Modern Application Development
(Java Spring Boot)

Project Title:

Personal Expenses Tracker

College: Vellore Institute of Technology

Team No: 427

Team Members:

20BME0250 – SRIRAM VARMA UPPALAPATI

20BCI7177 – AASHISH REDDY TADURI

20BCE7387 – SAI DEEPAK POKURI

20BCI7078 – PRATHEEK SHETTY

1. **Introduction**:

1.1 **Overview**

The "Personal Expenses Tracker" is a project aimed at helping individuals manage and track their personal finances. It is a digital tool or application designed to assist users in recording and categorizing their expenses, monitoring their spending habits, and gaining insights into their financial behaviors.

The primary purpose of the Personal Expenses Tracker is to provide users with a convenient and organized way to keep track of their income and expenses. Users can input their financial transactions, such as purchases, bills, and income sources, into the tracker. These transactions are typically categorized into different expense categories like groceries, entertainment, transportation, and so on.

By consistently logging their expenses, users can develop a clearer understanding of their spending patterns and identify areas where they may be overspending or can make adjustments to save money.

1.2 **Purpose**

The "Personal Expenses Tracker" project serves several purposes and offers several benefits to users. Here are some of the key uses that can be realized using a Personal Expenses Tracker:

a. Financial Awareness
b. Expense Monitoring
c. Budgeting and Goal Setting
d. Financial Insights
e. Expense Organization
f. Enhanced Financial Control

Overall, the use of a Personal Expenses Tracker can lead to improved financial management, increased savings, reduced debt, and better overall financial well-being. It promotes financial awareness, control, and discipline, helping individuals achieve their financial objectives and live a more financially secure life.

2. **Literature survey**

2.1 **Existing Problem**

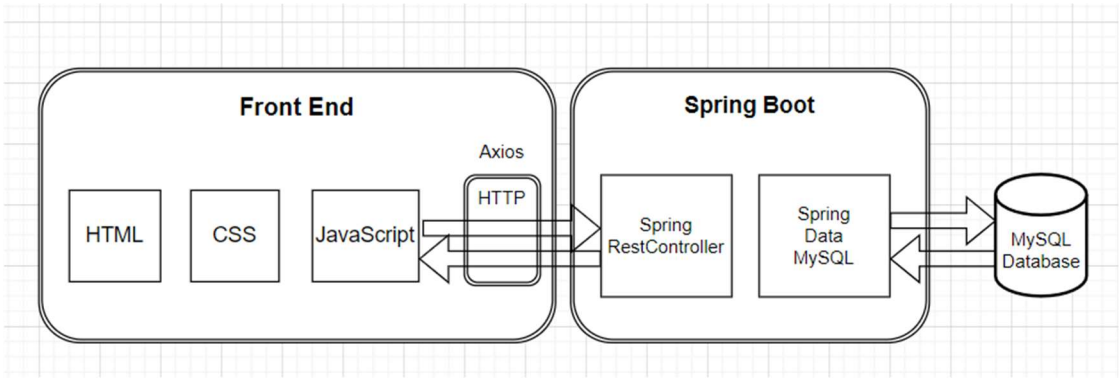
One of the primary challenges is the need for users to manually enter their expenses and income. This can be time-consuming and prone to errors, leading to incomplete or inaccurate financial records. Personal Expenses Tracker applications can sometimes have complex user interfaces, making it difficult for users, especially those who are less tech-savvy, to navigate and understand the features. Some Personal Expenses Trackers may have a predefined set of expense categories that may not align with users' specific needs or preferences.

2.2 **Proposed Solution**

Integration with financial institutions or receipt-scanning technology can automate data entry. By connecting to users' bank accounts or credit cards, transactions can be imported automatically. A user friendly interface with intuitive design and clear instructions can help simplify the user experience. Allowing users to create and modify categories based on their unique expenses ensures that the tracker remains relevant and adaptable to different users' requirements.

3. **Theoretical analysis**

3.1 Block Diagram



3.2 Hardware and Software Requirements

1. Java Development Kit (JDK): JDK is required to compile and run Java applications, providing the necessary tools and libraries. Download and install the latest JDK version from Oracle's website.

