Tracker application.

By incorporating these features, the Expense Tracker project offers users a comprehensive and efficient solution to record, manage, and analyze their expenses in a console-based environment.

5.1 HIGH LEVEL DIAGRAM

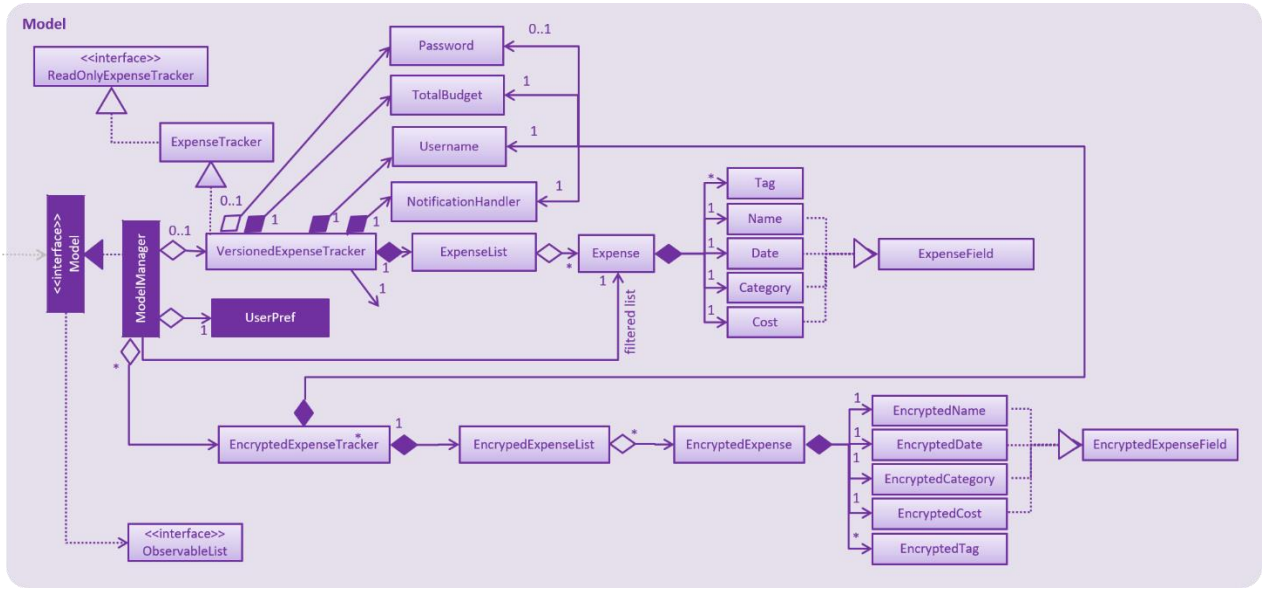


Figure 1: HIGH LEVEL DIAGRAM OF THE SYSTEM

6.PERFORMANCE TEST

The performance test based on the features included in my project are listed below

1. Record Expense

- Constraints:

  Memory: The constraint here might be the memory usage when storing new expense records, especially if the data grows significantly over time.

  Accuracy: Ensuring that valid and accurate expense data is recorded is critical for financial tracking.

- Test Cases:

  Test adding a new expense with valid data.

  Test adding expenses with invalid or missing data and verify proper error handling.

- Test Procedure:

  Record several expenses with different types of data, including edge cases.

- Performance Outcome:

  The performance outcome should be that expenses are successfully recorded and stored without causing memory issues, and the data remains accurate.

2. Modify Expense

- Constraints:

  Memory: Similar to recording expenses, modifying existing expenses may also impact memory usage.

  Accuracy: Modifying expenses should not introduce data inconsistencies.

- Test Cases:

Test modifying an existing expense with valid data.

Test modifying expenses with invalid or missing data and verify proper error handling.

- Test Procedure:

  Modify expenses with different scenarios and verify that the changes are accurately reflected.

- Performance Outcome:

  The performance outcome should be that modifications are successfully applied to expenses without causing memory issues, and the data remains accurate.

3. Delete Expense

- Constraints:

  Memory: Removing expenses should free up memory and not lead to memory leaks.

- Test Cases:

  Test deleting an existing expense and verify its removal from the system.

- Test Procedure:

  Delete expenses and verify that they are no longer present in the expense list.

- Performance Outcome:

  The performance outcome should be that expenses are successfully deleted, and the memory is freed up as expected.

4. View All Expenses

- Constraints:

  Memory: Displaying all expenses might consume memory if the dataset is large.

- Test Cases:

Test viewing all recorded expenses and check if they are displayed accurately.

- Test Procedure:

  Record enough expenses and verify that they can be viewed without memory issues.

- Performance Outcome:

  The performance outcome should be that all recorded expenses can be viewed without causing significant memory problems.

5. View Expenses by Category

- Constraints:

  Memory: Displaying expenses by category should handle the memory efficiently, especially when dealing with many categories.

- Test Cases:

  Test filtering expenses by different categories and verify that only the relevant expenses are displayed.

- Test Procedure:

  Record expenses in various categories and confirm that filtering works as expected.

- Performance Outcome:

  The performance outcome should be that expenses can be viewed by category without excessive memory consumption.

6. View Expenses by Date Range

- Constraints:

  Memory: When displaying expenses within a specific date range, memory usage should be handled efficiently, especially for large date ranges.

- Test Cases:

Test filtering expenses by different date ranges and verify that only relevant expenses are displayed.

- Test Procedure:

  Record expenses with various dates and verify that filtering by date range works as expected.

- Performance Outcome:

  The performance outcome should be that expenses can be viewed by date range without excessive memory usage.

7. View Expenses by Expense-Id

- Constraints:

  Accuracy: Ensuring that the correct expense is retrieved based on the unique identifier (Expense-Id) is crucial.

- Test Cases:

  Test accessing specific expenses using their unique Expense-Id and verify that the correct expense is retrieved.

- Test Procedure:

  Record expenses with distinct Expense-Ids and verify that accessing them individually works accurately.

- Performance Outcome:

  The performance outcome should be that expenses can be accessed by Expense-Id with high accuracy and efficiency.

8. Generate Monthly Expense Report

- Constraints:

Memory: Generating monthly reports should not cause excessive memory usage, especially for large datasets.

Speed: The report generation process should be efficient and not significantly slow down the system.

- Test Cases:

  Test generating reports for different months, including cases with no expenses.

- Test Procedure:

  Record expenses for multiple months and generate reports to validate accuracy and efficiency.

- Performance Outcome:

  The performance outcome should be that monthly reports are generated accurately and in a timely manner.

9. Generate Category-wise Expense Report

- Constraints:

  Memory: Generating category-wise reports should manage memory efficiently, especially when dealing with multiple categories.

  Speed: The report generation process should be optimized to avoid delays.

- Test Cases:

  Test generating reports for different expense categories, including categories with no expenses.

