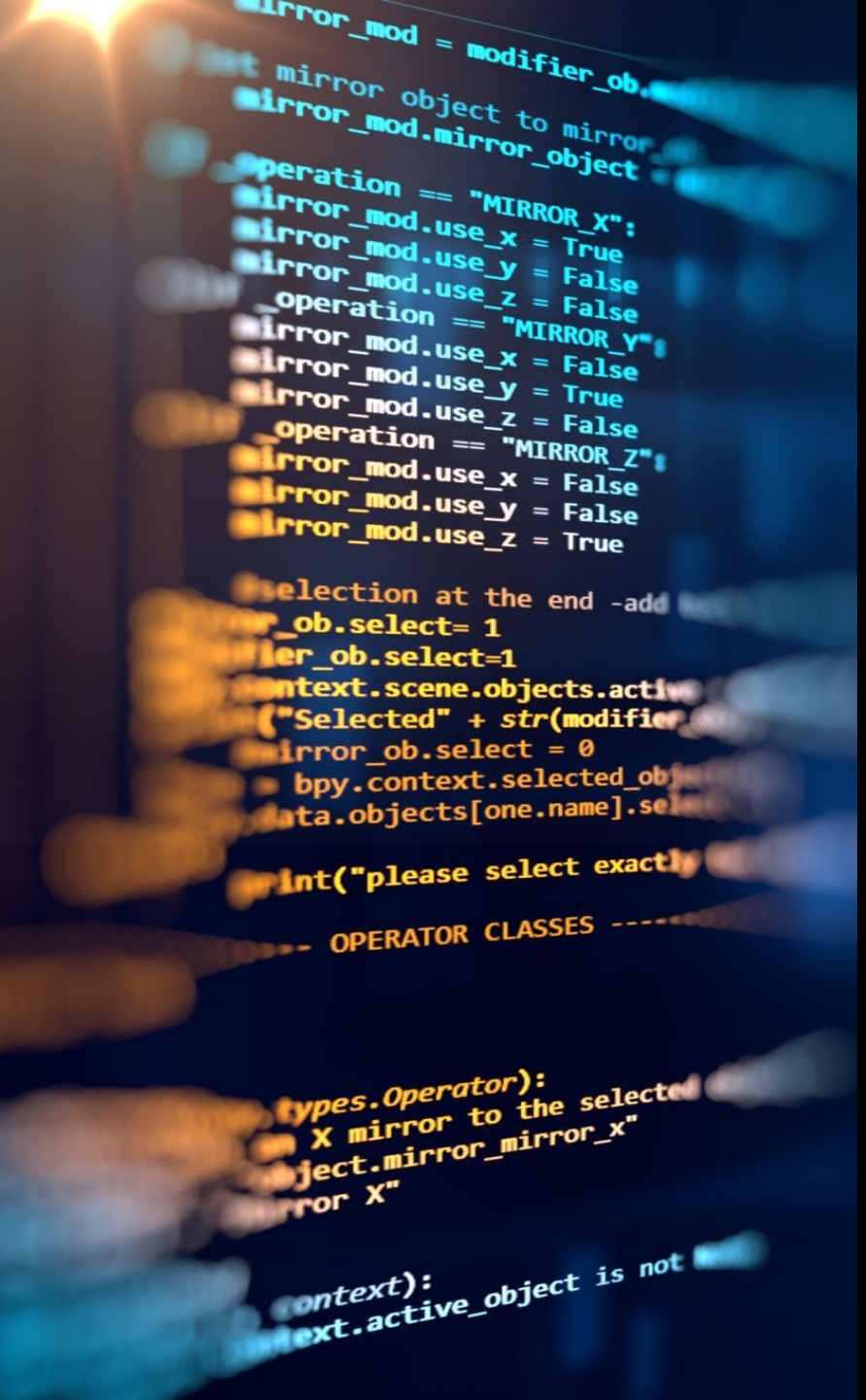
Personal Information Manager

JIANG Guanlin (21093962D)
MENG Guanlin (20099185D)
HU Yuhang (21106395D)
YE Feng (21098249D)