2. Spring Boot: Spring Boot simplifies Java application development by providing predefined configurations, automatic dependency management, and a streamlined development experience. Use Spring Initializr or Spring Tools for your IDE to create a Spring Boot project.

3. MySQL Database: MySQL is a popular relational database management system. Install MySQL Community Server and optionally MySQL Workbench, a graphical tool for managing MySQL databases.

4. MySQL Connector/J: MySQL Connector/J is the official JDBC driver for connecting Java applications to MySQL databases. Include this dependency in your project to enable connectivity and interaction with MySQL.

4. Experimental Investigation

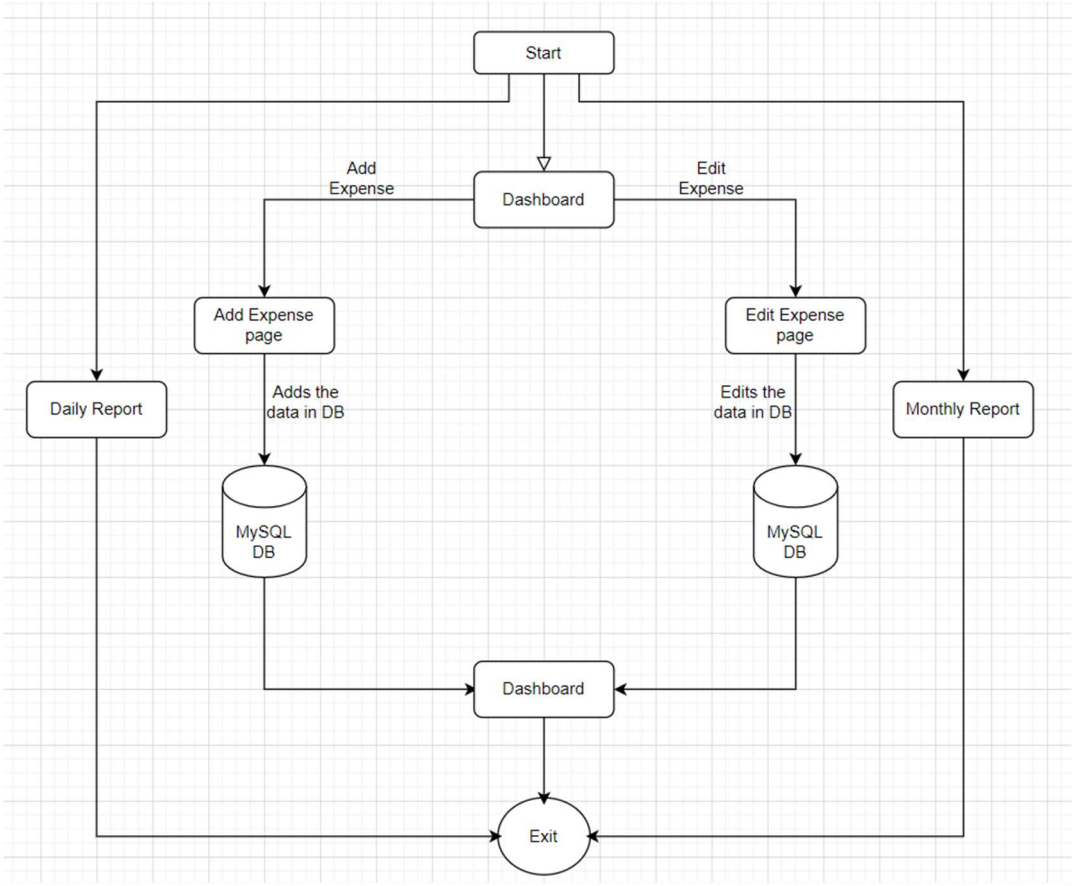
During the development of the "Personal Expenses Tracker" application, several experimental investigations were conducted to improve its functionality, usability, and overall effectiveness:

1. User Interface (UI) Testing: Conduct usability testing with a group of potential users to evaluate the application's user interface. Observe how users interact with the application, identify pain points, and gather feedback on the design, layout, and intuitiveness of the interface. This will help refine the UI and enhance user experience.

