Continuous File Synchronization System

October 29, 2024

Chapter 1

Introduction

1.1 Motivation

In today's digital age, the need for seamless, real-time file synchronization across devices has become essential. With the growth of remote work, distributed teams, and multi-device use, individuals and organizations require reliable solutions that ensure data is accessible and up-to-date without manual intervention. While cloud storage providers offer some solutions, they often come with privacy concerns, limitations on customization, and dependency on third-party servers.

To address these challenges, our graduation project focuses on developing a robust file synchronization system inspired by platforms like Syncthing, aiming for a decentralized, peer-to-peer approach that priotrizes user privacy, security, and ease of use. our intention is to eliminate reliance on centralized servers, providing users with greater control over their data while ensuring conflict-free file synchronization across devices. Through this system, we seek to contribute to the field of distributed computing by offering an alternative that enhances data accessibility and resilience

1.2 Problem Statement

Background

In a world where digital collaboration, remote work, and data accessibility are becoming increasingly essential, users often need real-time access to the most up-to-date versions of files across multiple devices. Traditional file-sharing services can be limited by connectivity issues, centralized storage constraints, or restrictions in updating files consistently across distributed devices. Hence, developing an efficient, decentralized file synchronization system to ensure continuous file availability and consistency across multiple devices in real time is crucial.

Objective

This project aims to design, develop, and implement a Continuous File Synchronization System that enables users to automatically synchronize files across devices while maintaining data integrity, handling conflicts effectively, and ensuring minimal data transfer. The system will allow users to sync files locally and remotely, with features such as file versioning, conflict resolution, and support for multiple operating systems.

Scope

The system will:

1. **Support Real-time Synchronization**: Files updated on one device should be reflected on all connected devices in real-time, ensuring minimal delay.

2. **Enable Cross-platform Compatibility**: The system should work on Windows, macOS, and Linux, allowing users to sync files across different operating systems.

3. **Ensure Data Integrity and Security**: Files should be transferred securely, and the system should ensure data integrity with mechanisms to verify file consistency.

4. **Handle Conflict Resolution**: The system should include a mechanism to handle conflicts when files are modified simultaneously on multiple devices.

5. **Implement Scalability**: The solution should scale efficiently to accommodate an increasing number of files, devices, and users.

Functional Requirements:

1. **Automatic Synchronization**: Detect file changes automatically and sync them across connected devices.

2. **Version Control**: Maintain version history of files to allow users to revert to previous versions if needed.

3. **Conflict Management**: Identify and resolve conflicts between different versions of the same file across devices.

4. **User Interface**: Provide an intuitive GUI for easy user interactions and configuration of sync settings.

1.3 Goals

1.4 Related Work

1.4.1 Currently Available Solutions

1.4.2 Features Matrix

1.5 Software Development Methodology Used

Chapter 2

Requirements Analysis

2.1 Surveys

2.2 Results of (Part 2.1) represented as charts

2.3 Functional Requirements

2.4 Non-functional Requirements

2.5 Use Case Diagrams

Chapter 3

Design

3.1 Sequence Diagrams

3.2 ERD

3.3 Data Flow Diagrams (DFD-level 0 and DFD-level 1)

3.4 Algorithms

3.5 UML Class Diagrams

Chapter 4

Implementation Aspects

4.1 Overall System Architecture (Blocks diagram)

4.2 Tools, Technologies and/or Programming Languages

4.3 Prototype