**Expense Tracker — Complete Java Project**

This document contains a complete Java project for \*\*Expense Tracker with Monthly Analytics\*\* using Java, JavaFX, SQLite and JFreeChart. Files are separated with `=== FILE: <path> ===` markers. Copy each section into the corresponding file in a Maven project.

FILE: pom.xml

xml

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>expense-tracker</artifactId>

<version>1.0.0</version>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<maven.compiler.source>17</maven.compiler.source>

<maven.compiler.target>17</maven.compiler.target>

<javafx.version>20</javafx.version>

<jfree.version>2.1.0</jfree.version>

</properties>

<dependencies>

<!-- SQLite JDBC -->

<dependency>

<groupId>org.xerial</groupId>

<artifactId>sqlite-jdbc</artifactId>

<version>3.41.2.1</version>

</dependency>

<!-- JavaFX modules -->

<dependency>

<groupId>org.openjfx</groupId>

<artifactId>javafx-controls</artifactId>

<version>${javafx.version}</version>

</dependency>

<dependency>

<groupId>org.openjfx</groupId>

<artifactId>javafx-fxml</artifactId>

<version>${javafx.version}</version>

</dependency>

<!-- JFreeChart -->

<dependency>

<groupId>org.jfree</groupId>

<artifactId>jfreechart</artifactId>

<version>${jfree.version}</version>

</dependency>

<!-- For logging (optional) -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-simple</artifactId>

<version>2.0.9</version>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Shade to create fat jar -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-shade-plugin</artifactId>

<version>3.4.1</version>

<executions>

<execution>

<phase>package</phase>

<goals><goal>shade</goal></goals>

<configuration>

<transformers>

<transformer implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">

<mainClass>com.example.expensetracker.MainApp</mainClass>

</transformer>

</transformers>

</configuration>

</execution>

</executions>

</plugin>

<!-- Compiler plugin -->

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.11.0</version>

<configuration>

<release>17</release>

</configuration>

</plugin>

</plugins>

</build>

</project>

**\=== FILE: src/main/resources/main.fxml ===**

**xml**

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.\*?>

<?import javafx.scene.layout.\*?>

<VBox xmlns="http://javafx.com/javafx" xmlns:fx="http://javafx.com/fxml"

fx:controller="com.example.expensetracker.ui.MainController" spacing="8" padding="8">

<HBox spacing="10">

<VBox spacing="8" prefWidth="280">

<Label text="Add Expense" />

<TextField fx:id="amountField" promptText="Amount" />

<DatePicker fx:id="datePicker" />

<ComboBox fx:id="categoryCombo" promptText="Category" />

<TextField fx:id="descField" promptText="Description" />

<Button text="Add Expense" onAction="#onAddExpense" />

<Separator />

<Label text="Categories" />

<ListView fx:id="categoryList" prefHeight="200"/>

<HBox spacing="6">

<TextField fx:id="newCategoryField" promptText="New category" />

<Button text="Add" onAction="#onAddCategory" />

</HBox>

</VBox>

<VBox spacing="8" HBox.hgrow="ALWAYS">

<Label text="Expenses" />

<TableView fx:id="expenseTable" prefHeight="400">

<columns>

<TableColumn text="Date" fx:id="colDate" />

<TableColumn text="Category" fx:id="colCategory" />

<TableColumn text="Description" fx:id="colDesc" />

<TableColumn text="Amount" fx:id="colAmount" />

</columns>

</TableView>

<HBox spacing="10">

<Button text="Refresh" onAction="#onRefresh" />

<Button text="Export CSV" onAction="#onExportCsv" />

</HBox>

</VBox>

<VBox spacing="8" prefWidth="400">

<Label text="Charts" />

<StackPane fx:id="chartPane" prefHeight="400" />

</VBox>

</HBox>

</VBox>

**\=== FILE: src/main/java/com/example/expensetracker/MainApp.java ===**

**java**

package com.example.expensetracker;

import javafx.application.Application;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.image.Image;

import javafx.stage.Stage;

public class MainApp extends Application {

@Override

public void start(Stage primaryStage) throws Exception {

Parent root = FXMLLoader.load(getClass().getResource("/main.fxml"));

primaryStage.setTitle("Expense Tracker");

primaryStage.setScene(new Scene(root, 1100, 600));

primaryStage.getIcons().add(new Image(MainApp.class.getResourceAsStream("/icon.png"))); // optional

primaryStage.show();

