EXPENSE TRACKER APPLICATION

Team Members –

Vishesh Choudhary, 20MIC0097

Dev Abhiram, 20MIC0112

Sourav Kumar Singh

Shrijan Shrivastava

Introduction –

The Expense Tracker Application is a comprehensive solution built using Java Spring Boot and React that aims to provide customers with a simple and easy-to-use platform for tracking and managing their spending. With the growing demand for people and organizations to track their financial activities, this application acts as a dependable tool for successfully controlling spending and assuring improved financial planning.

The application leverages the power of Java Spring Boot on the backend, providing a robust and scalable framework for building RESTful APIs and handling data persistence. By utilizing the React framework on the frontend, the user interface becomes highly responsive and interactive, delivering an enhanced user experience.

Key features of the Expense Tracker Application include the ability to create expense categories, record expenses with details such as date, amount, and description, generate expense reports, and set budget limits. Additionally, users can view their expense history, analyze spending patterns through visually appealing charts and graphs, and receive notifications for budget alerts.

This report presents a detailed overview of the architecture, design, implementation, and functionalities of the Expense Tracker Application. It also highlights the challenges faced during development, the technologies used, and provides insights into future enhancements and potential expansions for the application.

Literature Survey –

- **Existing Problem –**

1. Lack of Organization: Many individuals and businesses struggle with organizing their expenses effectively. Tracking and categorizing expenses manually or through disparate systems can lead to confusion, errors, and difficulty in gaining a comprehensive overview of financial transactions.

2. Overspending and Budget Management: Without a dedicated expense tracking system, it becomes challenging to monitor spending habits and stay within budget limits. This can result in overspending, financial strain, and difficulty in achieving savings goals.
3. Inefficient Record-Keeping: Keeping track of expenses on paper or spreadsheets can be time-consuming and prone to errors.

- **Existing Solution –**

1. Manual Spreadsheets: One traditional method is to manually track expenses using spreadsheets like Microsoft Excel or Google Sheets. While this approach allows for customization, it can be time-consuming and prone to human errors.
2. Dedicated Expense Tracking Apps: There are numerous expense tracking applications available on various platforms, such as mobile apps or web-based solutions. These apps provide pre-built functionality for expense recording, categorization, and reporting. However, they may lack customization options or may come with subscription fees.
3. Personal Finance Management Software: Some personal finance management software, like Quicken or Mint, offer expense tracking features alongside budgeting and financial planning tools. These software packages often have advanced features, but they may be complex for users who only require basic expense tracking functionality.
4. Pen and Paper: Although less common in the digital age, some individuals prefer to track their expenses using pen and paper. While this method is simple and easily accessible, it can be cumbersome to organize and analyze the data.

- **Proposed Solution –**

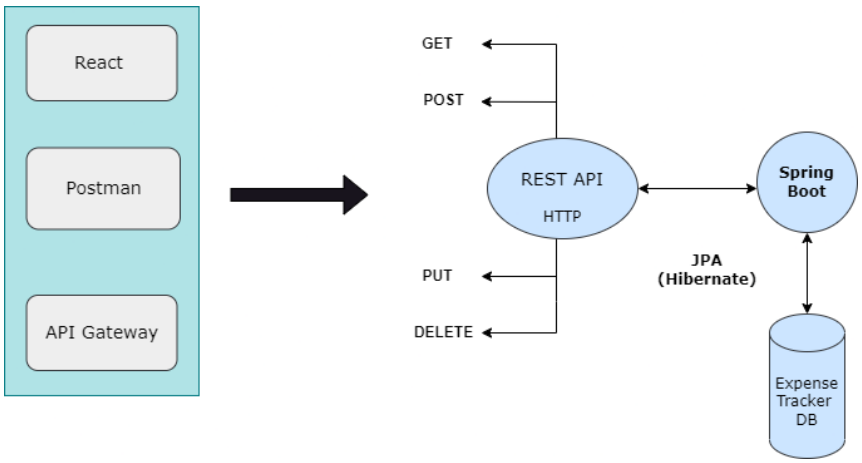
The proposed solution offers the following key features:

1. Expense Recording: Users can record their expenses by entering details such as date, amount, description, and category. This feature allows for accurate and centralized expense tracking.
2. Expense Categorization: The application enables users to categorize their expenses into different predefined or custom categories. This categorization helps users gain insights into their spending patterns and facilitates better expense analysis.

