Shopping Lists on the Cloud - Design Plan

Group Members:
Jorge Costa - up201706518
José Cunha - up201905451
Ana Carneiro - 202008569
Margarida Pinho - 201704599

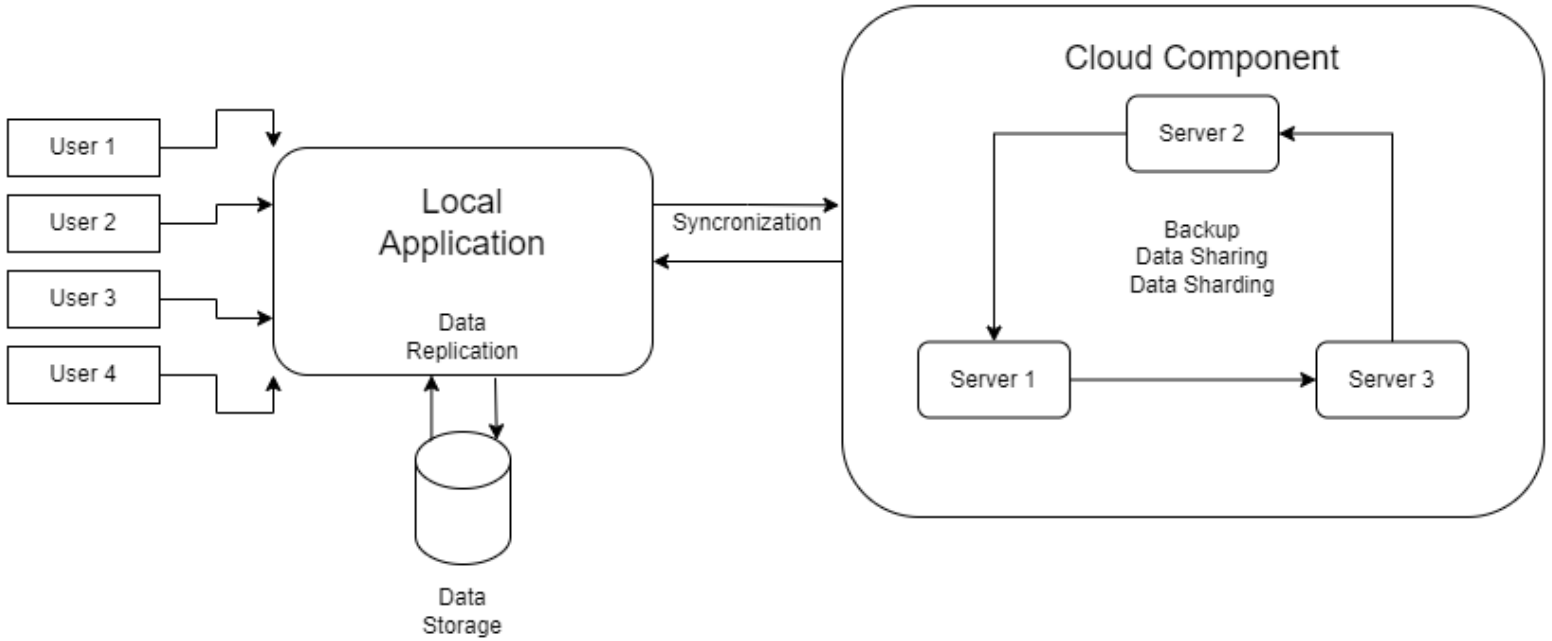
Project Description: This project aims to develop a local-first shopping list application that allows users to create and manage shopping lists. The application will run on user devices, allowing data persistence locally while also incorporating a cloud component for data sharing and backup storage.

System Overview: The system will consist of three primary components:
- **Local Application:** The user interface where users create, manage, and interact with shopping lists.
- **Cloud Component:** A backend infrastructure for data synchronization and storage.
- **Data Storage:** Each shopping list will have a unique ID, and users can add and delete items associated with flags and target quantities.

Programming Language: Python will be used as the primary programming.

Data Replication and Consistency: We will be using Last-Writer-Wins using vector clocks with ZMQ for message passing to manage data consistency, and later transition to Conflict-free Replicated Data Types (CRDTs) as the system evolves.

Scalability: We will be implementing data sharding using Consistent Hashing and Ring Topology to distribute shopping lists across multiple servers for efficient data access.
Also we will use load balancing techniques to ensure even distribution of incoming requests.



Architecture: Adhering to the principles of Local First, the Local Application will reside on the user's device, where it will manage Data Storage and Replication. Synchronization between users will be facilitated through the Cloud Component. For Backup, Data Sharing, and Data Sharding, multiple Cloud servers will employ techniques such as Consistent Hashing, Ring Topology, and Load Balancing.

Message Protocol: We will define a message protocol for communication between local applications and the cloud component. The protocol structure will include:

- **Message Types:** We will be defining various message types, like synchronization requests, data sharing, and error handling.
- **Message Formats:** We will specify the structure of each message type, including the fields and data to be included.
- **Error Handling:** We will be defining how errors will be communicated within the message protocol, including error codes and descriptions.

Example:

```
{"messageType" "syncRequest",
 "listId" "12345",
 "data"{
    "items"[
      {"name" "Milk", "quantity" 2},
      {"name" "Bread", "quantity" 1},
```