}

public static void main(String[] args) {

launch(args);

}

}

**\=== FILE: src/main/java/com/example/expensetracker/db/DB.java ===**

**java**

package com.example.expensetracker.db;

import java.sql.\*;

public class DB {

private static final String DB\_URL = "jdbc:sqlite:expenses.db";

private static Connection conn;

public static synchronized Connection getConnection() throws SQLException {

if (conn == null || conn.isClosed()) {

conn = DriverManager.getConnection(DB\_URL);

try (Statement st = conn.createStatement()) {

st.execute("PRAGMA foreign\_keys = ON;");

}

initSchema(conn);

}

return conn;

}

private static void initSchema(Connection c) throws SQLException {

try (Statement st = c.createStatement()) {

st.execute("""

CREATE TABLE IF NOT EXISTS categories (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL UNIQUE,

color TEXT

);

""");

st.execute("""

CREATE TABLE IF NOT EXISTS expenses (

id INTEGER PRIMARY KEY AUTOINCREMENT,

category\_id INTEGER,

amount REAL NOT NULL CHECK(amount >= 0),

description TEXT,

expense\_date TEXT NOT NULL,

created\_at TEXT DEFAULT (datetime('now')),

FOREIGN KEY (category\_id) REFERENCES categories(id) ON DELETE SET NULL

);

""");

st.execute("CREATE INDEX IF NOT EXISTS idx\_expenses\_date ON expenses(expense\_date);");

st.execute("CREATE INDEX IF NOT EXISTS idx\_expenses\_category ON expenses(category\_id);");

// Seed some default categories if empty

try (ResultSet rs = st.executeQuery("SELECT COUNT(\*) AS cnt FROM categories")) {

if (rs.next() && rs.getInt("cnt") == 0) {

st.execute("INSERT INTO categories(name, color) VALUES('Food', '#FF9800')");

st.execute("INSERT INTO categories(name, color) VALUES('Transport', '#2196F3')");

st.execute("INSERT INTO categories(name, color) VALUES('Rent', '#4CAF50')");

st.execute("INSERT INTO categories(name, color) VALUES('Entertainment', '#9C27B0')");

}

}

}

}

}

**\=== FILE: src/main/java/com/example/expensetracker/model/Category.java ===**

**java**

package com.example.expensetracker.model;

public class Category {

private int id;

private String name;

private String color;

public Category(int id, String name, String color) {

this.id = id;

this.name = name;

this.color = color;

}

public int getId() { return id; }

public String getName() { return name; }

public String getColor() { return color; }

@Override

public String toString() { return name; }

}

**\=== FILE: src/main/java/com/example/expensetracker/model/Expense.java ===**

**java**

package com.example.expensetracker.model;

public class Expense {

private int id;

private String date; // YYYY-MM-DD

private String categoryName;

private String description;

private double amount;

public Expense(int id, String date, String categoryName, String description, double amount) {

this.id = id;

this.date = date;

this.categoryName = categoryName;

this.description = description;

this.amount = amount;

}

public int getId() { return id; }

public String getDate() { return date; }

public String getCategoryName() { return categoryName; }

public String getDescription() { return description; }

public double getAmount() { return amount; }

}

**\=== FILE: src/main/java/com/example/expensetracker/dao/CategoryDao.java ===**

**java**

package com.example.expensetracker.dao;

import com.example.expensetracker.db.DB;

import com.example.expensetracker.model.Category;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

public class CategoryDao {

public List<Category> listAll() throws SQLException {

String sql = "SELECT id, name, color FROM categories ORDER BY name";

try (Connection c = DB.getConnection(); PreparedStatement ps = c.prepareStatement(sql); ResultSet rs = ps.executeQuery()) {

