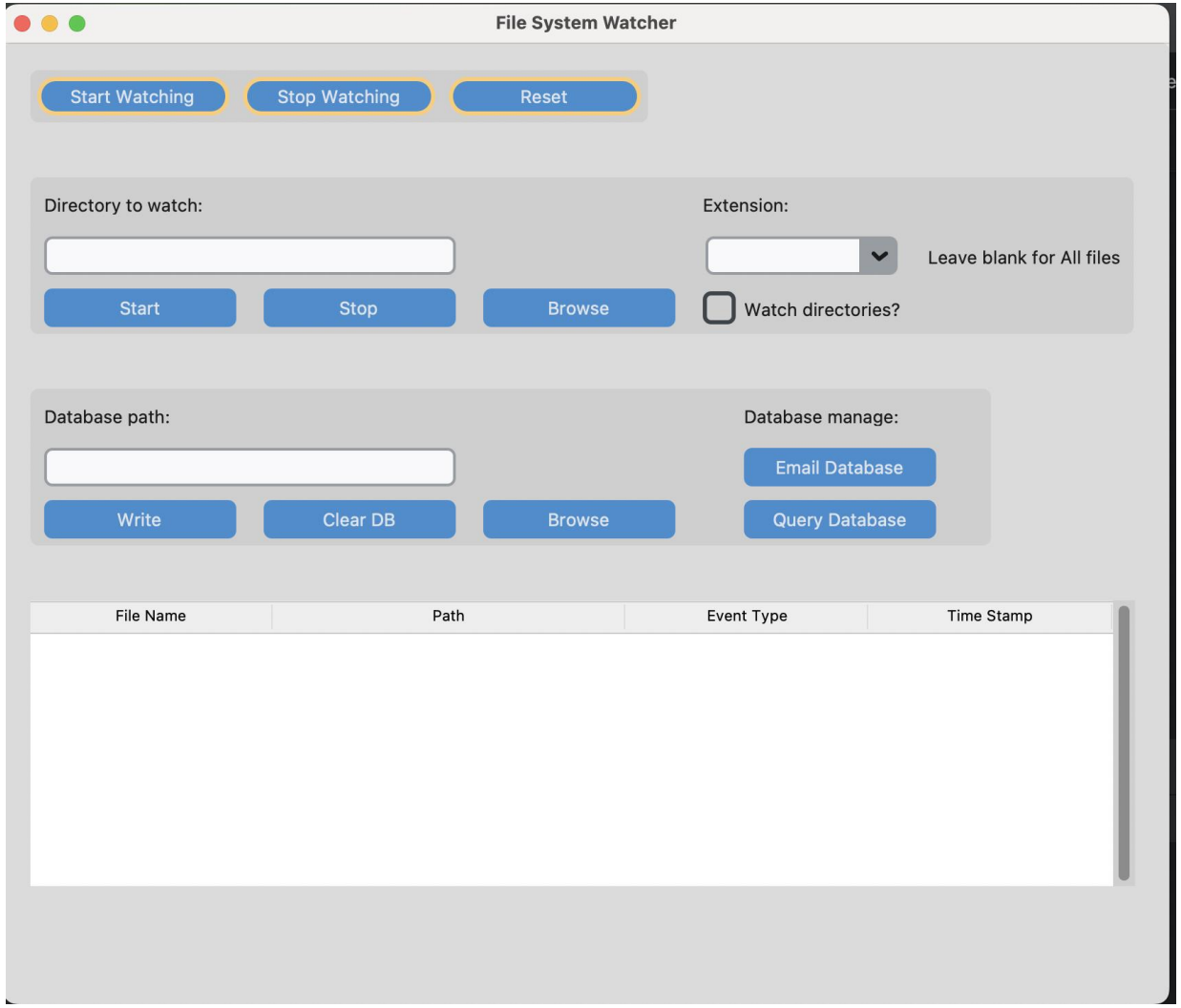
File System Watcher



By Team One

Weilan Liang

Aden Abdulani

Dereje Teshager

Agenda

- Description of project
- What each person worked on
- Updated UML Class Diagram
- Highlight design patterns, OO principles and other design features you feel are important
- How much time we spent on the project
- List challenges you faced and how you resolved them
- List any remaining work or future work you would like to perform
- List what Team learned

Project Description

Our project is a comprehensive directory monitoring GUI application designed to track and log file system events in real time. It enhances basic directory monitoring capabilities with additional features that improve usability, data management, and user control.

Below are the key features:

Cont...

1. Directory Monitoring Feature

This feature continuously monitors a user specified directory (and optionally its subdirectories) for file system events, including file creation, deletion, modification and movement. It utilizes the Watchdog library to capture these events in real time.

Additionally, users can filter monitoring to track only specific file types based on their extensions, allowing for more targeted event tracking.

Cont...