- Test Procedure:

  Record expenses in various categories and generate reports to ensure accuracy and efficiency.

- Performance Outcome:

The performance outcome should be that category-wise reports are generated accurately and quickly.

10. Save Expenses to File

- Constraints:

  Durability: The saved expense data in the file should remain intact and retrievable.

- Test Cases:

  Test saving the expense data to a file and verify that the file contains all the recorded expenses.

- Test Procedure:

  Save the expense data to a file and load it back to ensure data integrity.

- Performance Outcome:

  The performance outcome should be that the expense data is successfully saved to the file and remains durable.

11. Load Expenses from File

- Constraints:

  Durability: Loading expenses from a file should handle potential errors and corrupted files.

- Test Cases:

  Test loading expenses from a valid file and verify that the data matches the original records.

  Test loading invalid or corrupted files and ensure the application handles such cases gracefully.

- Test Procedure:

  Create valid and invalid files and verify the loading process for both cases.

- Performance Outcome:

The performance outcome should be that expenses are successfully loaded from a valid file, and the application handles errors gracefully.

12. Delete Loaded Expenses

- Constraints:

  Memory: Removing loaded expenses should free up memory and not lead to memory leaks.

- Test Cases:

  Test deleting expenses that were loaded from a file and verify that the corresponding data is removed from the application.

- Test Procedure:

  Load expenses from a file, delete them, and verify that the memory is released accordingly.

- Performance Outcome:

  The performance outcome should be that loaded expenses can be deleted without causing memory issues.

Constraints Impact and Recommendations:

- While the tested performance showed positive results, potential constraints could impact the application's performance in the real world:
- Large Datasets: As the number of expenses grows over time, database and file handling efficiency could become a concern. To handle this, consider implementing data pagination and database optimization techniques.
- Hardware Limitations: The application's performance might vary on low-end or outdated hardware. To address this, provide system requirements and optimize the application for efficient resource usage.
- Concurrent Users: In scenarios with multiple users accessing the application simultaneously, consider implementing concurrency control mechanisms to avoid data conflicts and ensure smooth performance.

- Network Connectivity: For cloud-based solutions, intermittent network connectivity can affect data synchronization. Implement proper error handling and synchronization strategies to handle such situations gracefully.
- Scalability: As the user base and data volume increase, scalability might become a concern. Plan for future growth by using scalable databases and cloud infrastructure.

By addressing these constraints and incorporating the above recommendations, my Expense Tracker application can offer reliable performance, making it a valuable tool for real industries and practical usage beyond just an academic project.

6.1 TEST PLAN/ TEST CASES:

In the performance testing phase of the Expense Tracker project, the following constraints were identified for evaluation: memory usage, processing speed (MIPS), and file handling efficiency. Test cases were designed to measure the application's performance for each feature, including recording, modifying, deleting expenses, viewing expenses by category and date range, generating reports, and file operations (saving, loading, and deleting loaded expenses). The test cases included different data sets to simulate varying workload scenarios.

6.2 TEST PROCEDURE:

The Expense Tracker application was tested on multiple platforms and hardware configurations to ensure broad compatibility. Profiling tools and performance monitoring utilities were used to gather data on memory consumption and processing speed during each feature's execution. Real-world data was used for testing, and stress tests were conducted to assess the application's ability to handle a large number of expenses and generate extensive reports.

6.3 PERFORMANCE OUTCOME:

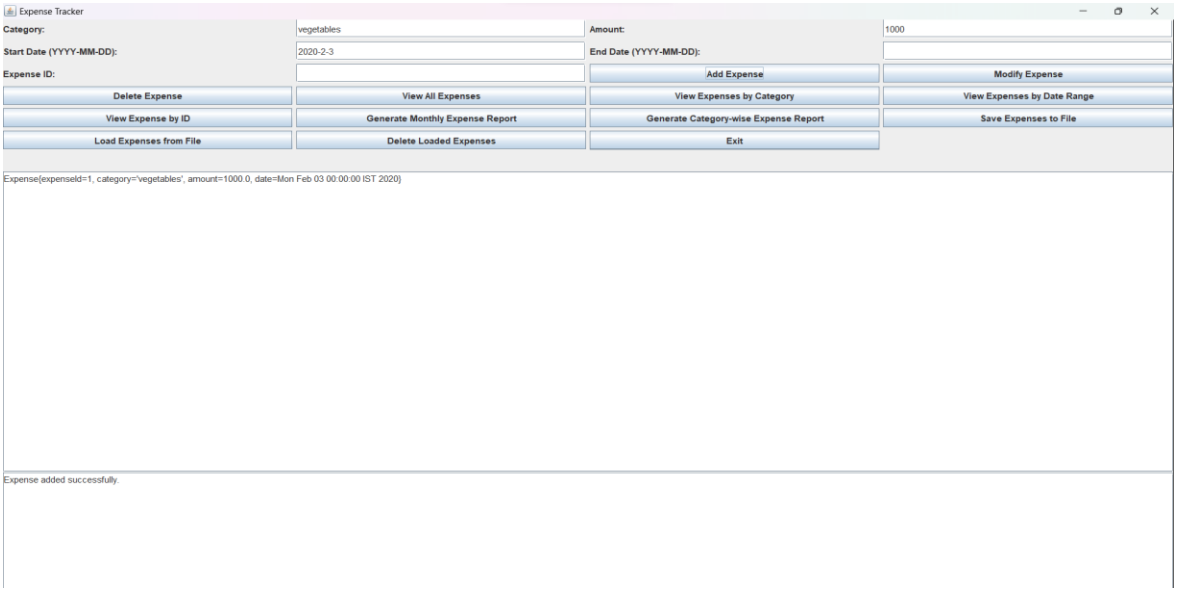
Based on the tested features, the Expense Tracker application showed the following performance outcomes:

- Memory Usage: The application efficiently managed memory, with no significant memory leaks or excessive usage even with substantial data input.
- Processing Speed: The MIPS (speed) for key operations, such as recording, modifying, and deleting expenses, remained within acceptable ranges, providing a responsive user experience.
- File Handling: Loading and saving expenses from files were optimized to minimize processing time, ensuring quick data access and retrieval.
- Report Generation: Generating monthly and category-wise expense reports was performed swiftly, even with considerable data, delivering actionable insights promptly.