List<Category> list = new ArrayList<>();

while (rs.next()) {

list.add(new Category(rs.getInt("id"), rs.getString("name"), rs.getString("color")));

}

return list;

}

}

public int create(String name, String color) throws SQLException {

String sql = "INSERT INTO categories(name, color) VALUES(?, ?)";

try (Connection c = DB.getConnection(); PreparedStatement ps = c.prepareStatement(sql, Statement.RETURN\_GENERATED\_KEYS)) {

ps.setString(1, name);

ps.setString(2, color);

ps.executeUpdate();

try (ResultSet keys = ps.getGeneratedKeys()) {

if (keys.next()) return keys.getInt(1);

}

}

return -1;

}

}

**\=== FILE: src/main/java/com/example/expensetracker/dao/ExpenseDao.java ===**

**java**

package com.example.expensetracker.dao;

import com.example.expensetracker.db.DB;

import com.example.expensetracker.model.Expense;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

public class ExpenseDao {

public void addExpense(Integer categoryId, double amount, String description, String dateIso) throws SQLException {

String sql = "INSERT INTO expenses(category\_id, amount, description, expense\_date) VALUES(?,?,?,?)";

try (Connection c = DB.getConnection(); PreparedStatement ps = c.prepareStatement(sql)) {

if (categoryId == null) ps.setNull(1, Types.INTEGER); else ps.setInt(1, categoryId);

ps.setDouble(2, amount);

ps.setString(3, description);

ps.setString(4, dateIso);

ps.executeUpdate();

}

}

public List<Expense> listByMonth(int year, int month) throws SQLException {

String monthStr = String.format("%04d-%02d", year, month);

String sql = "SELECT e.id, e.amount, e.description, e.expense\_date, c.name AS category\_name " +

"FROM expenses e LEFT JOIN categories c ON e.category\_id = c.id " +

"WHERE strftime('%Y-%m', expense\_date) = ? ORDER BY expense\_date DESC";

try (Connection c = DB.getConnection(); PreparedStatement ps = c.prepareStatement(sql)) {

ps.setString(1, monthStr);

try (ResultSet rs = ps.executeQuery()) {

List<Expense> out = new ArrayList<>();

while (rs.next()) {

out.add(new Expense(rs.getInt("id"), rs.getString("expense\_date"), rs.getString("category\_name"), rs.getString("description"), rs.getDouble("amount")));

}

return out;

}

}

}

}

**\=== FILE: src/main/java/com/example/expensetracker/ui/MainController.java ===**

**java**

package com.example.expensetracker.ui;

import com.example.expensetracker.dao.CategoryDao;

import com.example.expensetracker.dao.ExpenseDao;

import com.example.expensetracker.model.Category;

import com.example.expensetracker.model.Expense;

import com.example.expensetracker.util.CsvExporter;

import javafx.collections.FXCollections;

import javafx.fxml.FXML;

import javafx.scene.control.\*;

import javafx.scene.layout.StackPane;

import java.time.LocalDate;

import java.util.List;

public class MainController {

@FXML private TextField amountField;

@FXML private DatePicker datePicker;

@FXML private ComboBox<Category> categoryCombo;

@FXML private TextField descField;

@FXML private ListView<Category> categoryList;

@FXML private TextField newCategoryField;

@FXML private TableView<Expense> expenseTable;

@FXML private TableColumn<Expense, String> colDate;

@FXML private TableColumn<Expense, String> colCategory;

@FXML private TableColumn<Expense, String> colDesc;

@FXML private TableColumn<Expense, Double> colAmount;

@FXML private StackPane chartPane;

private CategoryDao categoryDao = new CategoryDao();

private ExpenseDao expenseDao = new ExpenseDao();

@FXML

public void initialize() {

refreshCategories();

datePicker.setValue(LocalDate.now());

setupTableColumns();

refreshExpenses();

