A Project Structure CSS

```
└── src
    └── main
        ├── java
        │   └── com
        │       └── example
        │           └── dailyexpenses
        │               ├── controller
        │               │   ├── UserController.java
        │               │   └── ExpenseController.java
        │               ├── model
        │               │   ├── User.java
        │               │   └── Expense.java
        │               ├── repository
        │               │   ├── UserRepository.java
        │               │   └── ExpenseRepository.java
        │               └── service
        │                   ├── UserService.java
        │                   └── ExpenseService.java
        └── resources
            └── application.properties
```

(I) 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>daily-expenses-sharing</artifactId>
    <version>1.0-SNAPSHOT</version>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.5.4</version>
    </parent>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
```

```xml
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-validation</artifactId>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

2. USER

```
 package com.example.dailyexpenses.model;
import javax.persistence.*;
import javax.validation.constraints.Email;
import javax.validation.constraints.NotEmpty;
@Entity
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
     @NotEmpty(message = "Name is required")
    private String name;
 @Email(message = "Email should be valid")
    @NotEmpty(message = "Email is required")
    private String email;
  @NotEmpty(message = "Mobile number is required")
    private String mobile;
 // Getters and Setters
}
```

3.Expense

```
package com.example.dailyexpenses.model;
import javax.persistence.*;
import javax.validation.constraints.NotNull;
import java.util.List;
@Entity
public class Expense {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
   @NotNull(message = "Amount is required")
    private Double amount;

    @ElementCollection
    private List<Participant> participants;
 private SplitMethod splitMethod;
 // Getters and Setters
   public static class Participant {
        private Long userId;
        private Double amount;
        private Double percentage;
        // Getters and Setters
    } public enum SplitMethod {
        EQUAL, EXACT, PERCENTAGE
    }
}
```

4. UserRepository

```
package com.example.dailyexpenses.repository;
import com.example.dailyexpenses.model.User;
import org.springframework.data.jpa.repository.JpaRepository;
public interface UserRepository extends JpaRepository<User, Long> {
}
```

5. ExpenseRepository

```
 package com.example.dailyexpenses.repository;
import com.example.dailyexpenses.model.Expense;
import org.springframework.data.jpa.repository.JpaRepository;
public interface ExpenseRepository extends JpaRepository<Expense, Long> {
}
```

6.UserServices

```
package com.example.dailyexpenses.service;
import com.example.dailyexpenses.model.User;
import com.example.dailyexpenses.repository.UserRepository;
```

```
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;
@Service
public class UserService {
    @Autowired
    private UserRepository userRepository;
public User createUser(User user) {
        return userRepository.save(user);
    } public User getUserById(Long id) {
        return userRepository.findById(id).orElse(null);
    }
public List<User> getAllUsers() {
        return userRepository.findAll();
    }
}
```

7.Expensive Service

```
package com.example.dailyexpenses.service;
import com.example.dailyexpenses.model.Expense;
import
com.example.dailyexpenses.repository.ExpenseRepository;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;
@Service
public class ExpenseService {
    @Autowired
    private ExpenseRepository expenseRepository;

    public Expense addExpense(Expense expense) {
        return expenseRepository.save(expense);
    } public List<Expense> getExpensesByUserId(Long userId) {
        // Implement logic to get expenses by user ID
        return null;
    }public List<Expense> getAllExpenses() {
        return expenseRepository.findAll();
    }
}
```

8.Usercontrolled

```
package com.example.dailyexpenses.controller;
```

```
import com.example.dailyexpenses.model.User;
import com.example.dailyexpenses.service.UserService;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.*;

import javax.validation.Valid;
import java.util.List;

@RestController
@RequestMapping("/api/users")
@Validated
public class UserController {
    @Autowired
    private UserService userService;

    @PostMapping
    public ResponseEntity<User> createUser(@Valid
@RequestBody User user) {
        return ResponseEntity.ok(userService.createUser(user));
    }

    @GetMapping("/{id}")
    public ResponseEntity<User> getUserById(@PathVariable
Long id) {
        return ResponseEntity.ok(userService.getUserById(id));
    }

    @GetMapping
    public ResponseEntity<List<User>> getAllUsers() {
        return ResponseEntity.ok(userService.getAllUsers());
    }
}
```

8.expensive controlled

```
package com.example.dailyexpenses.controller;

import com.example.dailyexpenses.model.Expense;
import com.example.dailyexpenses.service.ExpenseService;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.*;

import javax.validation.Valid;
import java.util.List;

@RestController
@RequestMapping("/api/expenses")
@Validated
public class ExpenseController {
    @Autowired
    private ExpenseService expenseService;

    @PostMapping
    public ResponseEntity<Expense> addExpense(@Valid
@RequestBody Expense expense) {
        return
ResponseEntity.ok(expenseService.addExpense(expense));
    }

    @GetMapping("/user/{userId}")
    public ResponseEntity<List<Expense>> getExpensesByUserId(@PathVariable Long userId) {
        return ResponseEntity.ok(expenseService.getExpensesByUserId(userId));
    }

    @GetMapping
    public ResponseEntity<List<Expense>> getAllExpenses() {
        return ResponseEntity.ok(expenseService.getAllExpenses());
    }

    @GetMapping("/balance-sheet")
    public ResponseEntity<String> downloadBalanceSheet() {
        // Implement logic to generate and download balance sheet
        return null;
    }
}
```

9.scheme sql

```
CREATE TABLE IF NOT EXISTS user (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    mobile VARCHAR(255) NOT NULL
);

CREATE TABLE IF NOT EXISTS expense (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    amount DOUBLE NOT NULL,
    split_method VARCHAR(255) NOT NULL
);

CREATE TABLE IF NOT EXISTS participant (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    user_id BIGINT NOT NULL,
    expense_id BIGINT NOT NULL,
    amount DOUBLE,
    percentage DOUBLE,
    FOREIGN KEY (user_id) REFERENCES user(id),
    FOREIGN KEY (expense_id) REFERENCES expense(id)
);
```