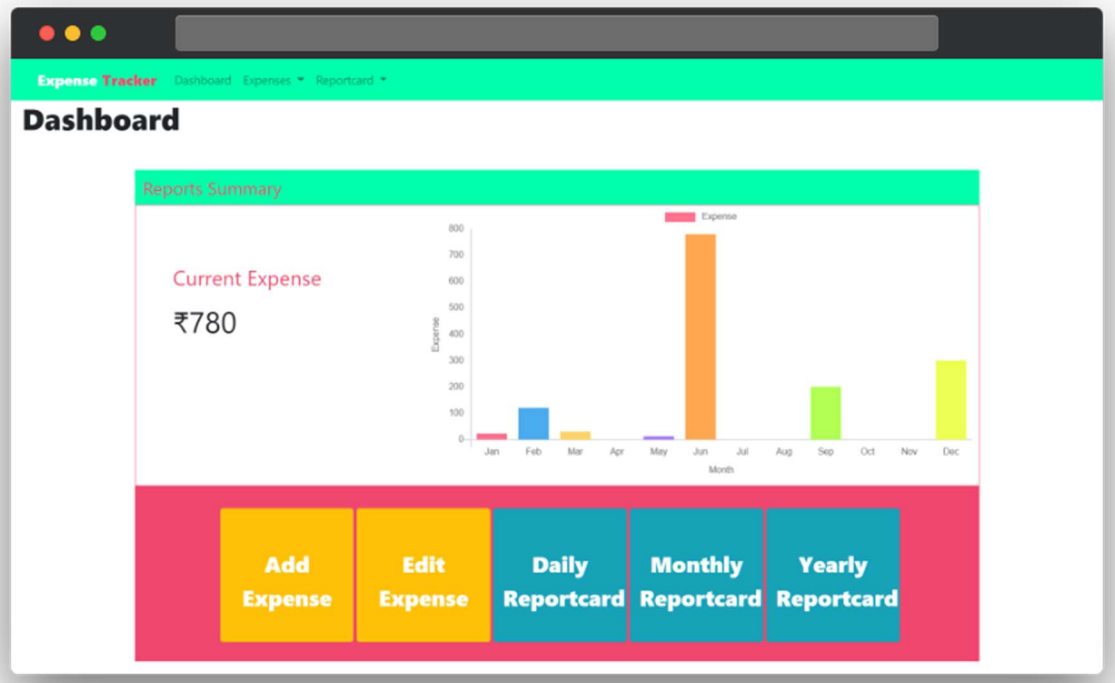
2. Data Entry Efficiency: Perform experiments to compare different data entry methods, such as manual entry, receipt scanning, or automatic synchronization with bank accounts. Measure the time required for each method, accuracy of data, and user preferences to determine the most efficient and user-friendly data entry approach.

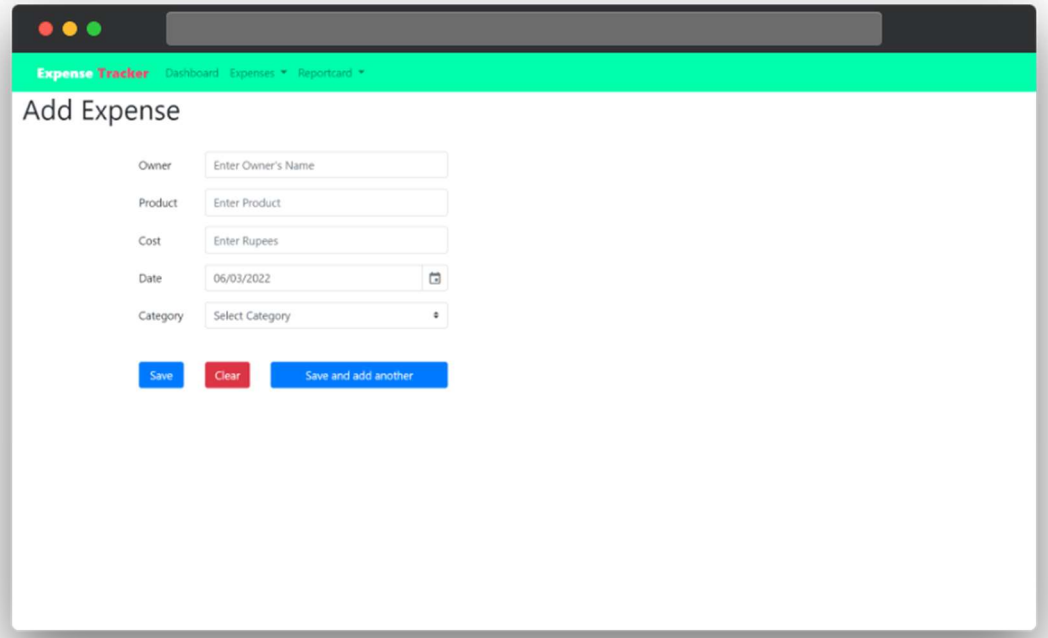
3. Personalized Insights: Explore different methods of providing personalized financial insights and recommendations to users based on their spending patterns. Measure user satisfaction and the effectiveness of these insights in helping users achieve their financial goals.