Outline

- System Requirements
- Overall Design & Architecture
- Learning from the project

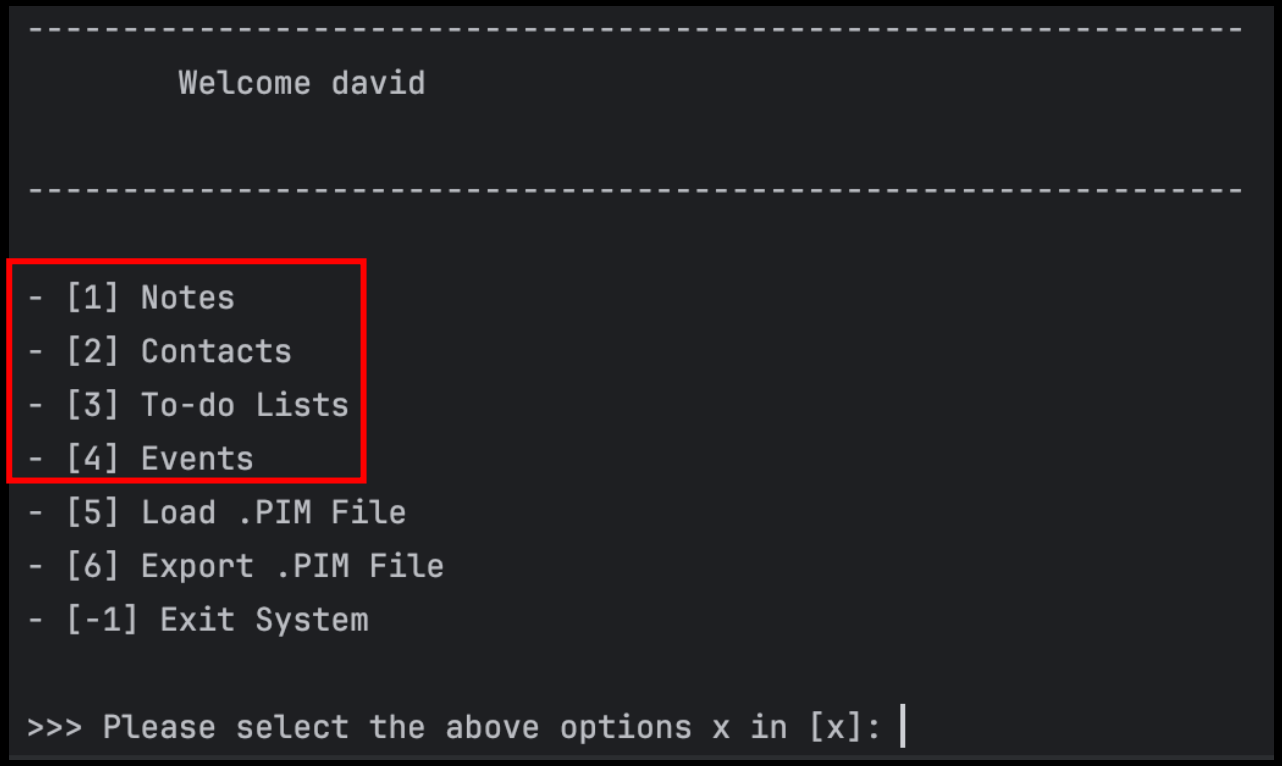


System Requirements

- Creation of a PIR
- Definition of a Search Criterion
- Search for Specific PIRs

