Local-First Shopping List Application with Cloud Component

Large Scale Distributed Systems

Afonso Martins - up202005900@edu.fe.up.pt
Anete Pereira - up202008856@edu.fe.up.pt
Eduardo Silva - up202005283@edu.fe.up.pt
Hugo Castro - up202006770@edu.fe.up.pt

Master in Informatics and Computing Engineering

November 7, 2023

Contents

1 Introduction

The Local-First Shopping List Application is designed to allow users to create and manage shopping lists both locally and in the cloud. Users can collaborate on shopping lists and ensure data availability, even when offline. The project aims to achieve high availability and scalability for millions of users.

2 System Architecture

2.1 Frontend (React with TypeScript)

- Local-First Approach: The frontend utilizes a local-first approach to ensure that users can create and modify shopping lists even when offline. Shopping lists are stored locally on the user's device. Local data is structured in a way that allows for efficient conflict resolution and synchronization with the cloud when a network connection is available.
- User Interface: Provides an intuitive and responsive user interface that allows users to create and manage shopping lists. Offers real-time updates for collaborative features when connected to the internet.

2.2 Backend (Node.js with TypeScript)

- Local-First Support:
  - The backend provides APIs and services for user authentication and list management.
  - Manages user accounts and list permissions to support the local-first approach effectively.
  - Each shopping list has a unique identifier (ID) associated with it, enabling users to access and collaborate on lists.
- Conflict Resolution and Versioning:
  - Initially uses Last-Writer-Wins with local clocks to handle conflicts and prioritize changes made by the most recent user.
  - As the system evolves, the transition to CRDTs allows for more sophisticated conflict resolution, ensuring data integrity and consistency.
  - Implements data versioning to track changes and revisions made to shopping lists, facilitating conflict resolution and auditability.

2.3 Cloud Component

- Server Distribution:
  - Employs a distributed architecture inspired by Amazon Dynamo to ensure scalability and availability.
  - Utilizes multiple distributed server nodes to provide fault tolerance and high availability. Each node is responsible for handling a subset of shopping lists (sharding) to distribute the data load effectively.
- Data Sharding:
  - Lists are partitioned based on their unique IDs to evenly distribute data and minimize data access bottlenecks.
  - Shards can be dynamically redistributed as needed to balance the system's load.
- High Availability:
  - Implements redundancy and failover mechanisms to ensure uninterrupted service in case of server failures.
  - Uses load balancers and auto-scaling to maintain system performance during high traffic loads.
  - Implements data replication across multiple server nodes to ensure data durability and availability.
- Data Synchronization:
  - Utilizes efficient data synchronization mechanisms to keep local copies and cloud data in sync.

- When a user is online, their local changes are synchronized with the cloud component, and vice versa.
- Incorporates versioning and conflict resolution logic to handle concurrent changes and merge data seamlessly.

3 Storage and Database Design

3.1 Users

- User profiles will store user information, including user ID, username, email, and hashed passwords for authentication, which provides a unique identifier for each user to associate them with their shopping lists

3.2 Shopping Lists

- Each shopping list will be represented by a set of Conflict-free Replicated Data Types (CRDTs). These can be nested, and will most likely be of different types (e.g. String, Integer/Count, List or Dictionary). This implementation will be based on the Automerge library, which will ensure resolution of conflicts when multiple users make concurrent changes to the same shopping list. CRDTs enable seamless synchronization of data across devices, maintaining data consistency even in the presence of network delays or conflicts.
- Shopping list metadata will include fields such as an ID, name, datetime fields for creation and last modification and the ID of the owner.
- Each shopping list will also have an associated collaborators object to store the user ID's of the people who can access the list and their permissions.

4 Data Synchronization

4.1 Local Persistence

Uses local storage or indexedDB for storing list data on the user's device. Implements an offline-first approach, allowing users to create and modify lists without an internet connection. Synchronizes local data with the cloud when a connection is available.

4.2 Cloud Storage

Stores list data in a distributed cloud storage system. Utilizes sharding to distribute data efficiently across multiple servers. Implements versioning and conflict resolution mechanisms for cloud data to handle concurrent changes by users.

5 Security

User authentication and authorization for list access. Secure transmission of data using HTTPS. Encryption of sensitive user data. Regular security audits and updates.

6 Testing

Comprehensive unit, integration, and end-to-end testing for the frontend and backend. Load testing to evaluate system scalability. Extensive testing of data synchronization and conflict resolution.

7 Conclusion

The Local-First Shopping List Application with a Cloud Component is a feature-rich and scalable solution that enables users to create, manage, and collaborate on shopping lists with high availability and data synchronization. The project follows best practices in software development and security to provide an optimal user experience.