}

private void setupTableColumns() {

colDate.setCellValueFactory(cd -> new javafx.beans.property.SimpleStringProperty(cd.getValue().getDate()));

colCategory.setCellValueFactory(cd -> new javafx.beans.property.SimpleStringProperty(cd.getValue().getCategoryName()));

colDesc.setCellValueFactory(cd -> new javafx.beans.property.SimpleStringProperty(cd.getValue().getDescription()));

colAmount.setCellValueFactory(cd -> new javafx.beans.property.SimpleObjectProperty<>(cd.getValue().getAmount()));

}

@FXML

public void onAddCategory() {

String name = newCategoryField.getText().trim();

if (name.isEmpty()) return;

try {

categoryDao.create(name, null);

newCategoryField.clear();

refreshCategories();

} catch (Exception ex) {

showError(ex);

}

}

@FXML

public void onAddExpense() {

try {

double amt = Double.parseDouble(amountField.getText().trim());

LocalDate d = datePicker.getValue();

Category cat = categoryCombo.getValue();

String desc = descField.getText();

expenseDao.addExpense(cat == null ? null : cat.getId(), amt, desc, d.toString());

amountField.clear();

descField.clear();

refreshExpenses();

} catch (Exception ex) { showError(ex); }

}

private void refreshCategories() {

try {

List<Category> cats = categoryDao.listAll();

var obs = FXCollections.observableArrayList(cats);

categoryCombo.setItems(obs);

categoryList.setItems(obs);

} catch (Exception ex) { showError(ex); }

}

private void refreshExpenses() {

try {

LocalDate now = LocalDate.now();

List<Expense> rows = expenseDao.listByMonth(now.getYear(), now.getMonthValue());

expenseTable.setItems(FXCollections.observableArrayList(rows));

ChartBuilder.drawChartsForMonth(chartPane, now.getYear(), now.getMonthValue());

} catch (Exception ex) { showError(ex); }

}

@FXML

public void onRefresh() { refreshExpenses(); }

@FXML

public void onExportCsv() { CsvExporter.exportExpensesToCsv(expenseTable.getItems()); }

private void showError(Exception e) {

e.printStackTrace();

new Alert(Alert.AlertType.ERROR, e.getMessage()).showAndWait();

}

}

**\=== FILE: src/main/java/com/example/expensetracker/ui/ChartBuilder.java ===**

**java**

package com.example.expensetracker.ui;

import com.example.expensetracker.db.DB;

import org.jfree.chart.ChartFactory;

import org.jfree.chart.ChartPanel;

import org.jfree.chart.JFreeChart;

import org.jfree.data.category.DefaultCategoryDataset;

import org.jfree.data.general.DefaultPieDataset;

import javafx.embed.swing.SwingNode;

import javafx.scene.layout.StackPane;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

public class ChartBuilder {

public static void drawChartsForMonth(StackPane pane, int year, int month) {

pane.getChildren().clear();

String monthStr = String.format("%04d-%02d", year, month);

try (Connection c = DB.getConnection()) {

// Pie: category breakdown

DefaultPieDataset pie = new DefaultPieDataset();

try (PreparedStatement ps = c.prepareStatement(

"SELECT c.name, COALESCE(SUM(e.amount),0) AS total " +

"FROM categories c LEFT JOIN expenses e ON e.category\_id = c.id AND strftime('%Y-%m', e.expense\_date)=? " +

"GROUP BY c.id, c.name ORDER BY total DESC")) {

ps.setString(1, monthStr);

try (ResultSet rs = ps.executeQuery()) {

while (rs.next()) {

String name = rs.getString("name");

double total = rs.getDouble("total");

if (total > 0) pie.setValue(name, total);

}

}

}

JFreeChart pieChart = ChartFactory.createPieChart("Spending by Category (" + monthStr + ")", pie, true, true, false);

ChartPanel piePanel = new ChartPanel(pieChart);

// Bar: daily totals

DefaultCategoryDataset dataset = new DefaultCategoryDataset();

try (PreparedStatement ps = c.prepareStatement(

"SELECT expense\_date AS day, COALESCE(SUM(amount),0) AS total " +

"FROM expenses WHERE strftime('%Y-%m', expense\_date) = ? GROUP BY expense\_date ORDER BY expense\_date")) {

ps.setString(1, monthStr);

try (ResultSet rs = ps.executeQuery()) {

while (rs.next()) {

String day = rs.getString("day");

dataset.addValue(rs.getDouble("total"), "Total", day);