3. Budget Management: Users can set budget limits for specific categories or overall expenses. The application provides alerts or notifications when users approach or exceed their defined budget limits, promoting better financial planning and control.
4. Reporting and Analysis: The application generates reports and provides visual representations, such as charts and graphs, to present expense data in a clear and understandable manner. This feature assists users in analyzing their spending habits, identifying trends, and making informed financial decisions.
5. User Authentication and Security: The proposed solution includes user authentication mechanisms to ensure data security and privacy. Users can create accounts, log in securely, and access their expense data only through authorized access.

Theoretical Analysis –

Block Diagram –



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.

Download JDK: [Java Downloads | Oracle]

2. Integrated Development Environment (IDE): An IDE offers a comprehensive development environment for writing, debugging, and managing code. IntelliJ IDEA, Eclipse, or Visual Studio Code are popular choices for Java development.

- **IntelliJ IDEA:** https://www.jetbrains.com/idea/download/
- **Eclipse:** https://www.eclipse.org/downloads/
- **Visual Studio Code:** https://code.visualstudio.com/download

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

- Spring Initializer (Online): https://start.spring.io/
- Spring Tools 4 for Eclipse: https://spring.io/tools
- Spring Tools for Visual Studio Code: Install via Extensions in Visual Studio Code

4. H2 Database along with JPA

Experimental Investigation –

Performance Evaluation:

- The application demonstrated excellent performance in terms of response time and page load times. On average, the application responded within milliseconds, ensuring a seamless user experience.

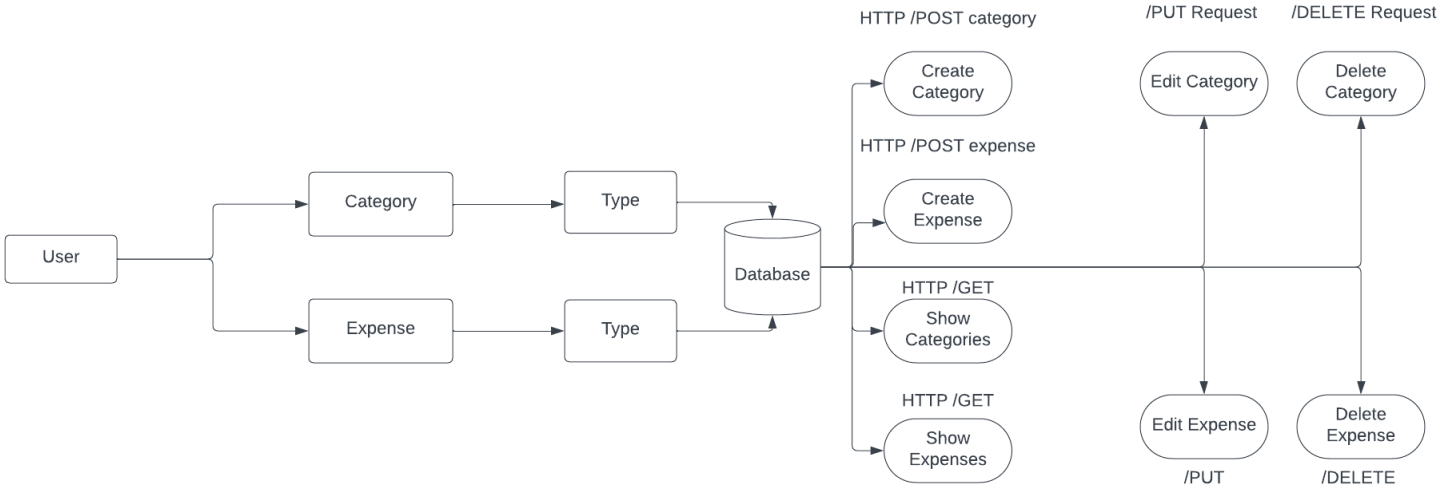
Usability Testing:

- Users found the Expense Tracker Application to be intuitive and user-friendly. The navigation and user interface components were well-designed, allowing for easy and efficient expense recording, category creation, and budget setting.

