

Personal Finance Tracker

A PROJECT REPORT

Submitted by

KARANPREET SINGH	22BCS12898
GURLIN KAUR SANDHU	22BCS12649
DEEPIKA RANI	22BCS12754

in partial

fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE



Chandigarh University

April 2025



BONAFIDE CERTIFICATE

Certified that this project report “**Personal Finance Tracker**” is the bonafide work of Karanpreet Singh, Gurlin Kaur Sandhu & Deepika Rani who carried out the project work under my supervision.

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We would like to take this opportunity to express our heartfelt gratitude to our teacher for guiding us through the project. Your unwavering support, encouragement, and guidance have been instrumental in completing this project successfully. Your passion for the subject matter and your dedication to teaching have inspired us to work hard and pursue our goals relentlessly. Your constructive feedback and valuable inputs have helped us improve our work and broaden our perspective. Your patience and kindness have made it easy for us to approach you with any doubts or questions, and your willingness to go above and beyond to help us has been truly remarkable. Your commitment to your students' success is evident in the way you give your time and attention to each of us. We are grateful to have had the opportunity to learn from you, not just about the subject matter, but also about life. Your words of wisdom and encouragement have stayed with us and will continue to guide us in our future endeavours. Once again, thank you for your guidance and support throughout the project. We are proud of what we have achieved, and we owe it all to you.

Karanpreet Singh

Gurlin Kaur Sandhu

Deepika Rani

(Students BE-CSE, 5th Semester)

TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION	8
1.1 Identification of Client/ Need/ Relevant Contemporary issue	8
1.2 Identification of Problem	9
1.3 Identification of Tasks	9
1.4 Timeline.....	10
1.5 Organization of the Report	11
CHAPTER 2. DESIGN FLOW/PROCESS.....	12
2.1 Evaluation & selection of specifications/Features.....	12
2.2 Design Constraints	12
2.3 Analysis and feature finalization subject to constraints	13
2.4 Design flow.....	13
2.5 Design selection	14
2.6 Implementation plan/methodology	14
CHAPTER 3. RESULT ANALYSIS AND VALIDATION	16
3.1 Implementation of solution	16
CHAPTER 4. CONCLUSION AND WORK	18
4.1 Conclusion	18
4.2 Future Work	18

List of Figures

Figure 1.1 Gantt Chart.....	10
Figure 2.1 Flowchart	14
Figure 3.1 Home Directory.....	16
Figure 3.2 addTransactions.jsp	16
Figure 3.3 viewTransactions.jsp.....	17

List of Tables

Table 1.1.....	11
----------------	----

ABSTRACT

The Personal Finance Tracker is a lightweight, web-based application developed using Java, Servlets, JSP, and XML to help individuals manage and analyze their financial transactions with ease and security. Designed for offline usage, the system enables users to add, store, and review income and expense records without relying on cloud services or external databases. XML is employed as the primary data storage format, ensuring portability, transparency, and simplicity while maintaining a structured representation of user transactions. The application features a clean user interface developed in JSP, allowing users to seamlessly input transaction details such as date, category, amount, and description. Servlets handle the core logic, including data processing and XML manipulation, ensuring a modular and maintainable codebase. The project adheres strictly to standard Java EE technologies, avoiding modern libraries and frameworks to focus on core Java web development skills. The system was designed under constraints such as limited development time, offline operability, and strict data privacy requirements, making it suitable for academic and personal environments. Through iterative testing and validation, the application demonstrated effective performance in terms of functionality, usability, and reliability. This project not only serves as a practical financial tool but also provides a strong foundation for future enhancements such as database integration, analytics, and multi-user support.

CHAPTER 1.