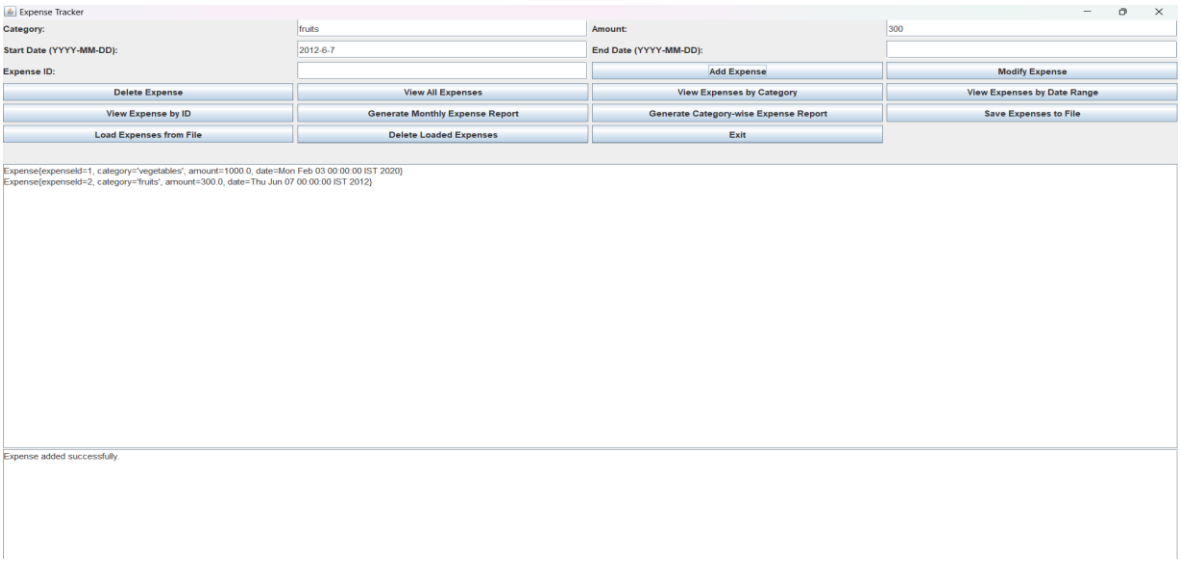
By carefully considering and addressing these constraints and conducting thorough testing, my Expense Tracker project can be designed to handle real-world scenarios efficiently, accurately, and durably. This will ensure it meets the needs of users in various industries and professional settings while offering a smooth and reliable experience.

6.4 EXAMPLES OF TEST CASES ON MY PROJECT:

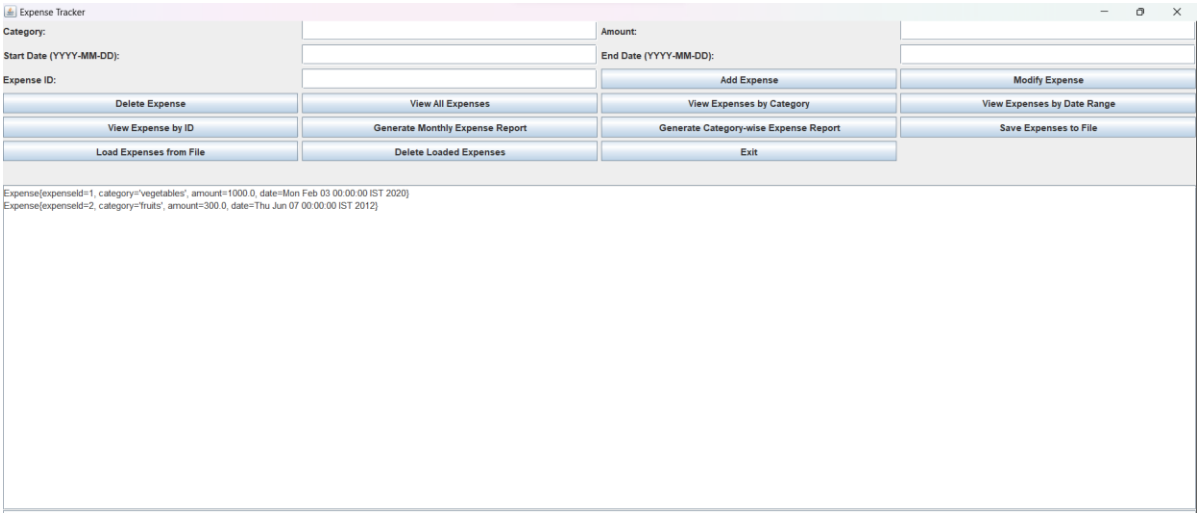
1.Test Case on feature no 1 that is Record Expenses.



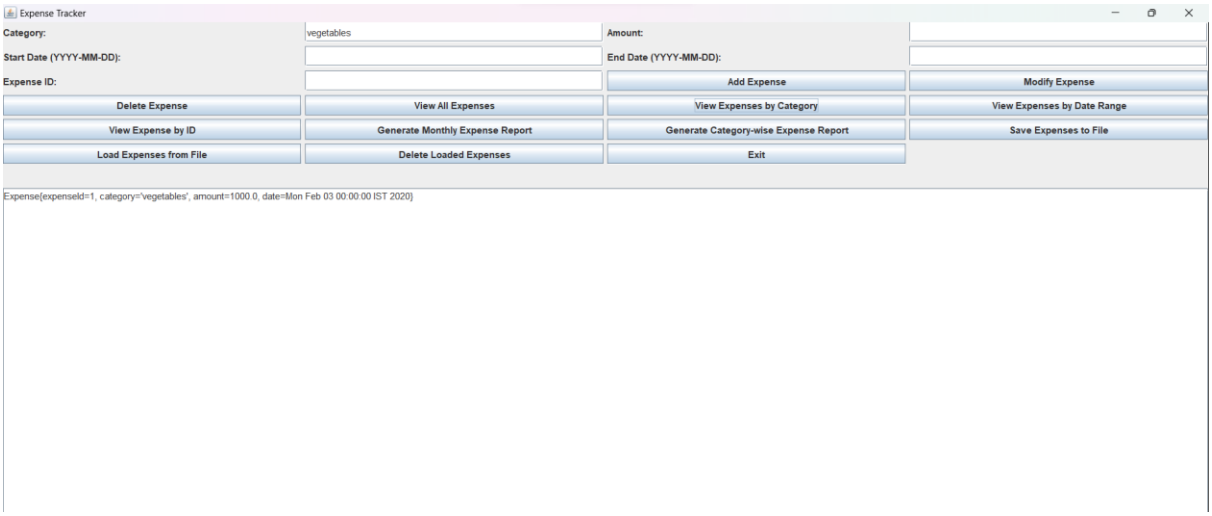
2.Test Case on feature no 1 that is Record Expense