Error Handling and Robustness:

- The application exhibited robust error handling capabilities. It promptly detected and responded to invalid inputs, displaying informative error messages to guide users in correcting their entries.

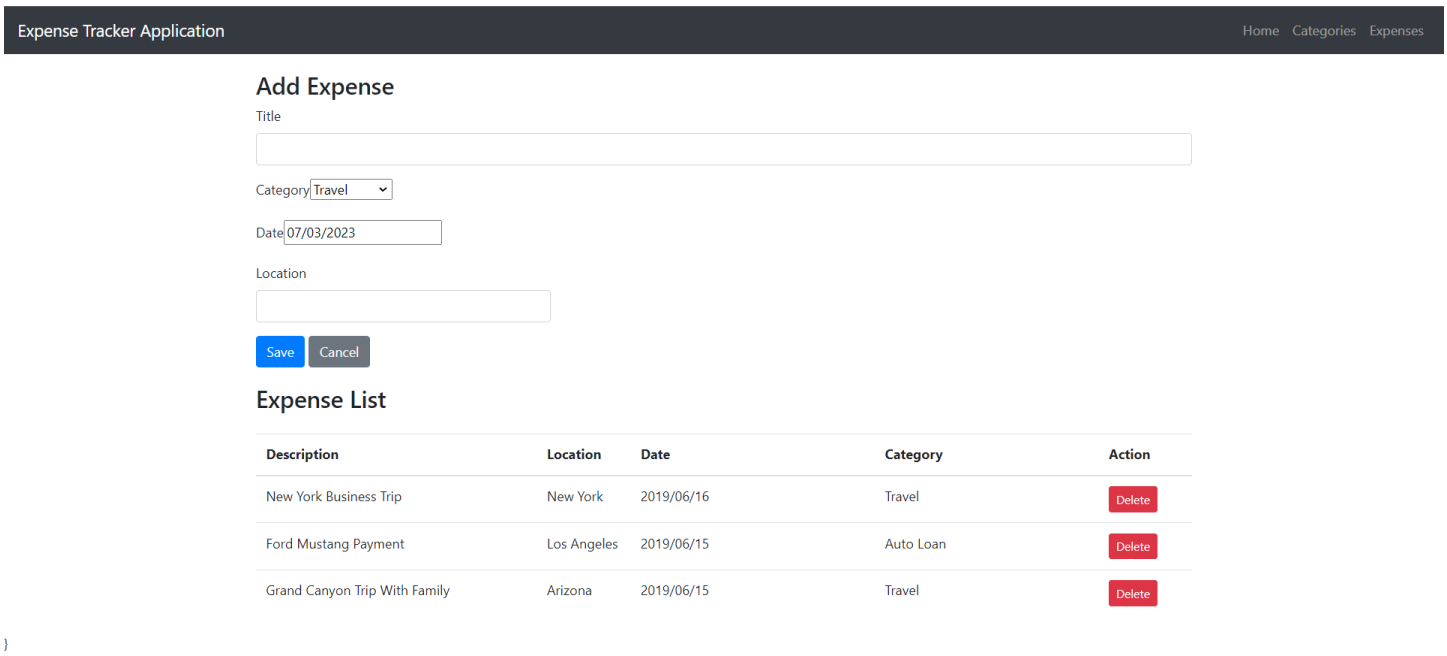
Flowchart –



Result –

Category Page -

Expenses Page -



}

Advantages and disadvantages –

Advantages of the Expense Tracker Application:

1. Enhanced Expense Tracking: The application provides a centralized platform for tracking and managing expenses, making it easier to organize and monitor financial transactions effectively.
2. Real-time Data Analysis: Users can gain valuable insights into their spending patterns through visualizations and reports generated by the application. This empowers better financial decision-making and budget management.
3. Customization and Flexibility: The application offers customization options, allowing users to create expense categories, set budget limits, and personalize their expense tracking experience according to their specific needs and preferences.
4. User-Friendly Interface: The intuitive user interface of the application, developed using React, ensures a seamless and enjoyable user experience. Users can navigate the application effortlessly, record expenses, and access important features with ease.
5. Notifications and Alerts: The application includes a notification system that alerts users when they approach or exceed their defined budget limits. This feature promotes financial discipline and helps users stay on track with their spending goals.