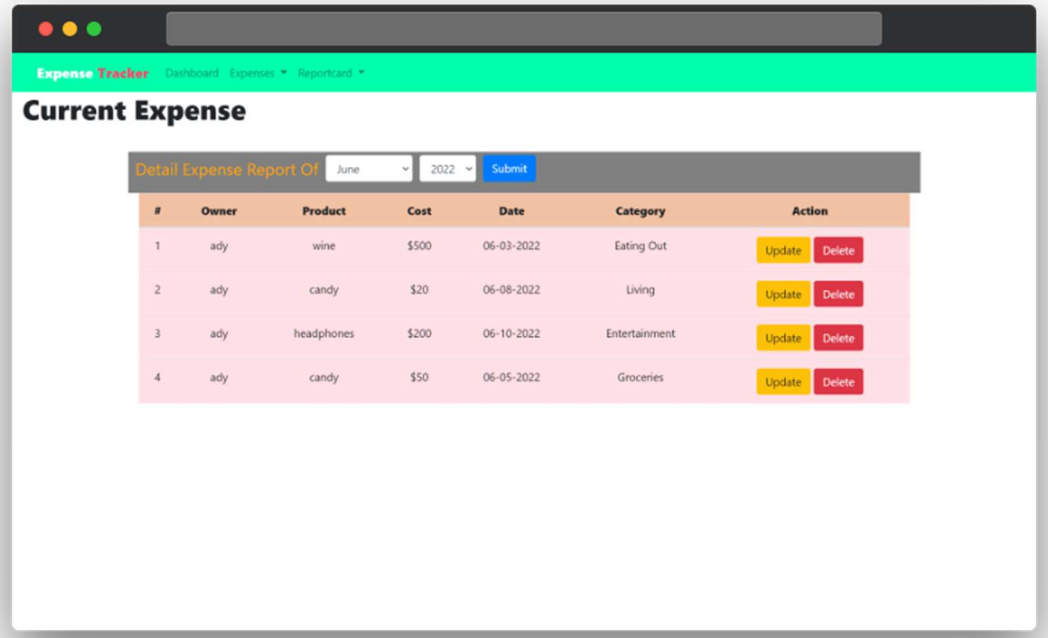
5. Flowchart


Start
Add Expense
Dashboard
Edit Expense
Add Expense page
Edit Expense page
Daily Report
Adds the data in DB
Edits the data in DB
Monthly Report
MySQL DB
MySQL DB
Dashboard
Exit