INTRODUCTION

1.1. Identification of Client /Need / Relevant Contemporary Issue:

The primary client for this project is the individual user—specifically students, working professionals, or homemakers—who require a simple and efficient way to manage their personal finances. These individuals often lack access to premium finance tools or the technical expertise to use complex financial software. In addition to the client, several key stakeholders influence the development and success of the project. These include academic mentors who guide the technical direction, peer users who provide real-time feedback, and the project developers themselves, who are responsible for interpreting user needs into an executable application. While end users focus on ease of use and functionality, mentors prioritize architectural soundness and code quality, making stakeholder interaction central to aligning the solution with both usability and academic expectations.

The need for such a solution arises from the increasing demand for lightweight, user-friendly, and offline financial tools. Through informal surveys and observational research among peers and young professionals, it was found that existing personal finance applications are often cloud-based, overly feature-heavy, or require account integration—factors that make them inaccessible or unappealing to casual users. Moreover, tools like spreadsheets are tedious to maintain and are error-prone, while subscription-based apps raise concerns about long-term usability and affordability. The essential requirements identified include a web interface for adding and viewing transactions, the ability to store data securely without internet dependency, and full control over financial records. XML was chosen as the storage format to meet transparency, simplicity, and compatibility goals. The technical and logistical constraints—such as limited time, non-use of databases, and focus on core Java EE technologies—were considered while prioritizing features that could be delivered effectively within the scope of a mini-project.

The broader contemporary context further justifies the relevance of the project. In an era of growing concern over data privacy, users are increasingly cautious about sharing personal financial information with third-party applications. The shift towards digital minimalism and offline tools has created a niche for basic yet functional applications that work independently of large infrastructures or persistent internet connections. Additionally, with economic uncertainties on the rise, there is a growing cultural emphasis on personal financial awareness and budgeting, especially among the younger population. From a technological standpoint, the project is timely—it reflects modern trends in web-based software architecture using Java Servlets, JSP, and XML, making it not only a practical solution but also an academically enriching tool aligned with current educational and industry standards. Thus, this project stands at the intersection of a real-world need, technological feasibility, and contemporary relevance.

1.2. Identification of Problem

- **Difficulty in Tracking Daily Expenses:** Many individuals, especially students and early-career professionals, struggle with consistently monitoring their spending habits due to a lack of effective tools.
- **Over-complexity of Existing Finance Apps:** Most available applications are bloated with features, require online accounts, or have subscription models, discouraging casual users.
- **Data Privacy Risks:** Cloud-based solutions often pose concerns regarding the safety and misuse of personal financial information.
- **Lack of Offline Tools:** There is a clear gap in the market for simple, lightweight tools that work offline and provide essential finance tracking functionalities.
- **Limited Customization in Available Software:** Users often want a personalized interface and control over data formatting (e.g., XML), which existing apps rarely provide.
- **Learning Curve of Advanced Tools:** Many popular personal finance applications have steep learning curves or require syncing with banks, which is not ideal for basic, local usage.
- **Lack of Modular, Educationally Relevant Solutions:** There's a shortage of finance-tracking tools that are modular and ideal for academic environments, both for learning and for user-based customization.
- **Manual Maintenance Burden:** Traditional spreadsheet methods require manual calculations and updates, often leading to user errors and inconsistency in recordkeeping.
- **Disorganized Financial Records:** Without structured digital tools, people often lose track of transactions, budgets, or recurring expenses, leading to poor financial planning.

1.3. Identification of Tasks

- **Identify Focus Area and Functional Objectives:** The primary focus is to create a web-based personal finance tracker that stores financial transactions using XML and allows user interaction through Java Servlets and JSP. The goal is to provide a minimal yet effective platform to record income and expenses, categorize transactions, and review financial history.
- **Conduct Informal Research and User Needs Assessment:** Informal discussions and surveys with students and early-career individuals were conducted to identify pain points with existing

solutions. The findings helped define feature requirements like offline usage, local data storage, and a clean UI.

