CSA0492-OPERATING SYSTEM FOR DYNAMIC STORAGE MANAGEMENT

FILE SYNCHRONIZATION UTILITY USING OS

MINI PROJECT-MAR 2024

GUIDE BY:

DR.YUVARANI

PRESENTED BY:

G.VENKATA SAI BHARGAV(192210699)

P.DEVI PRASAD(192210625)

P.VISHNU VARDHAN(192210520)

ABSTRACT:

- This project aims to develop a file synchronization utility leveraging operating system (OS) techniques. The utility will facilitate the efficient synchronization of files across multiple devices, ensuring data consistency and integrity. Through the utilization of OS functionalities, such as file handling and process management, the utility will provide a robust and scalable solution for users to synchronize their files seamlessly.

INTRODUCTION:

With the Propagation of digital devices and the increasing need for data accessibility, file synchronization has become essential in modern computing environments. Users often work across multiple devices and platforms, necessitating a reliable mechanism to ensure that files remain up-to-date across all endpoints. Traditional file synchronization methods often rely on third-party software or cloud services, which may pose security and privacy concerns. In this project, we propose the development of a file synchronization utility that harnesses the capabilities of the underlying operating system to achieve efficient and secure file synchronization.

OBJECTIVES:

- Automated Synchronization
- Real-Time Monitoring
- Efficient Resource Management
- Cross-Platform Compatibility
- Version Control
- Secure Data Transfer
- User-Friendly Interface
- Error Handling.

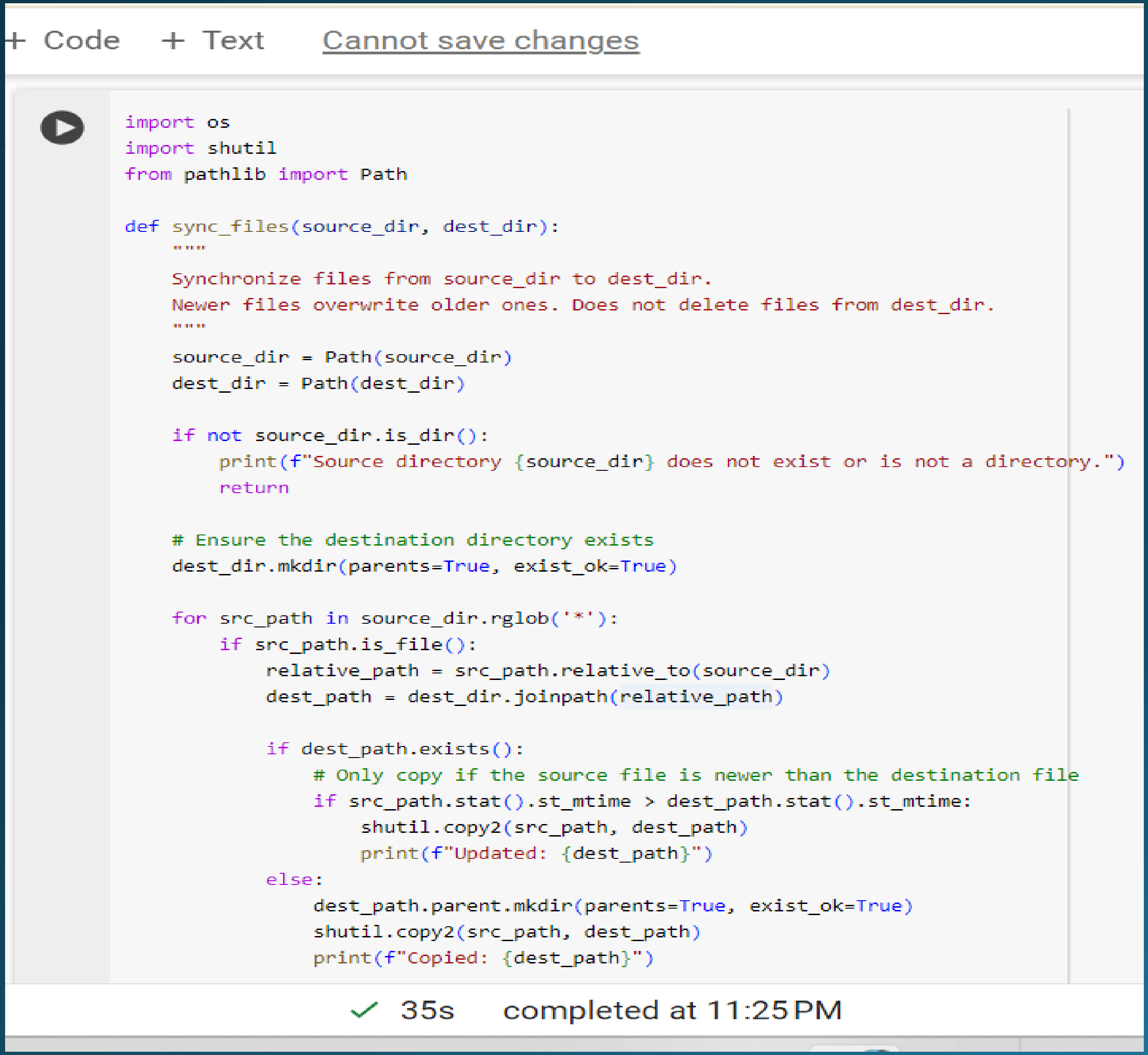
METHODOLOGY:

Requirement Analysis: Identify the specific requirements of the file synchronization utility, including supported platforms
Design Phase: Design the architecture of the utility, including components such as file monitoring modules, synchronization engines, user interfaces, and configuration management.
Development: Implement the file synchronization utility according to the design specifications.
File Monitoring and Change Detection: Develop modules for real-time file monitoring and change detection using OS-specific APIs
User Interface: Develop a user-friendly interface for configuring synchronization settings, monitoring synchronization progress, and managing synchronized files and folders.
Testing: Conduct thorough testing of the file synchronization utility to ensure functionality, reliability, and performance across different operating systems and scenarios.
Deployment: Package the file synchronization utility for distribution, ensuring compatibility with installation procedures on various operating systems.

ADVANTAGES:

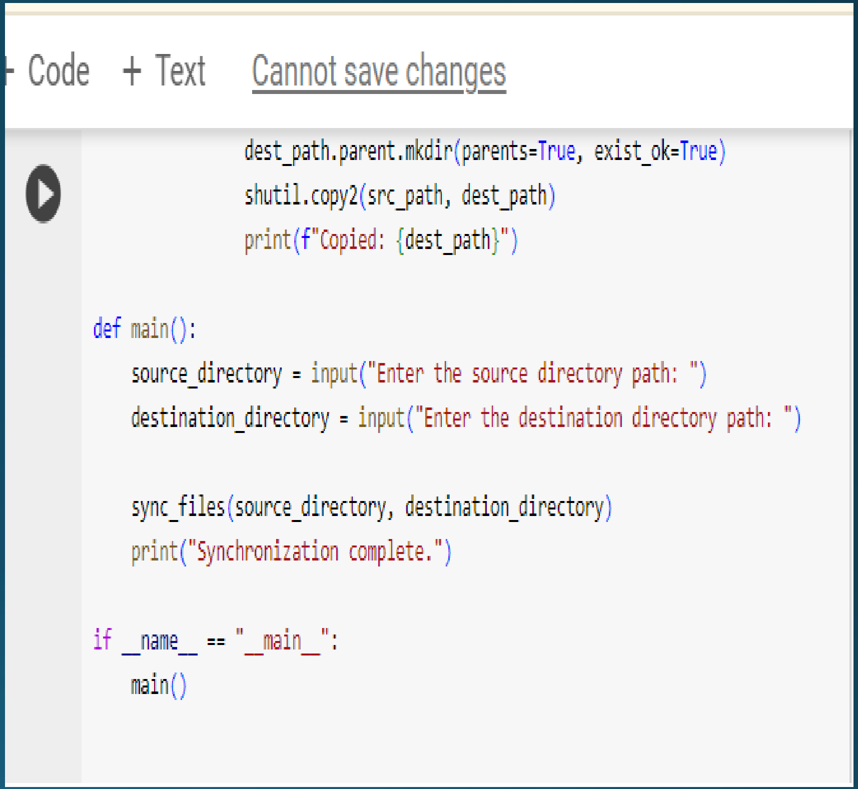
- ✓ Platform Compatibility
- ✓ Real-Time Monitoring
- ✓ Security Features
- ✓ File System Awareness
- ✓ User Experience
- ✓ Vendor Support and Updates

CODE:



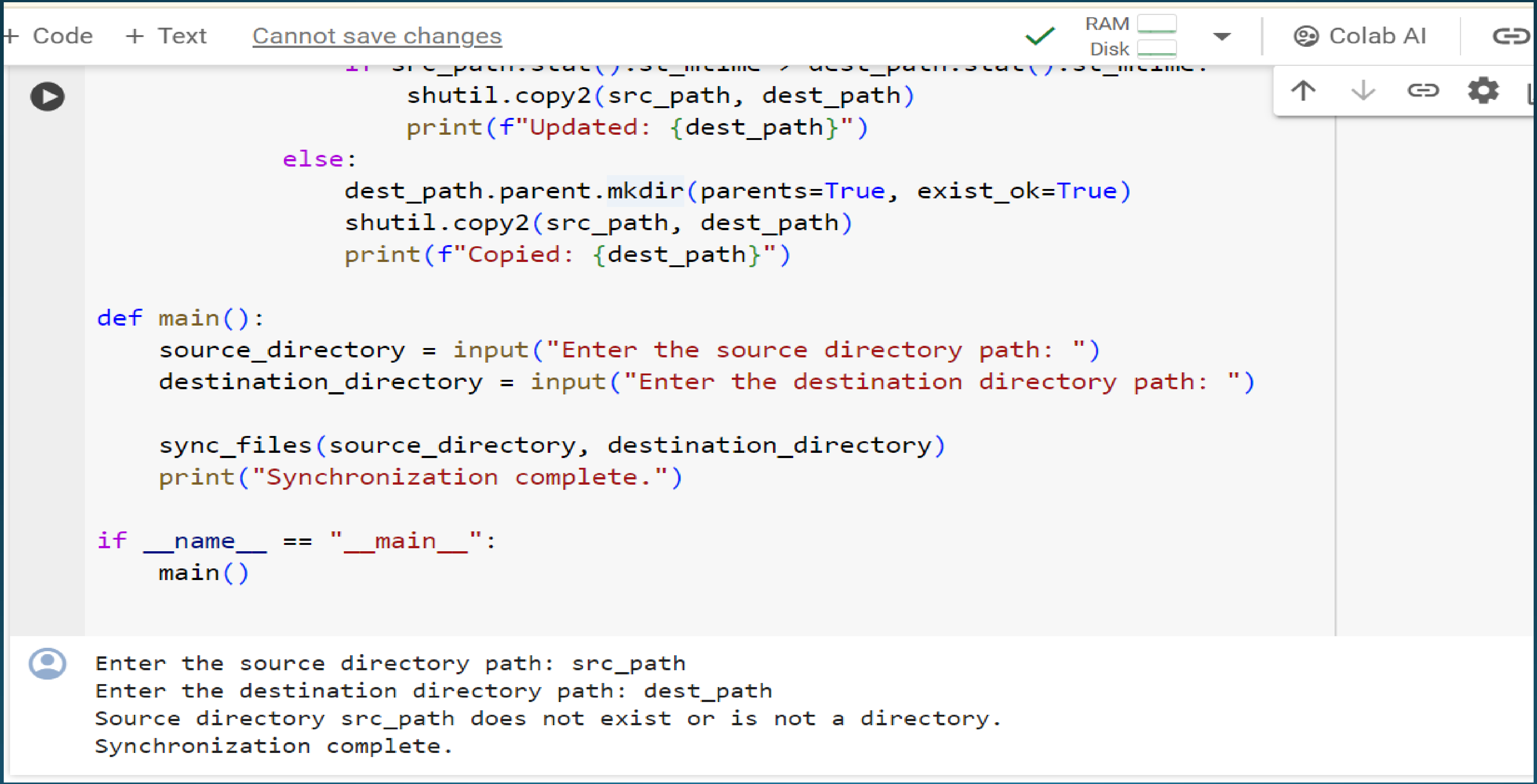
```python
import os
import shutil
from pathlib import Path

def sync_files(source_dir, dest_dir):
    """
    Synchronize files from source_dir to dest_dir.
    Newer files overwrite older ones. Does not delete files from dest_dir.
    """
    source_dir = Path(source_dir)
    dest_dir = Path(dest_dir)

    if not source_dir.is_dir():
        print(f"Source directory {source_dir} does not exist or is not a directory.")
        return

    # Ensure the destination directory exists
    dest_dir.mkdir(parents=True, exist_ok=True)

    for src_path in source_dir.rglob('*'):
        if src_path.is_file():
            relative_path = src_path.relative_to(source_dir)
            dest_path = dest_dir.joinpath(relative_path)

            if dest_path.exists():
                # Only copy if the source file is newer than the destination file
                if src_path.stat().st_mtime > dest_path.stat().st_mtime:
                    shutil.copy2(src_path, dest_path)
                    print(f"Updated: {dest_path}")
            else:
                dest_path.parent.mkdir(parents=True, exist_ok=True)
                shutil.copy2(src_path, dest_path)
                print(f"Copied: {dest_path}")
```

✓ 35s completed at 11:25 PM



```python
                dest_path.parent.mkdir(parents=True, exist_ok=True)
                shutil.copy2(src_path, dest_path)
                print(f"Copied: {dest_path}")

def main():
    source_directory = input("Enter the source directory path: ")
    destination_directory = input("Enter the destination directory path: ")

    sync_files(source_directory, destination_directory)
    print("Synchronization complete.")

if __name__ == "__main__":
    main()
```

OUTPUT SCREEN:



```python
                if src_path.stat().st_mtime > dest_path.stat().st_mtime:
                    shutil.copy2(src_path, dest_path)
                    print(f"Updated: {dest_path}")
            else:
                dest_path.parent.mkdir(parents=True, exist_ok=True)
                shutil.copy2(src_path, dest_path)
                print(f"Copied: {dest_path}")

def main():
    source_directory = input("Enter the source directory path: ")
    destination_directory = input("Enter the destination directory path: ")

    sync_files(source_directory, destination_directory)
    print("Synchronization complete.")

if __name__ == "__main__":
    main()
```

```
Enter the source directory path: src_path
Enter the destination directory path: dest_path
Source directory src_path does not exist or is not a directory.
Synchronization complete.
```

FUTURESCOPE

Future scope for file synchronization utilities using OS technologies is dynamic and multi-faceted, with opportunities to innovate, collaborate, and address emerging challenges in data management, security, mobility, and user experience.

CONCLUSION:

- In conclusion, developing a file synchronization utility using operating system (OS) capabilities offers a wide range of advantages and promising future prospects. Through this approach, developers can leverage the inherent features of OS platforms to create efficient, reliable, and user-friendly solutions for synchronizing files and folders across diverse environments.

Thank you

Adapt it with your needs and it will capture all the audience attention.