}

}

}

JFreeChart barChart = ChartFactory.createBarChart("Daily Spending", "Day", "Amount", dataset);

ChartPanel barPanel = new ChartPanel(barChart);

SwingNode snPie = new SwingNode();

SwingNode snBar = new SwingNode();

javax.swing.SwingUtilities.invokeLater(() -> {

snPie.setContent(piePanel);

snBar.setContent(barPanel);

});

javafx.scene.layout.VBox v = new javafx.scene.layout.VBox(8);

v.getChildren().addAll(snPie, snBar);

pane.getChildren().add(v);

} catch (Exception e) {

e.printStackTrace();

}

}

}

**\=== FILE: src/main/java/com/example/expensetracker/util/CsvExporter.java ===**

**java**

package com.example.expensetracker.util;

import com.example.expensetracker.model.Expense;

import javafx.stage.FileChooser;

import java.io.File;

import java.io.FileWriter;

import java.io.PrintWriter;

import java.util.List;

import javafx.scene.control.Alert;

public class CsvExporter {

public static void exportExpensesToCsv(List<Expense> rows) {

FileChooser chooser = new FileChooser();

chooser.setInitialFileName("expenses.csv");

chooser.getExtensionFilters().add(new FileChooser.ExtensionFilter("CSV Files", "\*.csv"));

File f = chooser.showSaveDialog(null);

if (f == null) return;

try (PrintWriter pw = new PrintWriter(new FileWriter(f))) {

pw.println("Date,Category,Description,Amount");

for (Expense ex : rows) {

String line = String.format("%s,%s,%s,%.2f",

ex.getDate(), escapeCsv(ex.getCategoryName()), escapeCsv(ex.getDescription()), ex.getAmount());

pw.println(line);

}

new Alert(Alert.AlertType.INFORMATION, "CSV exported: " + f.getAbsolutePath()).showAndWait();

} catch (Exception e) {

e.printStackTrace();

new Alert(Alert.AlertType.ERROR, "Failed to export: " + e.getMessage()).showAndWait();

}

}

private static String escapeCsv(String s) {

if (s == null) return "";

String out = s.replace("\"", "\"\"");

if (out.contains(",") || out.contains("\"") || out.contains("\n")) {

return "\"" + out + "\"";

}

return out;

}

}

**\=== FILE: README.md ===**

```markdown

# Expense Tracker (Java + JavaFX + SQLite + JFreeChart)

## Build

Requires Java 17+ and Maven.

1. `mvn clean package`

2. Run with: `java -jar target/expense-tracker-1.0.0.jar`

Note: if JavaFX launcher errors occur, run with the JavaFX module path or use jlink/jpackage to create a native package.

## What you get

- `expenses.db` created automatically in working directory.

- GUI to add categories and expenses, view monthly expenses, charts (pie + bar), and export CSV.

## Seed data

Database seeds a few categories automatically. To add sample expenses run `sqlite3 expenses.db` and insert sample rows or use the app.

## Expected results

- On launch you'll see the main window with 3 panels: Add Expense / Expenses list / Charts.

- Adding expenses updates the table and charts immediately.

- Export CSV writes a file with columns: Date,Category,Description,Amount.

```

---

\=== FILE: sample-seed.sql ===

```sql

INSERT INTO expenses(category\_id, amount, description, expense\_date) VALUES(1, 15.50, 'Lunch', '2025-09-01');

INSERT INTO expenses(category\_id, amount, description, expense\_date) VALUES(2, 2.75, 'Bus fare', '2025-09-01');

INSERT INTO expenses(category\_id, amount, description, expense\_date) VALUES(1, 30.00, 'Groceries', '2025-09-03');

INSERT INTO expenses(category\_id, amount, description, expense\_date) VALUES(4, 12.00, 'Movie', '2025-09-05');

```