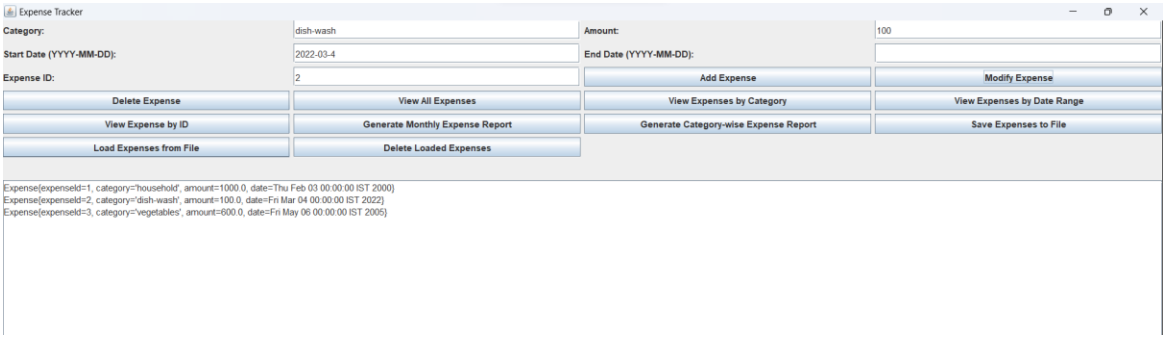
3.Test Case on feature no 4 that is View all Expenses



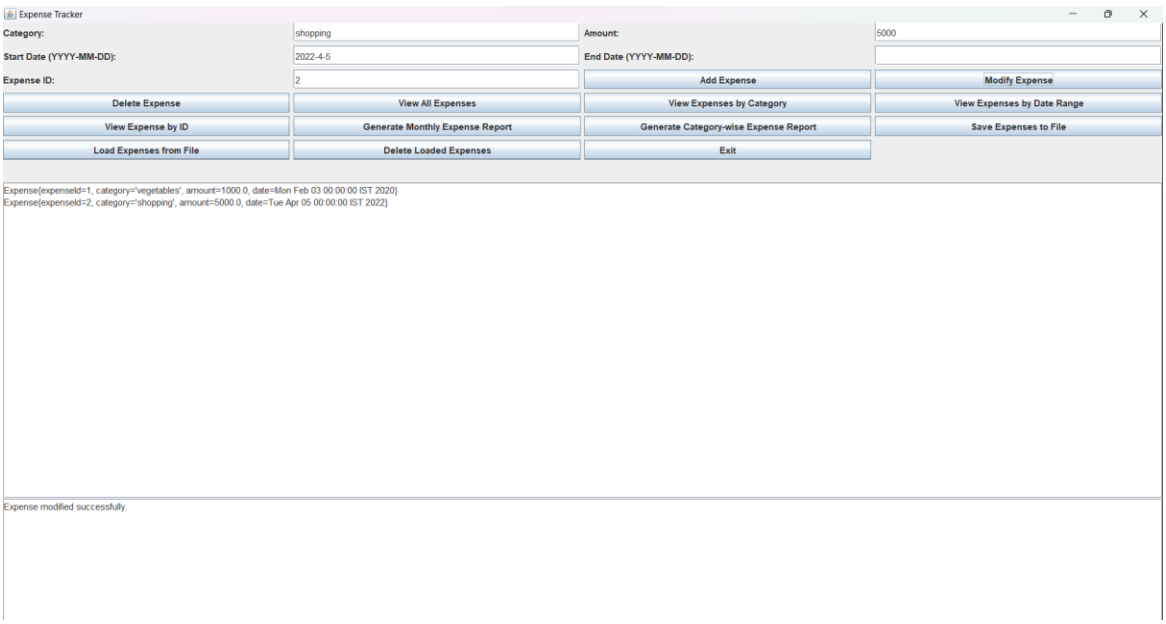
4.Test Case on feature no 5 that is View Expense by Category



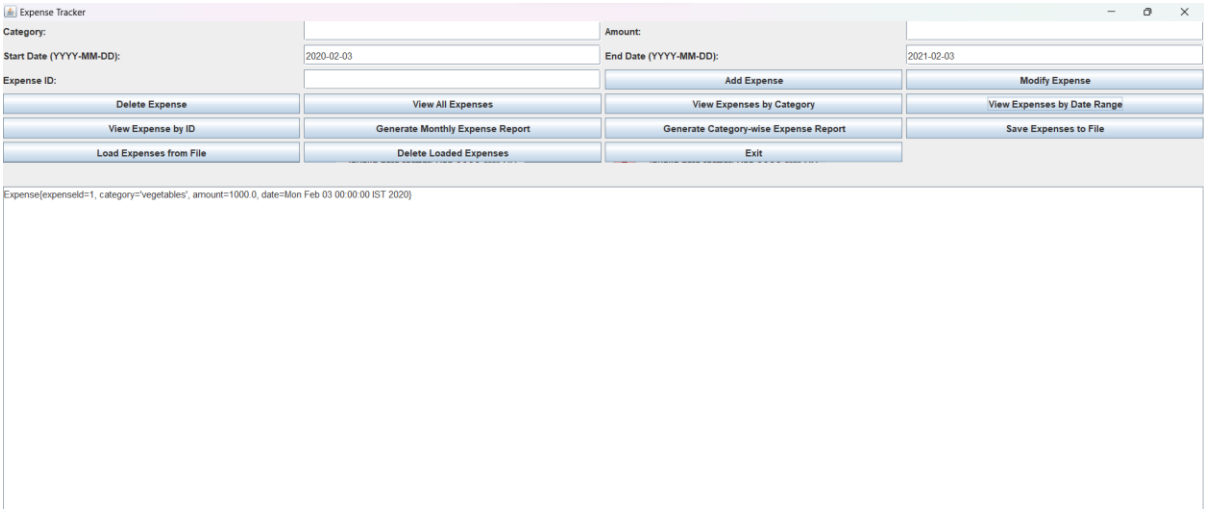
5.Test Case on feature no 2 that is Modify Expenses.



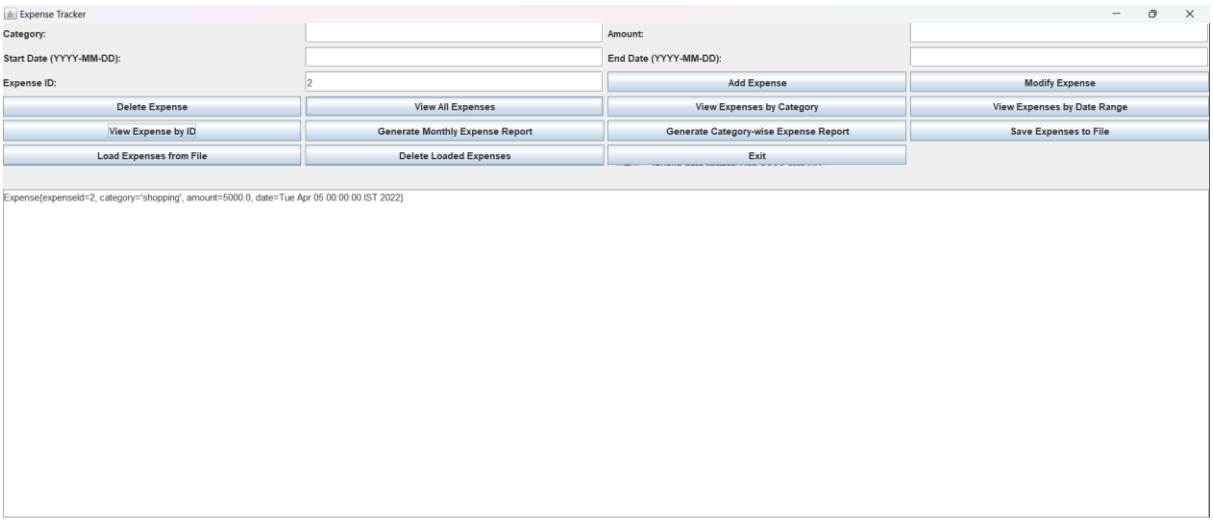
6.Test case to show that data has been modified properly.