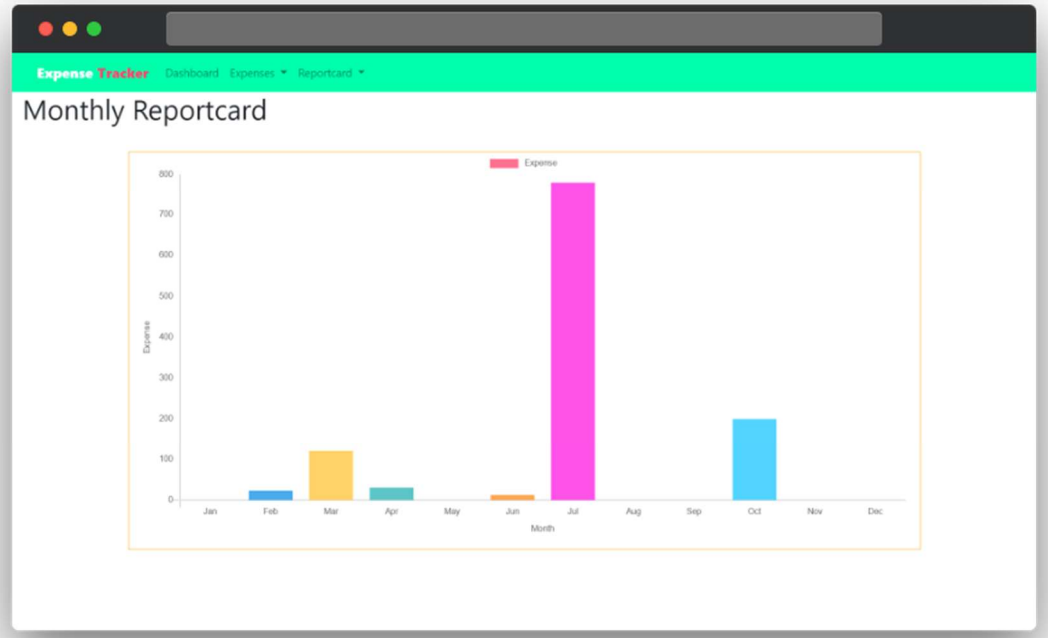
6. **Result**

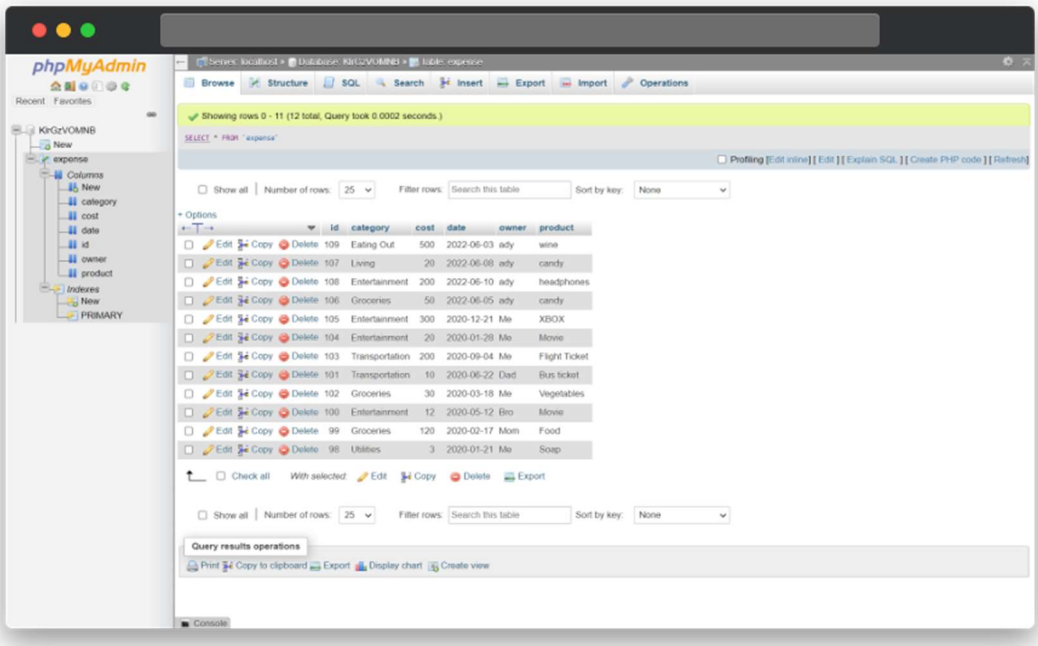

Expense Tracker
Dashboard
Expenses
Reportcard
Dashboard
Reports Summary
Current Expense
₹780
Expense
Add Expense
Edit Expense
Daily Reportcard
Monthly Reportcard
Yearly Reportcard