- **Develop Theoretical Framework and Feature Set:** The project leverages Java Servlet technology for request handling, JSP for dynamic HTML rendering, and XML as a structured data storage format. Each financial transaction consists of date, amount, category, and description tags within an XML document.
- **Design Architecture and Develop Modules:**
Break down the system into clear modules:
 - User Interface (JSP-based forms and pages)
 - Controller Layer (Servlets for routing and logic)
 - Data Layer (XML handling using DOM/SAX parsers)
 - Deployment Configuration (web.xml and Tomcat structure)
- **Test Implementation and Review Data Output:** Unit tests for individual servlet methods and XML generation/reading processes will be performed. Transaction addition and viewing features will be validated for accuracy and user experience.
- **Prepare and Finalize Documentation:** Structure the final report into chapters including Introduction, Design, Implementation, Testing, and Future Scope. Include code snippets, diagrams, and screenshots where applicable. The documentation will ensure technical clarity, user accessibility, and academic completeness.

1.4. Timeline



Figure 1.1

1.5. Organization of the Report

Chapter 1 Problem Identification: This chapter introduces the project and describes the problem statement discussed earlier in the report.

Chapter 2 Design Flow/ Process: The proposed objectives and methodology are explained. This presents the relevance of the problem. It also represents a logical and schematic plan to resolve the research problem

Chapter 3 Result Analysis and Validation: This chapter explains various performance parameters used in implementation. Experimental results are shown in this chapter. It explains the meaning of the results and why they matter.

Chapter 4 Conclusion and future scope: This chapter concludes the results explains the best method to perform this research to get the best results and defines the future scope of study that explains the extent to which the research area will be explored in the work.

CHAPTER 2.

DESIGN FLOW/PROCESS

3.1. Evaluation & Selection of Specifications/Features:

In developing the Personal Finance Tracker, the selection of features began with identifying essential user requirements that could deliver maximum value through a simplified interface and backend logic. Key features such as income/expense entry, category-based tracking, and transaction review were shortlisted based on usability and feasibility. The primary focus was on implementing core functions that aid users in monitoring their finances efficiently. The specifications took into account factors such as:

- **Data Structure Compatibility:** XML was chosen for its simplicity, readability, and compatibility with Java, making it ideal for storing structured financial data without external dependencies.
- **Technology Stack:** The use of Java Servlets and JSP ensures the application is browser-accessible, modular, and aligns with educational objectives.
- **System Portability:** The system is lightweight and deployable on any machine with a Java Runtime Environment and a servlet container like Tomcat.

Through iterative evaluation, the selected features ensured a balance between functionality, user simplicity, and system efficiency without overwhelming the end-user or complicating the backend logic.

3.2. Design Constraints:

While designing the system, several significant constraints shaped the overall implementation approach, demanding careful architectural decisions. A major limitation was the restriction against database integration, requiring XML to serve as the sole medium for data storage. This introduced complexities in handling larger datasets and performing queries, as XML lacks the inherent efficiency of relational databases, necessitating manual parsing and data manipulation. Additionally, the application had to be entirely offline-capable and platform-independent, mandating that all operations be locally executed and file-based without any reliance on internet connectivity or cloud infrastructure. To accommodate these constraints, the design emphasized simplicity due to limited time and resources,

yet still prioritized security and privacy, ensuring that sensitive financial data remained confined to the user's system. Compounding the challenge was the requirement to use only standard Java EE components, which ruled out the adoption of modern frameworks and libraries, thereby forcing manual implementation of session management, form validation, and file operations. Despite these hurdles, the final solution was crafted to be lightweight, secure, and functionally robust, offering users a self-contained personal finance tool that respected their privacy while adhering to the project's strict technological boundaries.

3.3. Analysis and Feature finalization subject to constraints:

After identifying potential features and understanding the design limitations, the team finalized a feature set optimized for both performance and simplicity. XML-based data handling was analyzed and proven suitable for the scale of this project, ensuring users could save and retrieve transactions seamlessly. Input validation and servlet-routing logic were tested to confirm reliable transaction flow from front-end JSP forms to XML storage. Features like editable transaction records or graphical analysis were initially considered but postponed due to file manipulation limitations in XML and time constraints. The finalized features include adding new transactions, listing them, and displaying category-wise breakdowns, offering essential financial oversight while remaining aligned with the technical constraints.