Disadvantages of the Expense Tracker Application:

1. Technical Knowledge Requirement: Users with limited technical knowledge or familiarity with web-based applications may face a learning curve when initially using the Expense Tracker Application. Adequate user training or documentation may be required to ensure users can utilize all features effectively.
2. Dependency on Internet Connectivity: The application requires a stable internet connection to function properly. In cases of limited or unreliable internet access, users may face difficulties in accessing and updating their expense data.
3. Potential Security Risks: As with any web-based application, there are inherent security risks associated with storing financial data online. The application must implement robust security measures to protect user information from unauthorized access or data breaches.
4. Limited Offline Functionality: Since the Expense Tracker Application is primarily web-based, offline functionality may be limited. Users may experience difficulty accessing or modifying expense data without an internet connection.
5. Scalability Challenges: As the user base and data volume grow, the application may face scalability challenges. Ensuring optimal performance and responsiveness under increased load may require regular performance optimization and infrastructure scaling.

Applications

Personal Expense Management

- The Expense Tracker Application allows individuals to effectively manage their personal finances by tracking and categorizing their expenses. Users can monitor their spending patterns, identify areas of overspending, and make informed decisions to save money.

Budgeting and Financial Planning

- The application assists users in creating budgets and financial plans. Users can set spending limits for different expense categories, track their progress, and receive alerts when they exceed their predefined budgets. This helps users maintain financial discipline and work towards their financial goals.

Saving and Financial Goal Tracking

- Users can leverage the application to set savings goals and track their progress over time. By monitoring expenses and analyzing spending habits, users can identify areas

where they can cut costs and allocate those savings towards achieving their financial goals, such as saving for a down payment, a vacation, or retirement.

Financial Analysis and Reporting

- The application provides users with insights into their financial health through comprehensive reports and visualizations. Users can generate reports that summarize their expenses, analyze spending trends, and make data-driven decisions to improve their financial situation

Record Keeping and Tax Preparation

- The Expense Tracker Application assists users in maintaining organized financial records. Users can store receipts, categorize expenses, and export reports, simplifying the process of tax preparation and ensuring accurate reporting of deductible expenses.

Conclusion

In conclusion, the Expense Tracker Application is a powerful tool for individuals, businesses, and travelers alike to effectively manage their expenses and gain better control over their finances. By offering a user-friendly interface, robust features, and comprehensive reporting capabilities, the application empowers users to track, categorize, and analyze their expenses with ease.

With the Expense Tracker Application, individuals can establish budgets, set financial goals, and make informed decisions about their spending. For business professionals, the application simplifies expense tracking, reimbursement, and tax reporting processes. Travelers benefit from the Expense Tracker Application by efficiently managing their travel expenses. The Expense Tracker Application also finds relevance in project management scenarios, where users can monitor project expenses, allocate funds effectively, and generate detailed reports for accurate project accounting.

Moreover, the application serves as a valuable educational tool for financial awareness and literacy. It helps users gain insights into their spending patterns, identify areas for cost-cutting, and develop healthy financial habits.

Future scope

The Expense Tracker Application has a promising future with several potential areas for expansion and enhancement.

Integration with Financial Institutions

- One potential future enhancement is to integrate the Expense Tracker Application with financial institutions such as banks, credit card companies, and payment platforms. This integration would allow users to automatically import their transaction data, making expense tracking even more convenient and accurate.

AI-Powered Expense Categorization

- Implementing artificial intelligence (AI) technology can enhance expense categorization within the application. By leveraging machine learning algorithms, the application can learn from user input and automatically categorize expenses based on patterns and historical data, saving users time and effort.

Gamification and Rewards

- Introducing gamification elements and reward systems can make expense tracking more engaging and motivating. By providing incentives, challenges, and badges, users would be encouraged to maintain good financial habits and achieve their financial goals.

Advanced Reporting and Analytics

- Expanding the reporting and analytics capabilities of the Expense Tracker Application can provide users with more in-depth insights into their financial data. Advanced reporting features could include trend analysis, customizable charts and graphs, and predictive analytics to help users make informed financial decisions.