Expense Tracker
Dashboard
Expenses
Reportcard
Add Expense
Owner
Enter Owner's Name
Product
Enter Product
Cost
Enter Rupees
Date
06/03/2022
Category
Select Category
Save
Clear
Save and add another


Expense Tracker
Dashboard
Expenses
Reportcard
Current Expense
Detail Expense Report Of
June
2022
Submit
Owner Product Cost Date Category Action
1 ady wine $500 06-03-2022 Eating Out Update Delete
2 ady candy $20 06-08-2022 Living Update Delete
3 ady headphones $200 06-10-2022 Entertainment Update Delete
4 ady candy $50 06-05-2022 Groceries Update Delete


Expense Tracker
Dashboard
Expenses
Reportcard
Daily Reportcard
Reportcard of the month
June
2022
Submit
Expense
Expense
Days


Expense Tracker
Dashboard
Expenses
Reportcard
Monthly Reportcard
Expense
Expense
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Month



7. Advantages and disadvantages

Advantages:

Financial Awareness: Personal Expenses Tracker applications promote financial awareness by allowing users to track and monitor their income and expenses. This helps users develop a better understanding of their spending habits and make informed financial decisions.

Expense Categorization: These applications enable users to categorize their expenses, making it easier to analyse spending patterns and identify areas where they can reduce costs or save money.

Budgeting and Goal Setting: Personal Expenses Trackers often provide budgeting tools that allow users to set spending limits and track their progress towards financial goals. This helps users stay on track and manage their finances more effectively.

Data Visualization: Many Personal Expenses Tracker applications offer visual representations, such as graphs or charts, to present a summary of income and expenses. This visual representation helps users grasp their financial situation at a glance and identify trends and patterns.

Financial Insights: Personal Expenses Trackers provide insights into spending behaviors, allowing users to gain a deeper understanding of their financial habits.

This insight helps users make informed decisions and take necessary steps to improve their financial well-being.

Disadvantages:

Manual Data Entry: One of the main disadvantages is the need for manual data entry. Users have to manually input their expenses and income, which can be time-consuming and prone to errors. This may discourage some users from consistently using the application.

Learning Curve: Some Personal Expenses Tracker applications can have a learning curve, especially for users who are not familiar with financial management or technology. This may make it challenging for certain individuals to fully utilize the application's features.

Privacy and Security Concerns: Storing personal financial data within an application raises privacy and security concerns. Users need to ensure that the application they choose follows appropriate security measures to protect their sensitive information. Technical Limitations: Personal Expenses Tracker applications may have technical limitations or dependencies, such as requiring an internet connection or compatibility issues with certain devices or operating systems. This can limit accessibility and usability for some users.

Lack of Customization: Some applications may have limited options for customization, such as predefined expense categories or limited reporting capabilities. This can restrict users who prefer more flexibility or have specific financial tracking requirements.

8. Applications

Used to keep a track on personal expenses

9. Conclusion

In conclusion, Personal Expenses Tracker applications offer significant advantages in helping individuals manage and track their personal finances effectively. These applications promote financial awareness by allowing users to monitor their income and expenses, categorize their spending, and set budgets and goals. The visual representations and insights provided by these applications help users understand their financial behaviours and make informed decisions about their money.

However, there are also some disadvantages associated with Personal Expenses Tracker applications. Manual data entry can be time-consuming and error-prone, and there may be a learning curve for users unfamiliar with financial management or technology. Privacy and security concerns must be addressed to ensure the protection

of personal financial data. Additionally, some applications may have technical limitations or lack customization options, which may restrict their usefulness for certain users.

Overall, the benefits of Personal Expenses Tracker applications outweigh the drawbacks, as they provide users with valuable tools for financial management, budgeting, and goal setting. With the right application and proper utilization, individuals can gain control over their finances, improve their saving habits, and work towards their financial objectives. It is essential for users to consider their specific needs and preferences when selecting a Personal Expenses Tracker application to ensure it aligns with their financial goals and offers the necessary features for their financial management needs.