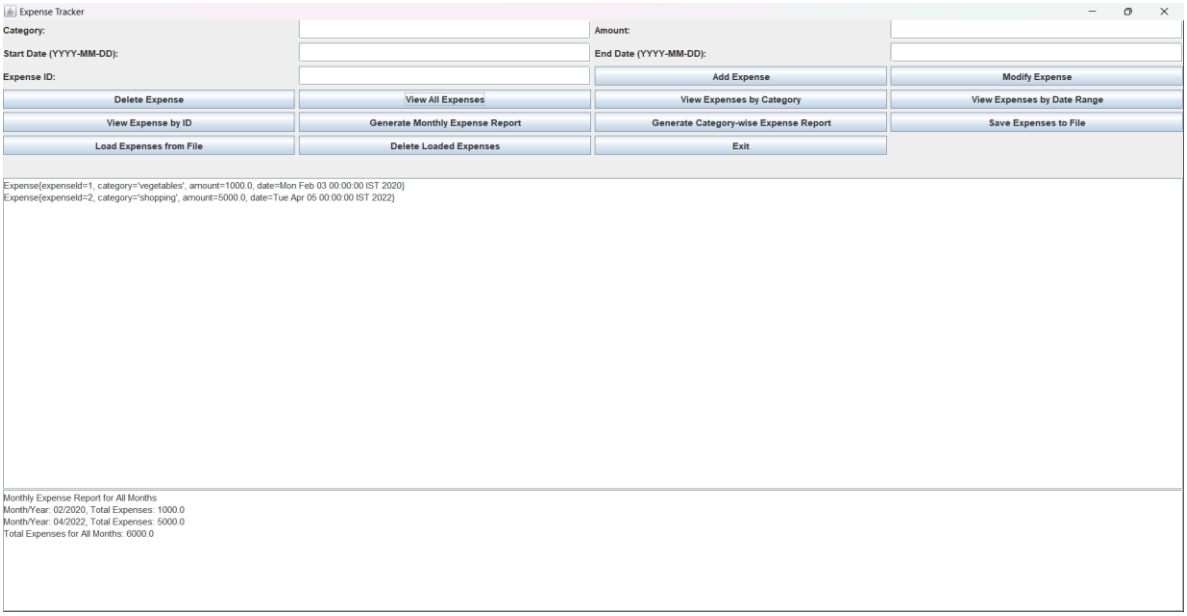
7. Test Case on feature no 6 that is View Expenses by Date Range



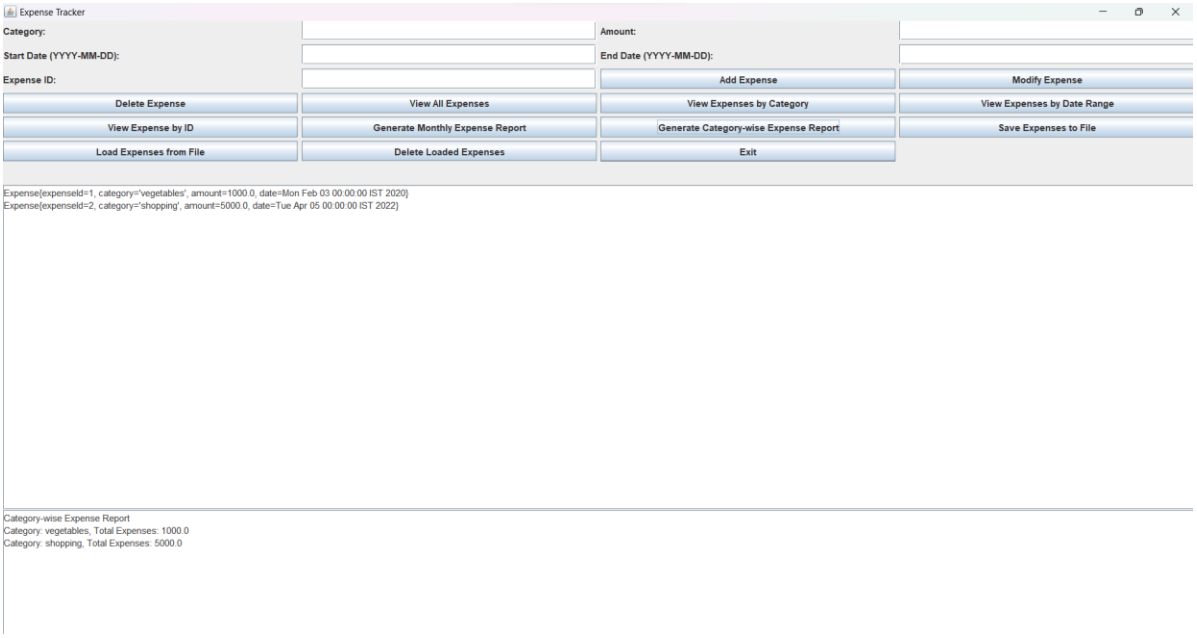
8 Test Case on feature no 7 that is View Expenses by Expense-Id.



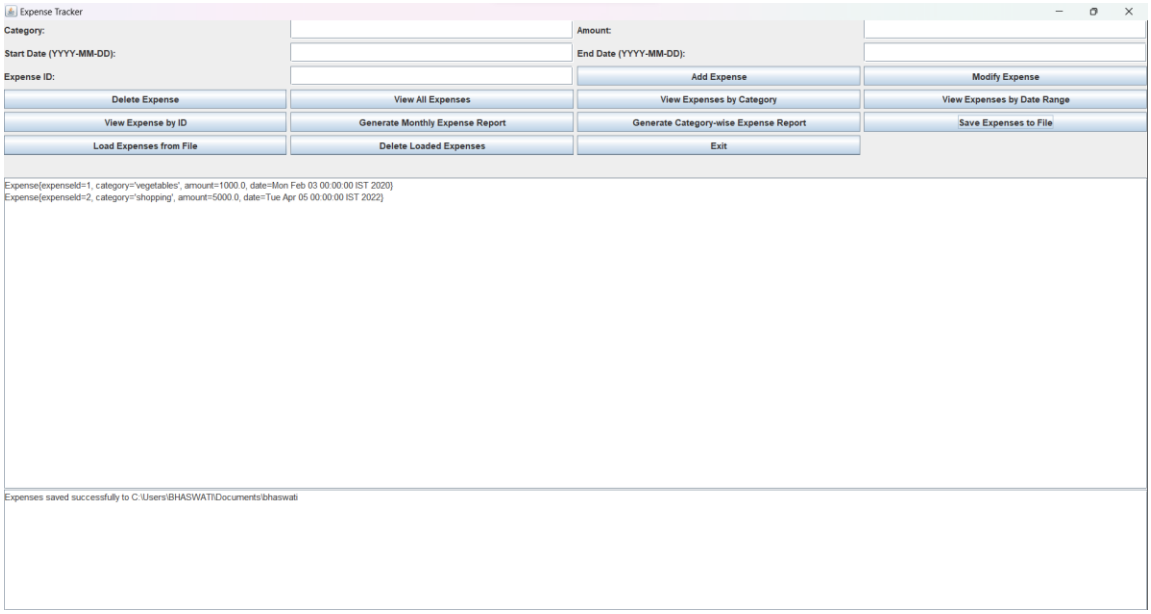
9.Test Case on feature no 8 that is Generate Monthly expense Rate