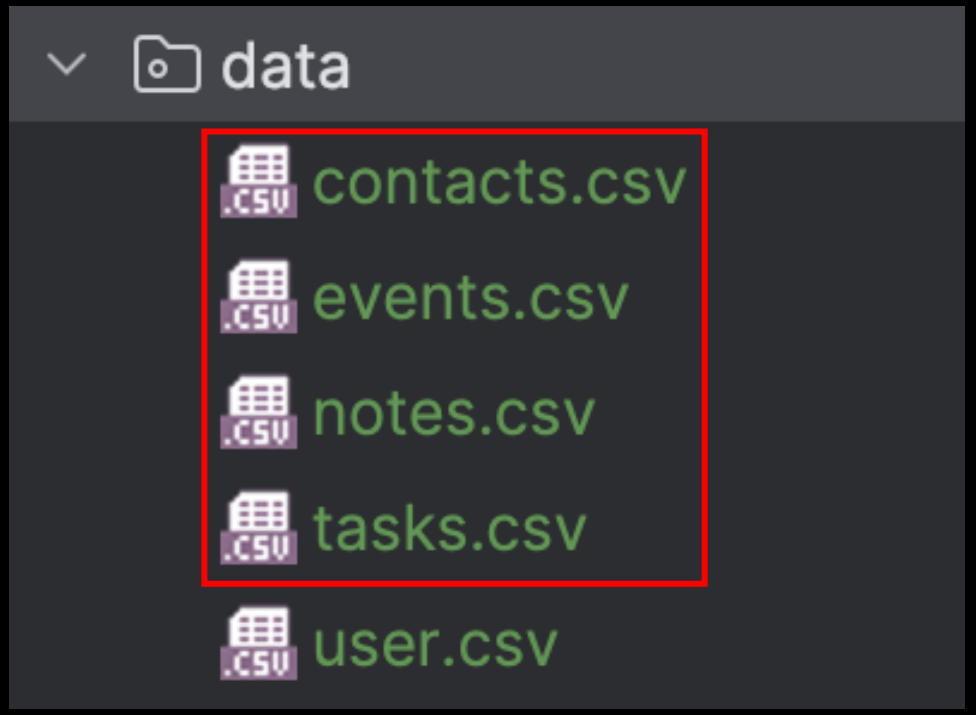
Creation of a PIR

- The system should provide a user-friendly interface for creating PIRs (A main menu can guide users enter into four pages for creating PIRs.)