3.4. Design Flow:

The application follows a clear and modular design flow structured around the MVC (Model-View-Controller) architecture. The process begins with the user interface (JSP pages) where users input transaction data. These inputs are handled by Java Servlets (Controllers) which process the data and update the XML file accordingly. A utility class handles XML read/write operations to maintain separation of concerns. When users request to view transactions, the controller fetches the XML data, parses it, and passes it to the corresponding JSP for rendering. Each stage of the flow—input, processing, storage, and display—is tightly integrated, allowing a smooth user experience and simplified backend logic.

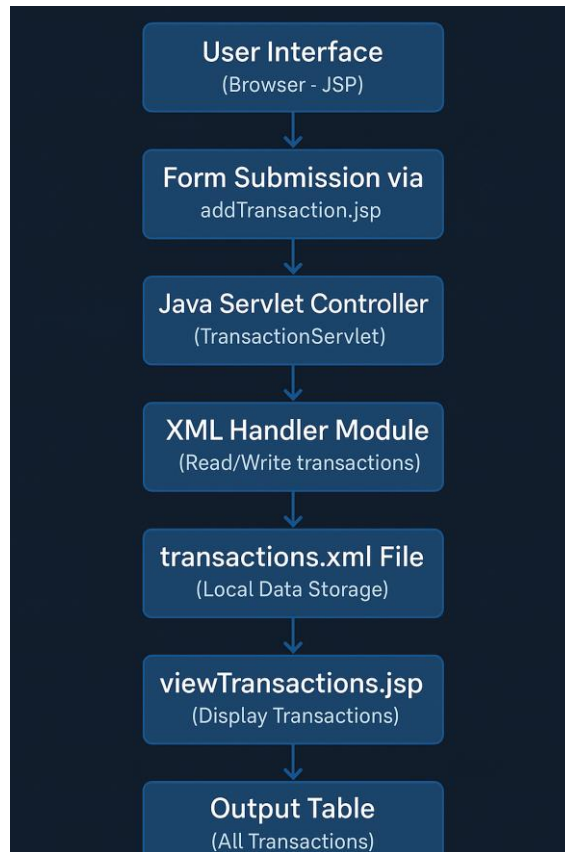


Figure 2.1

3.5. Design selection:

Two primary design approaches were considered: a Java desktop application using Swing, and a web-based solution using Servlets and JSP. The web-based approach was selected for its broader accessibility, platform independence, and alignment with Java EE learning objectives. Unlike desktop apps which are limited to local environments, JSP-servlet architectures allow use via browsers and offer modular structuring. Within the selected approach, various data storage options were evaluated. While database integration would offer better scalability, it was excluded to maintain offline compatibility and reduce setup overhead. XML emerged as the best alternative, balancing structure, ease of use, and local operability. Thus, the chosen design prioritizes simplicity, portability, and user control over data.

3.6. Implementation plan/methodology:

The implementation is divided into clearly defined stages to ensure smooth development and integration of system components:

- **Stage 1: UI and Frontend Creation** – JSP pages like `addTransaction.jsp` and `viewTransactions.jsp` were designed with basic forms and tables for input and output.
- **Stage 2: Backend Servlet Development** – Servlets were developed to handle form submissions, parse XML, and control routing.
- **Stage 3: XML Data Integration** – A custom XML utility module was built to perform read/write operations using DOM parsers in Java.
- **Stage 4: Testing and Validation** – The system was tested for data accuracy, XML consistency, and page routing under different use scenarios. Manual edge-case testing ensured robust input handling and fault tolerance.

Each phase supported iterative improvement and allowed for real-time debugging and refinement, ensuring the application met both academic and practical objectives..

CHAPTER 3.