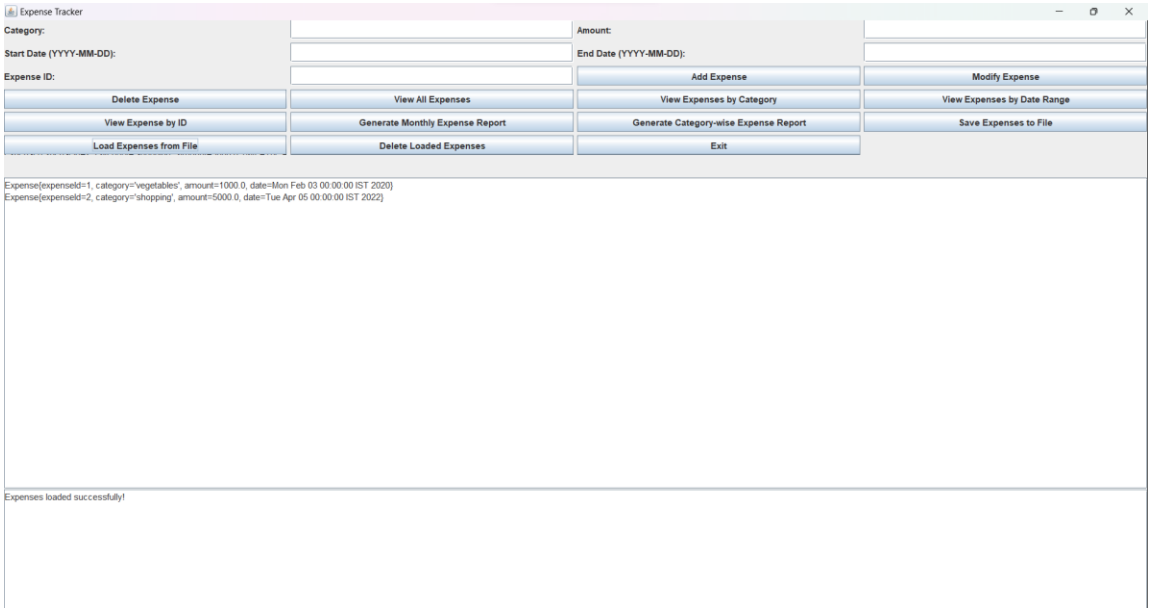
10.Test Case on feature no 9 that is Generate Category Wise Expense Report.



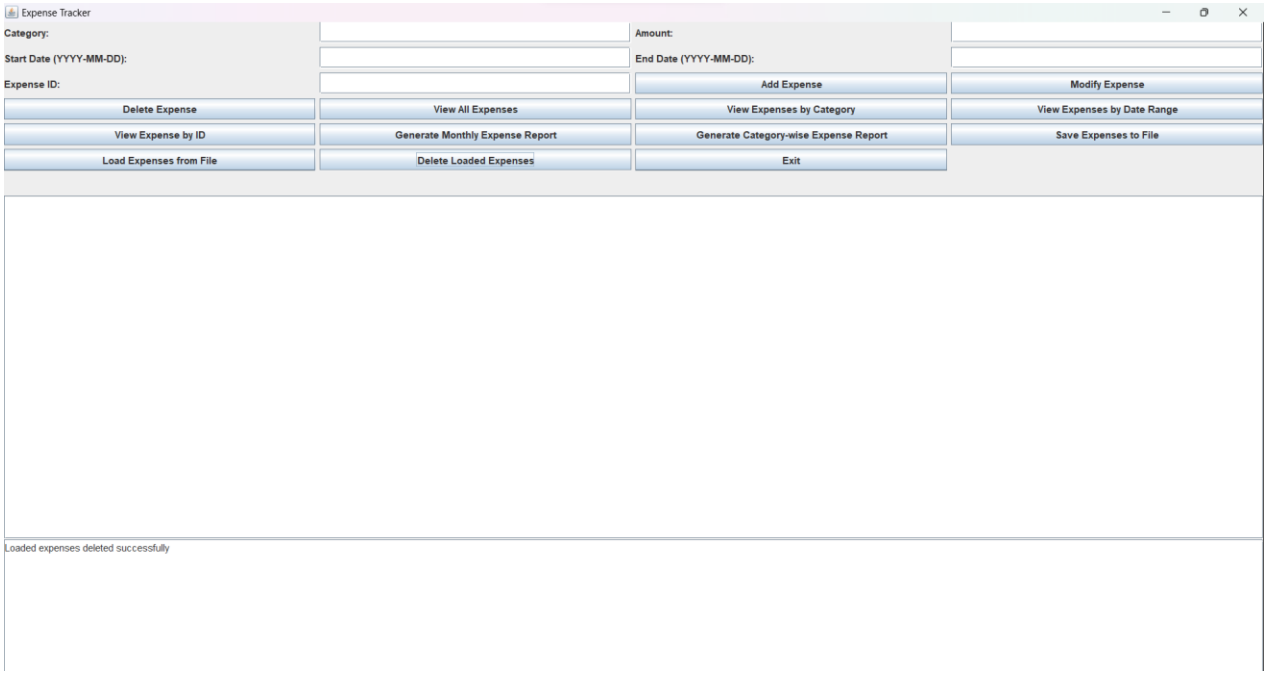
11.Test case on feature no 10 that is to Save Expenses to file.