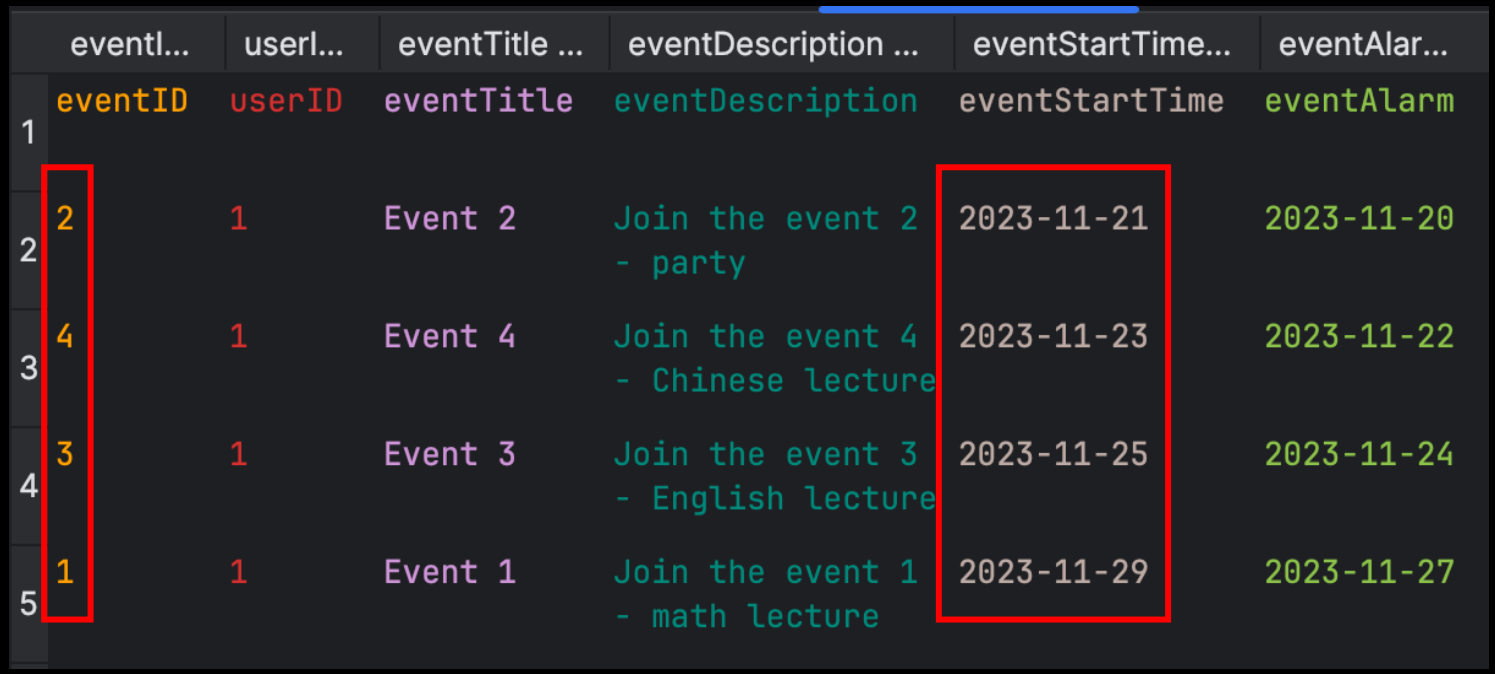
Creation of a PIR

- The system should store the created PIR into the corresponding csv file (database) according to four types of PIRs.



Creation of a PIR

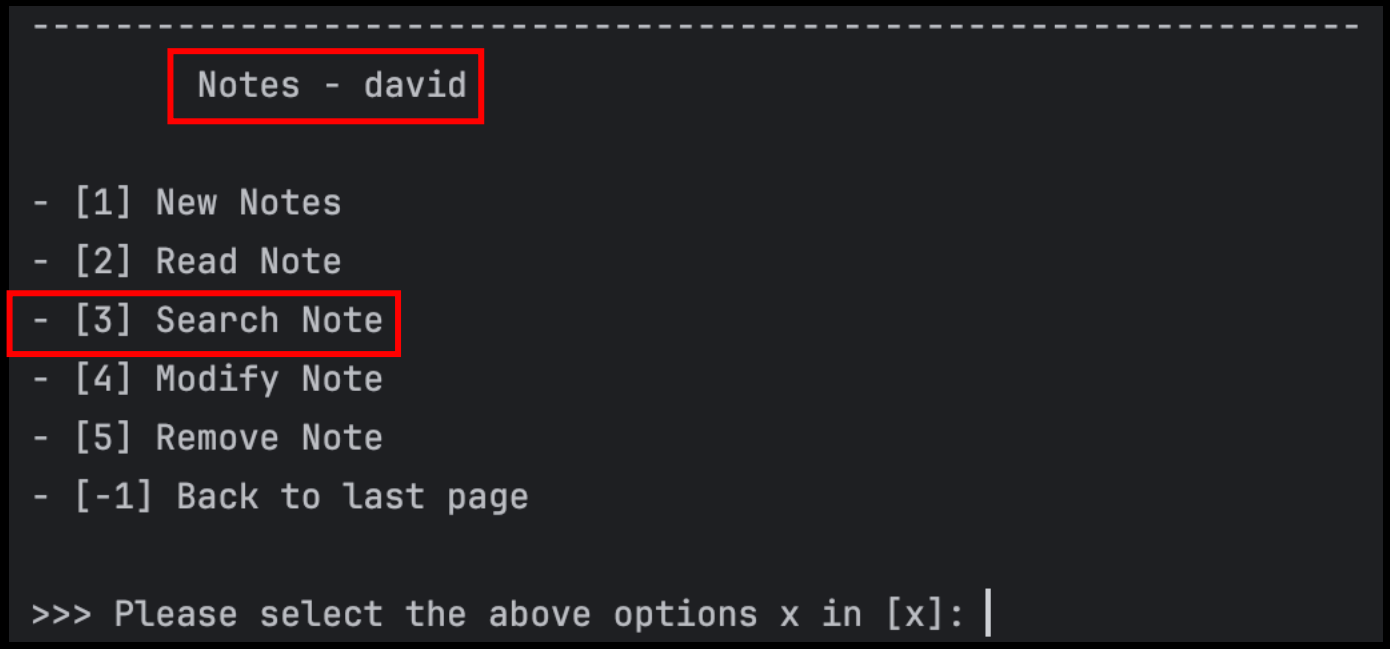
- The system should sort the created PIR by time and/or user ID in each csv file (database).