RESULT ANALYSIS AND VALIDATION

3.1. Implementation of solution:

The implementation of the Personal Finance Tracker was carried out using a structured modular approach, integrating both front-end and back-end components via Java-based web technologies. The development process utilized modern engineering tools within the scope of standard Java EE practices, with the core components comprising Java Servlets, JSP (Java Server Pages), and XML for persistent data storage.

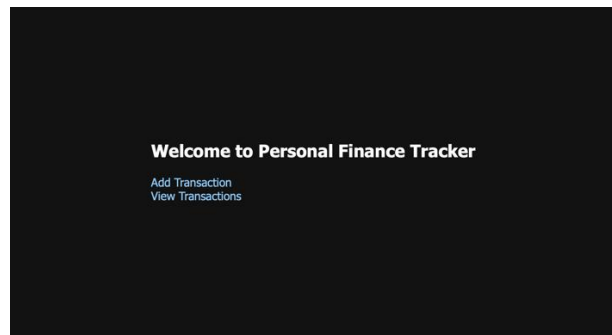


Figure 3.1

The front-end of the system was built using JSP, HTML, and CSS to provide a user-friendly and responsive interface. JSP pages such as `index.jsp`, `addTransaction.jsp`, and `viewTransactions.jsp` allowed users to interact with the application seamlessly. These pages were styled with basic CSS to enhance readability and guide users through transaction entry and review processes.

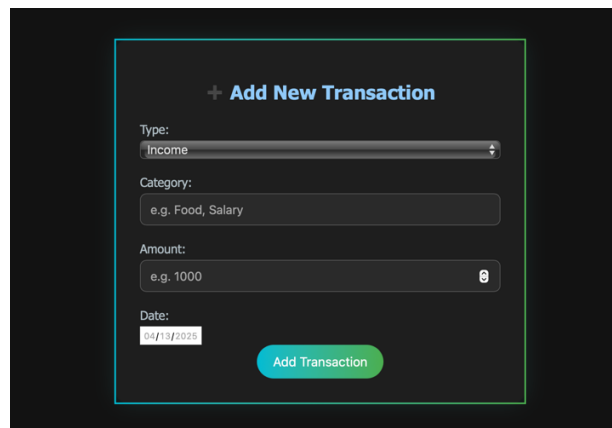
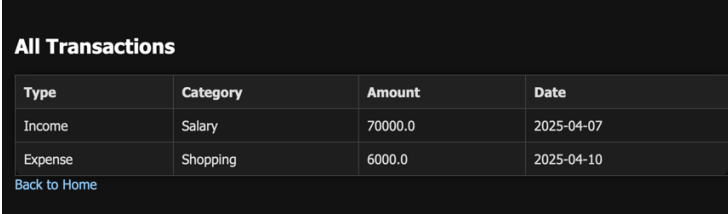


Figure 3.2

The image shows a screenshot of a web application interface. At the top, there is a header titled "All Transactions". Below the header is a table with four columns: "Type", "Category", "Amount", and "Date". The table contains two rows of data. The first row shows an "Income" transaction with a "Salary" category, an amount of "70000.0", and a date of "2025-04-07". The second row shows an "Expense" transaction with a "Shopping" category, an amount of "6000.0", and a date of "2025-04-10". Below the table, there is a link labeled "Back to Home".

Type	Category	Amount	Date
Income	Salary	70000.0	2025-04-07
Expense	Shopping	6000.0	2025-04-10

[Back to Home](#)

Figure 3.3

On the back-end, Java Servlets were developed to handle all business logic, including the processing of form inputs and interaction with the XML data store. When a user submits a new transaction, the request is handled by a servlet that parses the input, constructs a transaction node, and appends it to the existing XML file using DOM manipulation. A utility class was implemented for abstracting the XML read/write operations to ensure modularity and maintainability.

The data storage solution was built entirely on XML to conform to the offline, file-based requirement of the project. Each transaction is stored in a well-structured XML format with tags representing date, amount, category, and description. The data is parsed and rendered dynamically in the `viewTransactions.jsp` page, which displays all recorded transactions in a tabular layout.