10 - Future scope

The future scope of "Personal Expenses Tracker Applications" is promising, with several potential areas of development and advancement. Here are some key aspects that could shape the future of these applications:

Integration with Financial Institutions: Personal Expenses Trackers can further enhance their capabilities by integrating with financial institutions, such as banks and credit card providers. This integration would allow for seamless and automatic transaction imports, real-time balance updates, and deeper insights into spending habits.

Artificial Intelligence and Machine Learning: By leveraging AI and machine learning algorithms, Personal Expenses Trackers can offer more accurate and automated expense categorization, personalized spending recommendations, and predictive analysis. These advancements can provide users with more comprehensive and actionable insights into their financial habits

Voice-Activated Interfaces: As voice assistants and smart speakers gain popularity, incorporating voice-activated interfaces into Personal Expenses Trackers can provide users with hands-free control and convenience. Users would be able to input expenses, check account balances, and receive financial summaries using voice commands.

Integration with Digital Wallets and Payment Apps: With the increasing adoption of digital wallets and payment apps, integrating Personal Expenses Trackers with these platforms can streamline expense tracking. Automatic transaction imports from digital wallets and payment apps would eliminate the need for manual entry and provide a more holistic view of an individual's financial transactions.

Enhanced Data Visualization: Future Personal Expenses Trackers could offer more advanced and interactive data visualization tools. This would enable users to gain deeper insights into

their financial data, such as trend analysis, predictive modelling, and comparative benchmarks, allowing for better financial decision-making.

11 – **Bibilography**

- "Personal Finance Management App Development: A Case Study" by Suvodeep Majumder and Poulami Das (2020): This case study explores the development of a personal finance management application, including expense tracking functionality. It discusses the development process, challenges, and lessons learned.

- "Expense Tracker: An Android Application" by Sharmistha Khan, Pinki Roy, and Md. Osman Goni Sarker (2017): This paper presents the development of an Android-based expense tracking application. It discusses the features, user interface, and functionalities implemented in the application.

- "Personal Finance Management Applications: A Comprehensive Review" by Aditya Pawar, Girish B. Gharde, and Sachin J. Deshmukh (2016): This paper provides an overview of various personal finance management applications, including expense trackers. It compares different features, user interfaces, and security measures of these applications.

- "Building Your Own Expense Tracker" by David Tran (2020): This article provides a step-by-step guide on building a simple expense tracker application using Python. It covers concepts such as data storage, expense categorization, and data visualization.

Appendix

Expense Repository:

```
package com.hemlata.app.dao;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

import com.hemlata.app.entity.ExpenseStorage;

public interface ExpenseRepository extends JpaRepository<ExpenseStorage,
Integer> {

    @Query(value="SELECT id, owner, product, cost, DATE_FORMAT(date, '%m-%d-
%Y') as 'date',"
```

```
            + " category FROM expense where month(date) = :MONTH AND
year(date) = :YEAR",
            nativeQuery = true)
    public List<ExpenseStorage> findByMonth(@Param("MONTH") String MONTH,
@Param("YEAR") String YEAR);

    @Query(value="SELECT id, owner, product, cost, DATE_FORMAT(date, '%m-%d-
%Y') as 'date'," 
            + " category FROM expense where owner = :OWNER AND product =
:PRODUCT AND cost = :COST AND"
            + " date = :DATE AND category = :CATEGORY",
            nativeQuery = true)
    public List<ExpenseStorage> expenseEntryIsPresent(@Param("OWNER") String
OWNER,@Param("PRODUCT")
    String PRODUCT,@Param("COST") String COST, @Param("DATE") String DATE,
@Param("CATEGORY") String CATEGORY);

    @Query(value="SELECT sum(cost) as 'cost', DATE_FORMAT(e.date, '%m-%d-%Y')
as 'date' FROM "
            + "expense as e where month(e.date) = :MONTH AND year(e.date) =
:YEAR GROUP BY date",
            nativeQuery = true)
    public List<ICostAndDateQuery> getCostByDateList(@Param("MONTH") String
MONTH, @Param("YEAR") String YEAR);

    @Query(value="SELECT sum(cost) as 'cost', month(date) as 'month' FROM
expense GROUP BY month",
            nativeQuery = true)
    public List<ICostAndMonthQuery> getCostByMonthList();

    @Query(value="SELECT sum(cost) as 'cost', year(date) as 'year' FROM
expense GROUP BY year",
            nativeQuery = true)
    public List<ICostAndYearQuery> getCostByYearList();
}
```