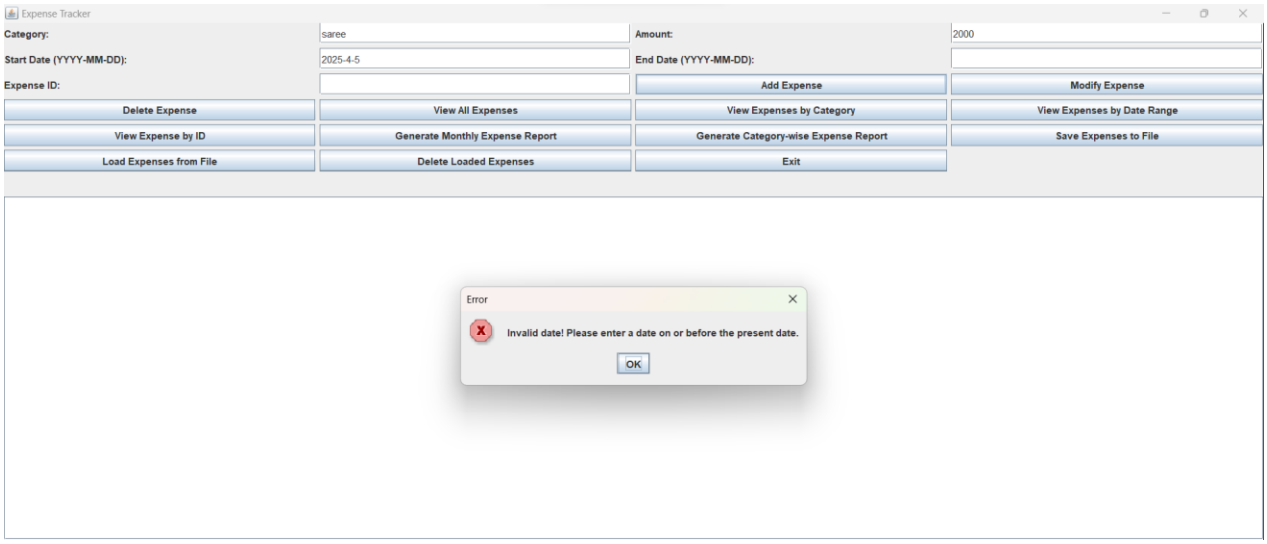
12.Test Case on feature no 11 that is Load Expenses from the file.



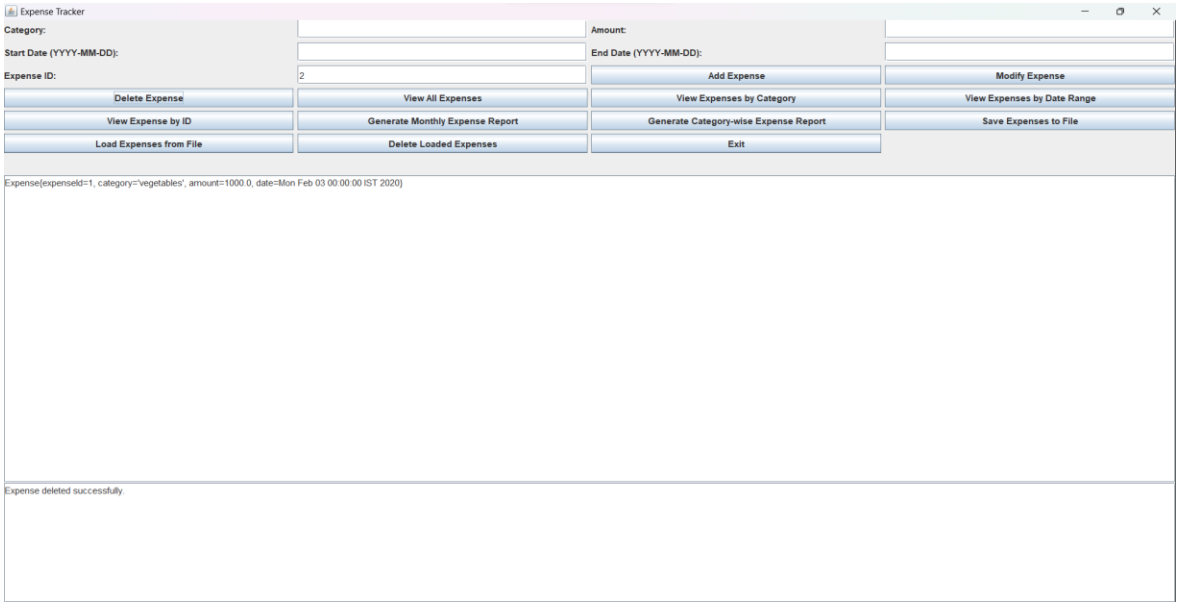
13.Test Case to show the Loaded expenses has been deleted successfully.