Testing and validation were performed iteratively. The XML structure was validated manually and through the DOM parser to ensure no malformed data was written. Servlet responses were tested for null inputs, missing parameters, and invalid transaction values. Each module was tested independently and as part of the integrated system to verify accuracy, functionality, and data persistence.

The project was developed and tested using IntelliJ IDEA as the IDE, and Apache Tomcat as the local server environment. The application was successfully deployed and executed on machines running the Java Runtime Environment (JRE), fulfilling the goals of accessibility, platform-independence, and offline operability. Overall, the implementation demonstrates a well-organized approach to solving the problem of personal finance tracking using web-based Java technologies and structured file storage..

CHAPTER 4.

CONCLUSION AND FUTURE SCOPE OF WORK

4.1 Conclusion:

The Personal Finance Tracker was successfully developed using Java, Servlets, JSP, and XML, fulfilling the core objective of enabling users to efficiently track, manage, and review their personal financial transactions through a simple and accessible web-based interface. By adopting a modular design and lightweight architecture, the system ensures easy deployment and offline usability, making it suitable for users without access to complex financial tools or cloud-based services.

The application allows users to add transactions with relevant details such as date, category, amount, and description, which are stored locally in an XML file. A well-structured interface allows users to view all recorded transactions dynamically through the browser, offering transparency and ease of use. The selection of XML as the storage medium meets the requirement for readability, flexibility, and independence from external databases.

While the project met its expected outcomes, some limitations were observed. These include the lack of advanced search, filtering, or editing capabilities and no graphical representation of financial trends. The use of XML, while effective for small datasets, may not scale efficiently for long-term usage or larger data volumes. Nevertheless, the project demonstrated a complete functional pipeline from user interaction to data persistence, validating the feasibility and relevance of the solution.

4.2 Future Scope:

There is significant potential to extend and enhance the current system to make it more feature-rich, scalable, and intelligent. Some recommended future enhancements include:

- **Database Integration:** Transitioning from XML to a relational (e.g., MySQL) or NoSQL (e.g., MongoDB) database would allow better scalability, faster querying, and improved data integrity.
- **User Authentication System:** Adding login/logout functionality would enable multi-user support and secure access to individual financial records.
- **Graphical Analytics:** Implementing charts and visual reports (e.g., pie charts, bar graphs) using libraries like Chart.js or Google Charts could provide users with insights into their spending patterns.

- **Search and Filter Features:** Allow users to search for transactions by date, category, or amount range to make navigation easier.
- **Export Functionality:** Enabling users to export their financial data in CSV or PDF format for external use or backup.
- **Mobile Responsiveness:** Enhancing UI responsiveness to ensure compatibility with smartphones and tablets for on-the-go access.
- **AI Integration:** Incorporating basic financial intelligence (e.g., monthly summaries, expense categorization, and budget prediction) using machine learning or rule-based logic.

These improvements can significantly elevate the system's usefulness and adaptability, transforming it from a basic tracker into a powerful personal finance management suite. The current project serves as a solid foundation for such future developments.

REFERENCES

- Eckel, B. (2006). *Thinking in Java* (4th ed.). Prentice Hall.
A comprehensive resource for understanding object-oriented programming in Java, including servlets and web-based application design.
- Hunter, E., & Crawford, W. (2004). *Java Servlet Programming* (2nd ed.). O'Reilly Media.
An authoritative guide on Java Servlets and JSP, providing essential knowledge for developing dynamic web applications.
- Kay, M. (2003). *XSLT: Programmer's Reference* (2nd ed.). Wrox Press.
A key reference for understanding XML structure, parsing, and transformations, applicable in handling transaction data for the tracker.
- Bhardwaj, S., & Yadav, A. (2020). *A Study on Personal Finance Management Applications: Usage, Benefits, and User Behavior*. *International Journal of Engineering Research & Technology (IJERT)*, 9(6), 314–319.