User Repository:

```
package com.hemlata.app.dao;

import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

import com.hemlata.app.entity.User;

@Repository("userRepository")
public interface UserRepository extends CrudRepository<User, String> {
```

```
    User findByEmailIdIgnoreCase(String emailId);
    @Query("select userid from User")
    public long[] ids();
}
```

Expense Storage:

```
package com.hemlata.app.entity;

import java.text.ParseException;

import java.text.SimpleDateFormat;
import java.util.Arrays;
import java.util.Date;
import java.util.List;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.Transient;
import javax.validation.constraints.Min;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;
/*
 * Class: This is the entity class which is linked with the database.
 */
@Entity
@Table(name="expense")
public class ExpenseStorage {

    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    @Column(name="id")
    private int id;

    @NotBlank(message="This field is mandatory.")
    @Column(name="owner")
    public String owner;

    @NotBlank(message="This field is mandatory.")
    @Column(name="product")
    public String product;

    @NotNull(message="This field is mandatory.")
```

```
    @Min(value=0, message="must be greater than or equal to zero")
    @Column(name="cost")
    public Integer cost;

    @NotBlank(message="This field is mandatory.")
    @Column(name="date")
    public String date;

    @Column(name="category")
    public String category;

    @Transient
    private String errorMsg;

    public ExpenseStorage() {
        errorMsg = "";
        SimpleDateFormat formatter = new SimpleDateFormat("MM/dd/yyyy");
        Date dateObj = new Date();
        this.date = formatter.format(dateObj);
    }
    public ExpenseStorage(@NotBlank(message = "This field is mandatory.")
String owner,
            @NotBlank(message = "This field is mandatory.") String product,
            @NotNull(message = "This field is mandatory.") @Min(value = 0,
message = "must be greater than or equal to zero") Integer cost,
            @NotBlank(message = "This field is mandatory.") String date,
String category) {
        this.owner = owner;
        this.product = product;
        this.cost = cost;
        this.date = date;
        this.category = category;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getOwner() {
        return owner;
    }
    public void setOwner(String owner) {
        this.owner = owner;
    }
    public String getProduct() {
        return product;
    }
```

```
    public void setProduct(String product) {
        this.product = product;
    }
    public Integer getCost() {
        return cost;
    }
    public void setCost(Integer cost) {
        this.cost = cost;
    }
    public String getDate() {
        return date;
    }
    public void setDate(String date) {
        this.date = date;
    }
    public String getCategory() {
        return category;
    }
    public void setCategory(String category) {
        this.category = category;
    }

    public String getErrorMsg() {
        return errorMsg;
    }

    public void setErrorMsg(String errorMsg) {
        this.errorMsg = errorMsg;
    }
    /*
     * Function: Convert the date in the given format.
     */
    public void setDateInSpecificFormat(String requiredFormat) {
        if(this.date != null) {
            List<String> dateFormats = Arrays.asList("MM/dd/yyyy", "yyyy-MM-
dd");
            for(String dateFormat: dateFormats) {
                try {
                    Date parsedDate = new
SimpleDateFormat(dateFormat).parse(this.date);
                    this.date = new
SimpleDateFormat(requiredFormat).format(parsedDate);
                } catch (ParseException e) {
                    System.out.println("Unable to parse the date = " +
this.date);
                }
            }
        }
```

```
    }
    @Override
    public String toString() {
        return "ExpenseStorage [id=" + id + ", owner=" + owner + ", product="
+ product + ", cost=" + cost + ", date="
                + date + ", category=" + category + "]";
    }
}
```