2. Event Logging and Database Management

All captured file system events are logged into an SQLite database, ensuring organized and structured data storage. Each event entry includes:

- Event Type (e.g., created, deleted or modified)
- File Path
- Timestamp
- File Name

This structured logging allows users to efficiently retrieve and analyze data as needed.

Cont...

3. User Interface and Control

A user-friendly **Graphical User Interface (GUI)**, built with **Tkinter**, provides intuitive control over the application.

Through the interface, users can:

- Configure monitoring settings (e.g., select directories, check/uncheck subdirectory tracking checkbox)
- Start and stop the file system watcher
- Specify the database location for event logging

Cont...

4. Advanced Query and Export Capabilities

To enhance data accessibility, the application includes a query feature that allows users to filter logged events based on:

- File Extension
- Event Type
- Time Frame

Cont...

5. Emailing and Export Feature

To enhance data accessibility and sharing, the application includes an emailing feature that allows users to send the event log database as a CSV file via email.

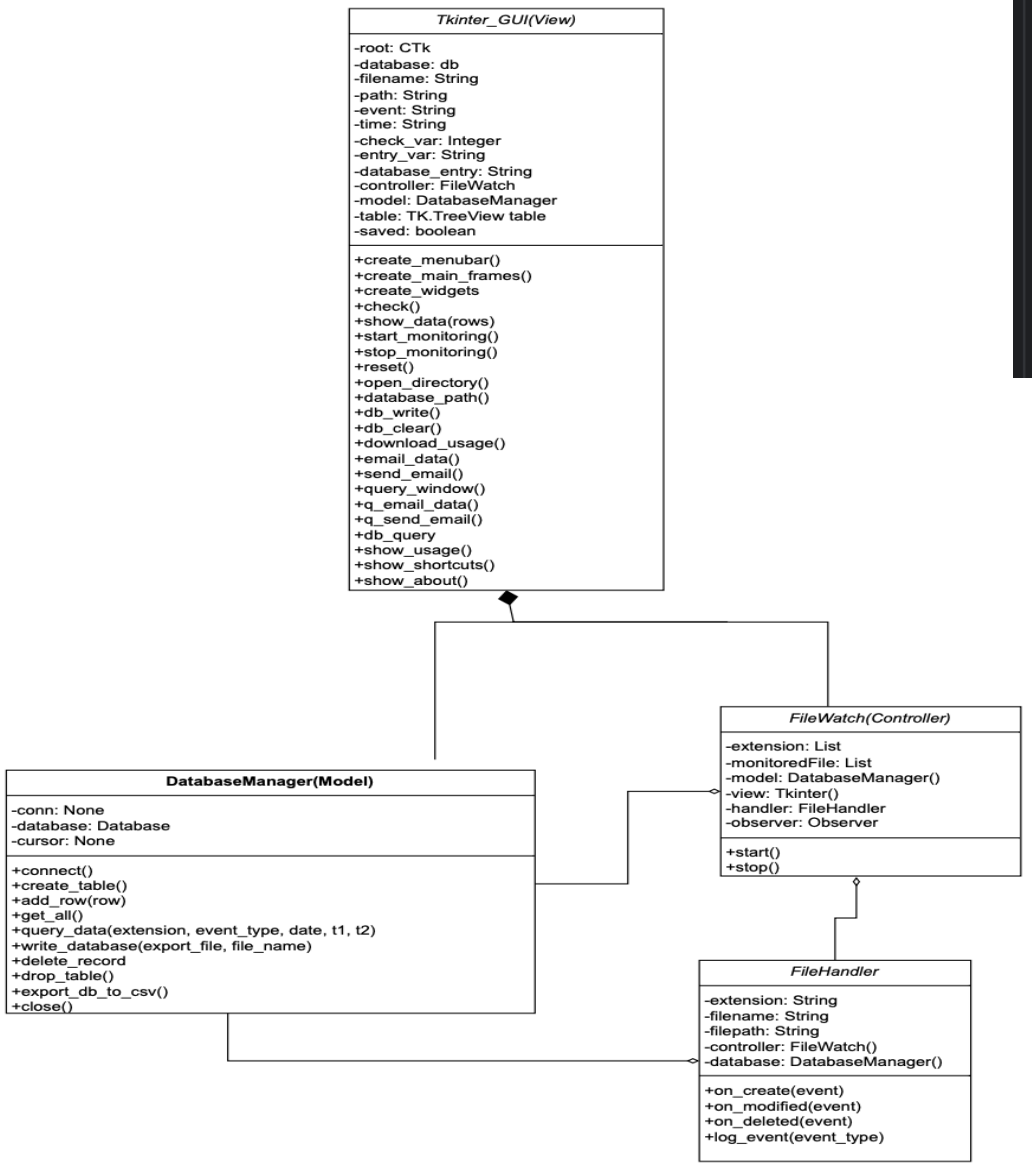
Additionally, users can export logged events in multiple formats, including:

- SQLite database files
- CSV files
- Other file types for flexible storage and analysis

What we worked on

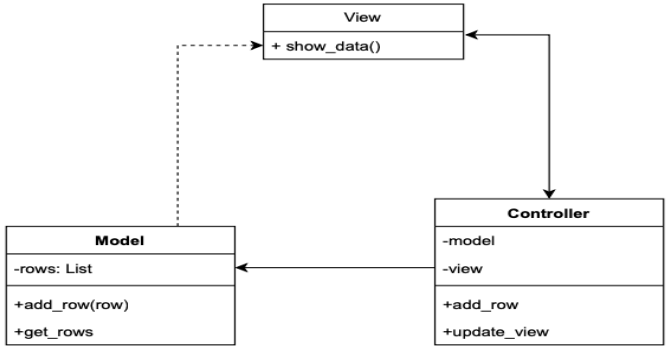
- All:
  - UML, SRS
  - Debugging problems and brainstorm to solve problems
- Dereje
  - Create Tkinter_GUI class and creates some functions on the other classes
- Aden
  - Contributed Tkinter_GUI class:Such as Help, About, usage guide, short-control, reset
  - Contributed Some functions on other classes, Documentation, and submission of project iterations
- Weilan Liang:
  - Create FileWatch, FileHandler, DatabaseManager, MVC(Model, View, Controller) classes
  - Build Query window on Tkinter class and create functions to allow user to query the database

Updated UML




```
if __name__ == "__main__":
    root = CTk()
    view = Tkinter_GUI(root)  # Now creat
    model = DatabaseManager()
    controller = FileWatch(model, view)
    view.controller = controller
    view.model = model

    root.mainloop()
```



- https://app.diagrams.net/#G1PQ8X5MZ0uNgNKsq3UNlGcGQBpkqoZwCl%23%7B%22pageId%22%3A%22PFmxeXQxJv6AQt7kSme6%22%7D

Design Patterns, OO principles

- Abstraction
  - We use abstraction because we want to MVC serves as blueprint for other classes, but the actual implementation can differ between subclasses.
  - Meaning that any subclass of `Model` an View must provide its own implementation of how data are added and displayed.
- MVC(Model-View-Controller)
  - This is essential for file watch program. The Controller (File Watcher) will received orders from View(by user) ask Model (DatabaseManager) to add data the database, notify the `View` (Tkinter) or other components whenever there is a change in the file system.
  - The controller is a mediator between Model and View; handling user input and updating the View.

How much time Team spent on the project

Members/ Iter Num	Aden A	Weilan L	Dereje T	Total Hours	Remard
6 Plus	57	65	60.5	182.5	
Total Teem Meetings			14 meetings		

Challenges, and How we resolved Cont...

A) Challenges:

Each Project Iteration had its own challenge:

1. Confusion on how to do UML relationships between classes.
2. Code writing like putting data from the SQL to tkinter frame.
3. How to handle database.
4. Updating and creating interactive ways between database to Tkinter.
5. Creating a second window for Query and how to filter database using SQL.

Challenges, and How we resolved Cont...

A) Challenges Continued:

6.Creating csv file, and sending an email attached with the watched data.

7.Converting the file watch program into MVC, during this the challenge was:

- a) How to call controller on GUI(VIEW)
- b) How to correctly implement MVC to all classes

8.How to query between times?

Challenges, and How we resolved Cont...

c) How the Team Resolved:

1. Discussing as a team, watching lecture and YouTube videos, reading lecture Powerpoints.
2. Reached consensus on imagination and agreed to modify according to the progress.
3. Continues contact and consultation to receive a comment from Professor Tom, and we consulted and received feedback, and clarification.

Challenges, and How we resolved Cont...

B) How the Team Resolved: Contin..

4. Recreating DatabaseManager class to manipulate data into the database.

5.The team wrote TKinter(View, observer) and DatabaseManager (Model, observable) classes, they communcate through FileWatch(Controller) to notify, and update the TKinter automatically.

6.Implemented MVC, and create csv file, and send that as an attachment with email,

List of remaining work

The remaining work of the projects:

- Unit test (In progress)
- Clean code & Documentation
- Making the GUI application a standalone window application so that it can be installed on any windows without the need of python or PyCharm.
- Testing the project on different platforms, mobile, laptop, or desktop,..

What we learned

Team learned:

I) How to start a project from scratch to finish

II) How to implement Model View Control to different classes

III) The importance of team-work, punctuality, effective communication, empathy, focus on project goal

IV) How to resolve if some project area become unclear/not working, like searching online, reading lecture power-points, consulting the professor

V) Many More..

Thank You

Feeling gratitude and not expressing it is like wrapping a present and not giving it.

Next: Our Program Demo