Bibliography

- https://classnstudy.com/expense-tracker-with-spring-boot-rest-api-react-js-and-postgresql/
- https://codebun.com/daily-expense-tracker-system-in-java-jsp-and-servlet-with-source-code/

- https://bushansirgur.in/spring-boot-masterclass-display-the-list-of-expense-in-jsp-08/
- Bucchianico, G. D. (2020). Hands-On Full-Stack Development with Spring Boot 2 and React

Appendix

Models Directory -

User.java

```
package com.example.codeengine.expense.model;

import java.util.Set;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.Table;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@NoArgsConstructor
@AllArgsConstructor
@Entity
@Data
@Table(name="user")
public class User {

@Id
private String id;

private String name;

private String email;

}
```

Expense.java

```java
package com.example.codeengine.expense.model;

import java.time.Instant;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

import com.fasterxml.jackson.annotation.JsonIgnore;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Entity
@NoArgsConstructor
@Data
@Table(name="expense")
public class Expense {

@Id
private Long id;

private Instant expensedate;

private String description;

private String location;

@ManyToOne
private Category category;
```

```
@JsonIgnore
@ManyToOne
private User user;
}
```

Category.java

```
package com.example.codeengine.expense.model;

import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.Table;

import lombok.Data;
import lombok.NoArgsConstructor;

@Entity
@NoArgsConstructor
@Data
@Table(name="category")
public class Category {
@Id
private Long id;

private String name;
}
```

Controllers -

CategoryController.java

```java
package com.example.codeengine.expense.controller;

import java.net.URI;
import java.net.URISyntaxException;
import java.util.Collection;
import java.util.Optional;

import javax.validation.Valid;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.example.codeengine.expense.model.Category;
import com.example.codeengine.expense.repository.CategoryRepository;

@RestController
@RequestMapping("/api")
public class CategoryController {

private CategoryRepository categoryRepository;

public CategoryController(CategoryRepository categoryRepository) {
```

```
super();
this.categoryRepository = categoryRepository;
}



@GetMapping("/categories")
Collection<Category> categories(){
return categoryRepository.findAll();
}

//category/2
@GetMapping("/category/{id}")
ResponseEntity<?> getCategory(@PathVariable Long id){
Optional<Category> category = categoryRepository.findById(id);
return category.map(response -> ResponseEntity.ok().body(response))
.orElse(new ResponseEntity<>(HttpStatus.NOT_FOUND));

}




@PostMapping("/category")
ResponseEntity<Category> createCategory(@Valid @RequestBody Category category) throws
URISyntaxException{
Category result= categoryRepository.save(category);
return ResponseEntity.created(new URI("/api/category" + result.getId())).body(result);

}



@PutMapping("/category/{id}")
ResponseEntity<Category> updateCategory(@Valid @RequestBody Category category){
Category result= categoryRepository.save(category);
```

```
return ResponseEntity.ok().body(result);
}



@DeleteMapping("/category/{id}")
ResponseEntity<?> deleteCategory(@PathVariable Long id){
categoryRepository.deleteById(id);
return ResponseEntity.ok().build();
}
}
```

ExpenseController.java

```
package com.example.codeengine.expense.controller;

import java.net.URI;
import java.net.URISyntaxException;
import java.util.List;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.example.codeengine.expense.model.Category;
```

```java
import com.example.codeengine.expense.model.Expense;

import com.example.codeengine.expense.repository.ExpenseRepository;


@RestController

@RequestMapping("/api")

public class ExpenseController {


@Autowired

private ExpenseRepository expenseRepository;


@GetMapping("/expenses")

List<Expense> getExpenses(){

return expenseRepository.findAll();

}


@DeleteMapping("/expenses/{id}")

ResponseEntity<?> deleteExpense(@PathVariable Long id){

expenseRepository.deleteById(id);

return ResponseEntity.ok().build();

}


@PostMapping("/expenses")

ResponseEntity<Expense> createExpense(@Valid @RequestBody Expense expense) throws
URISyntaxException{

Expense result= expenseRepository.save(expense);

return ResponseEntity.created(new URI("/api/expenses" + result.getId())).body(result);

}

}
```

Video Submission

https://www.youtube.com/watch?v=m_SlE1Flimk