	eventI...	userI...	eventTitle ...	eventDescription ...	eventStartTime...	eventAlar...
1	eventID	userID	eventTitle	eventDescription	eventStartTime	eventAlarm
2	2	1	Event 2	Join the event 2 - party	2023-11-21	2023-11-20
3	4	1	Event 4	Join the event 4 - Chinese lecture	2023-11-23	2023-11-22
4	3	1	Event 3	Join the event 3 - English lecture	2023-11-25	2023-11-24
5	1	1	Event 1	Join the event 1 - math lecture	2023-11-29	2023-11-27

Definition of a Search Criterion

- The system should support local search in four pages regarding four types of PIRs. For example, the following is the note page which can support local search for all information about notes.



Definition of a Search Criterion

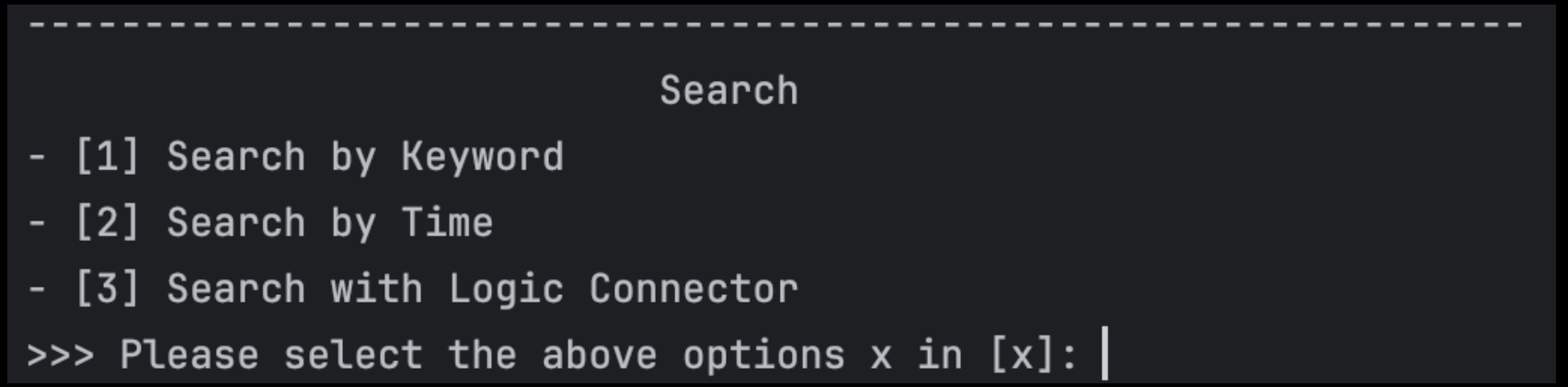
- The system should classify the data into two types including keywords and time for searching. (Use whether the data is preceded by an operator (>, <, =) to distinguish between keywords and time.)

  Keyword: hello

  Time: = 2023-11-19 20:30:00

Search for Specific PIRs

- Three ways for searching:
  - The system should allow users to search by a single keyword.
  - The system should allow users to search by a single time.
  - The system should allow users to search by multiple keywords and/or time with logical connectors (&&, ||, !).