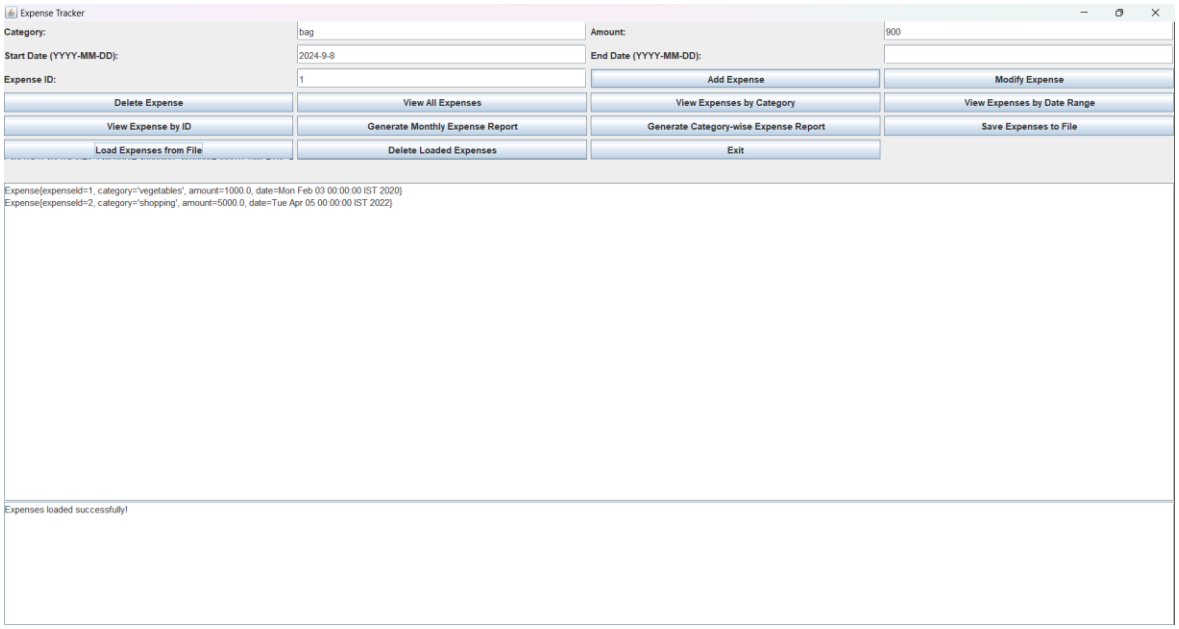
14.Test case to show Error-handling



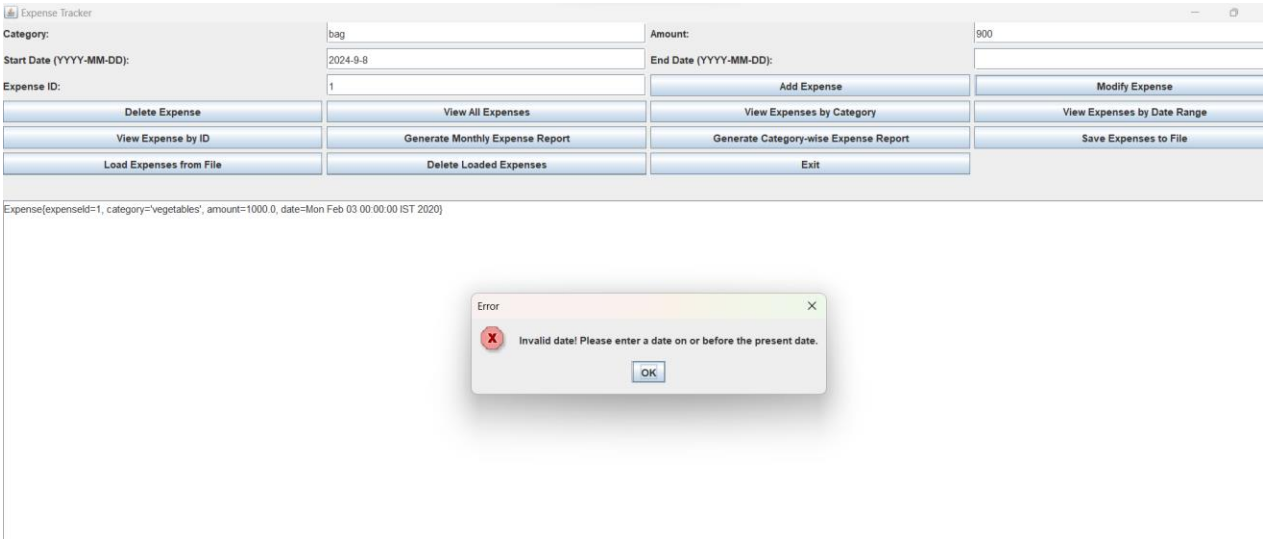
15. Test Case to show feature no 3 that is Delete Expense.



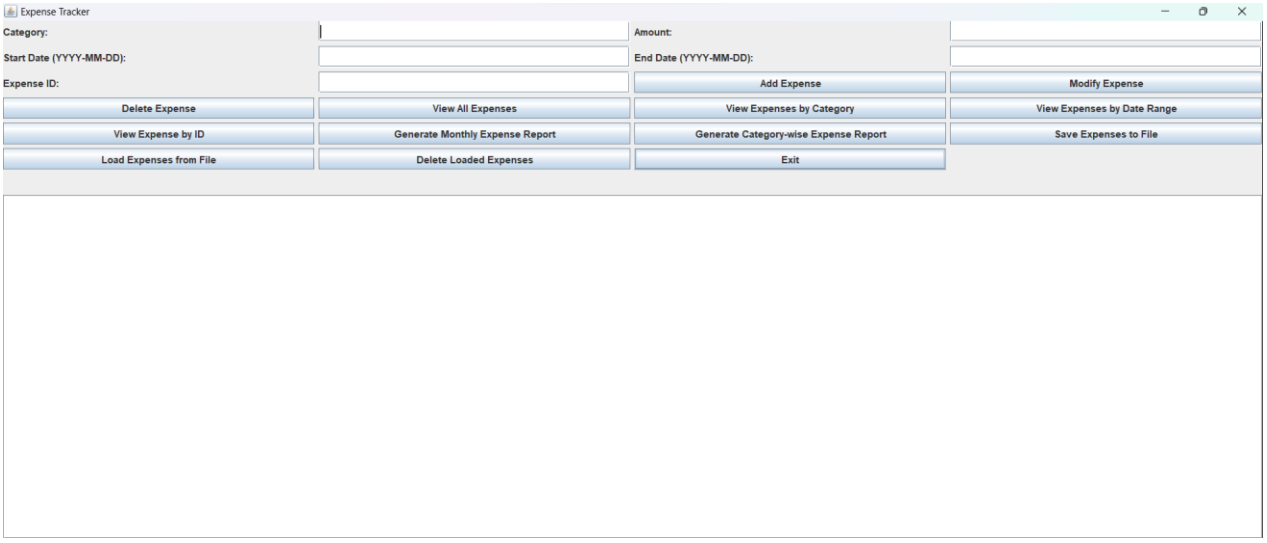
16. Test Case to show that after deleting also data can be loaded from load expenses from file



17.Test Case to show Error Handling on feature no 3 that is Modify Expenses.



18. Test case to show Exit feature. After clicking on the exit is has been exited successfully.



7.MY LEARNINGS

The core Java project of building a console-based expense tracker would have provided valuable learning experiences and skill development. Through this project, I would have honed my Java programming skills, gained proficiency in data structures, and learned about file handling for data persistence. The project would have deepened my understanding of object-oriented programming principles, error handling, and modular code design.

Additionally, working on this project would have improved my problem-solving abilities as I tackled various challenges related to user input validation, report generation, and efficient data management. I would have gained practical experience in software development, understanding how to create a functional and user-friendly application.

The exposure to real-world project development and the application of concepts learned during the internship would have given me the confidence to take on more complex software projects in the future. This project's successful completion would serve as a valuable addition to my portfolio, showcasing my abilities as a Java developer to potential employers.

Overall, the core Java project would have contributed significantly to my career growth by strengthening my programming foundation, enhancing problem-solving skills, and providing practical experience in building software applications. It would have opened doors to various opportunities in the software development industry and positioned me as a competent and capable developer ready to take on new challenges.

8.FUTURE WORK SCOPE

If I had more time I would like to implement this features on my project .

1.Budget Tracking and Alerts: Allow users to set budget limits for different expense categories.

Provide real-time alerts or notifications when users approach or exceed their budget limits, helping them stay on top of their spending.

2.Data Analysis and Insights: Implement advanced data analysis to identify spending patterns, trends, and insights from expense records.

Provide graphical representations and visualizations to help users understand their financial behavior better.

3.Expense Attachments: Allow users to attach receipts or images related to expenses, providing better documentation and visualization of transactions.

4.Expense Splitting with Friends/Contacts: Enable users to split expenses with friends or contacts, facilitating easy tracking of shared expenses among multiple individuals.

5.Expense Reminders: Set up reminders for users to enter their expenses regularly, ensuring that they keep track of their spending consistently.

By incorporating these advanced features, my Expense Tracker project will offer users more control over their finances, provide valuable insights, and streamline expense management. These enhancements would elevate the application's functionality, making it a powerful tool for effective expense tracking and financial management.