**Software Design Document (SDD)**

**1. Introduction**

**1.1. Purpose:**

The Software Design Document defines the architecture, components, and modules required to implement the flatmate bill-splitting system. It specifies the data flow, algorithms, classes, and other technical aspects of the system.

**1.2. Scope:**

The scope includes the technical implementation of bill calculation, input validation, PDF report generation, and file handling. This document outlines how these features will be realized programmatically.

**2. System Architecture**

The system will follow a **modular architecture** with the following components:

1. **Flatmate Class**:
   * Represents each flatmate, stores their name and the number of days they stayed in the flat.
2. **Bill Class**:
   * Represents the bill, containing the total amount and the billing period.
   * Contains the logic for calculating the payment owed by each flatmate.
3. **PDF Generator**:
   * Handles creating a PDF report that summarizes the payment details for the flatmates.
4. **User Input Validator**:
   * Ensures the inputs provided by the user (bill amount, days stayed, names) are valid.
   * Includes error handling for invalid or missing inputs.

**3. Class Design**

**3.1. Flatmate Class**

* **Attributes**:
  + name: A string representing the flatmate's name.
  + days\_stayed: An integer representing the number of days the flatmate stayed in the flat.
* **Methods**:
  + \_\_init\_\_(self, name, days\_stayed): Initializes a flatmate with their name and number of days stayed.

**3.2. Bill Class**

* **Attributes**:
  + bill\_amount: A float representing the total bill amount.
  + bill\_period: An integer representing the billing period in days.
* **Methods**:
  + \_\_init\_\_(self, bill\_amount, bill\_period): Initializes a bill with the total amount and period.
  + payment(self, flatmate\_one, flatmate\_two): Calculates the payment for each flatmate based on their days of stay.
  + print\_payment\_done(self, flatmate\_one, flatmate\_two): Prints the amount each flatmate has to pay.

**3.3. PDF Generator**

* **Methods**:
  + create\_pdf(flatmate1\_name, flatmate2\_name, flatmate1\_bill, flatmate2\_bill): Generates a PDF report summarizing the payments.

**4. Data Flow**

1. **Input Stage**:
   * User enters data about the bill (amount, period) and flatmates (names, days stayed).
2. **Processing Stage**:
   * The Bill object calculates the amount each flatmate has to pay.
3. **Output Stage**:
   * The PDF Generator module creates a PDF report containing the flatmates’ names and the corresponding amounts owed.

**5. Algorithms**

**5.1. Payment Calculation:**

The total payment for each flatmate is calculated based on their days of stay. The formula is:

Flatmate’s Payment=(Bill AmountTotal Days)×Flatmate’s Days StayedTotal Days\text{Flatmate's Payment} = \left( \frac{\text{Bill Amount}}{\text{Total Days}} \right) \times \frac{\text{Flatmate's Days Stayed}}{\text{Total Days}}Flatmate’s Payment=(Total DaysBill Amount​)×Total DaysFlatmate’s Days Stayed​

Where:

* Total Days = flatmate\_one.days\_stayed + flatmate\_two.days\_stayed
* This formula ensures that the total bill is split proportionally to the number of days each flatmate stayed.

**6. Error Handling**

* **Input Validation**:
  + Ensure that the bill amount is a positive float and the days stayed are positive integers.
  + Raise ValueError for invalid inputs and provide informative messages to the user.
* **Directory Management**:
  + If the directory for saving PDFs already exists, use os.makedirs(directory\_path, exist\_ok=True) to avoid errors.

**7. File Management**

* **Directory Structure**:
  + Create a directory named "generated pdf's/" with the current date as a subfolder. The files will be saved in the format Flatmate\_Bill\_v<timestamp>.pdf.

**8. External Libraries**

* **ReportLab**:
  + Used for generating PDFs.
* **OS Module**:
  + Used for creating directories and managing file paths.

**9. Test Plan**

**9.1. Unit Testing:**

* Test the Flatmate class initialization.
* Test the Bill.payment() method to ensure it correctly calculates the amount each flatmate owes.
* Test input validation methods to ensure that they catch invalid inputs.

**9.2. Integration Testing:**

* Test the end-to-end process from input to PDF generation.

**9.3. Performance Testing:**

* Test the time it takes to generate a PDF for various bill amounts and flatmate combinations.