Search for Specific PIRs

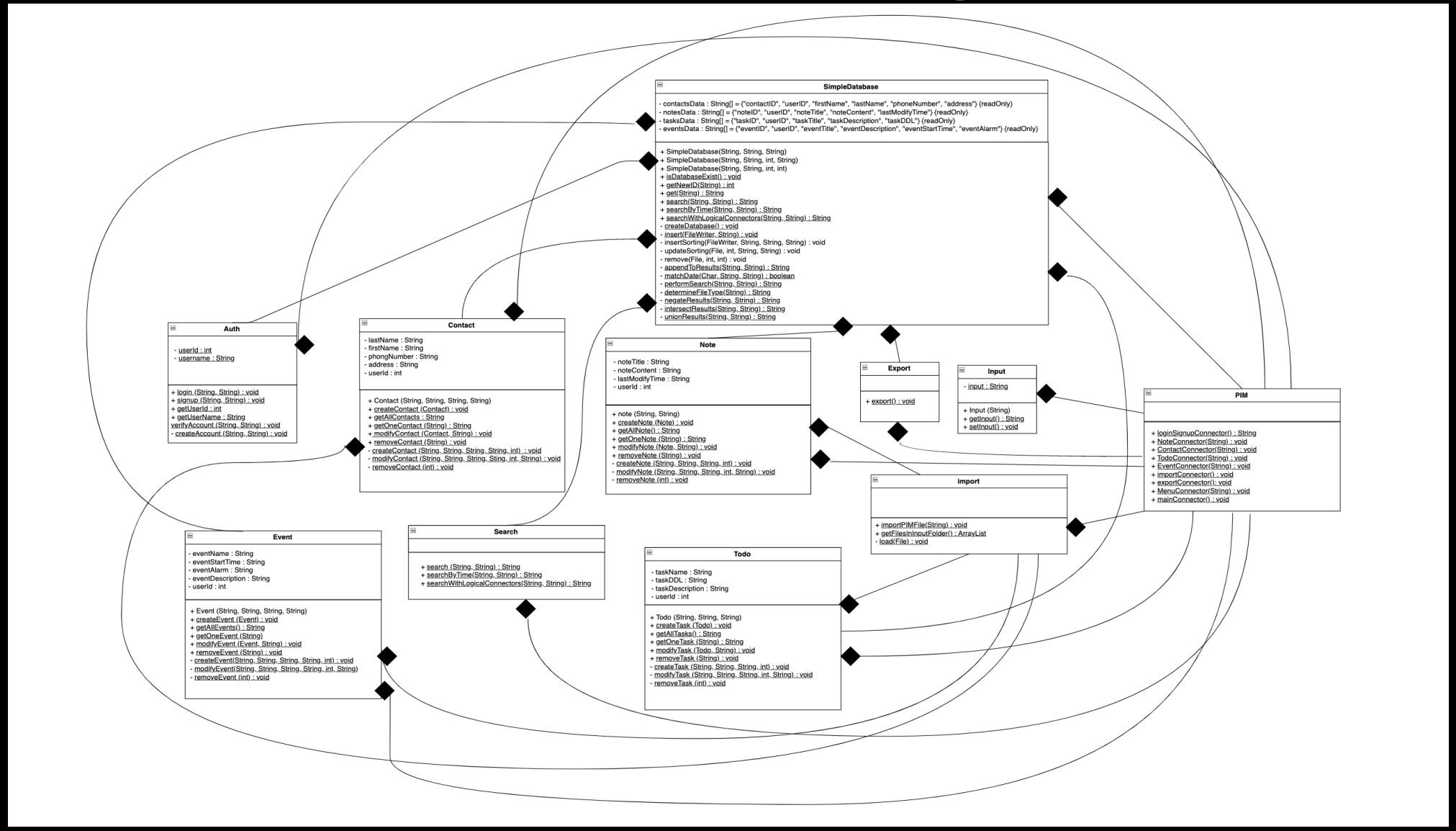
```
Searching Keyword(s): Hello
```

```
Searching by Time: = 2023-11-25 10:30:00
```

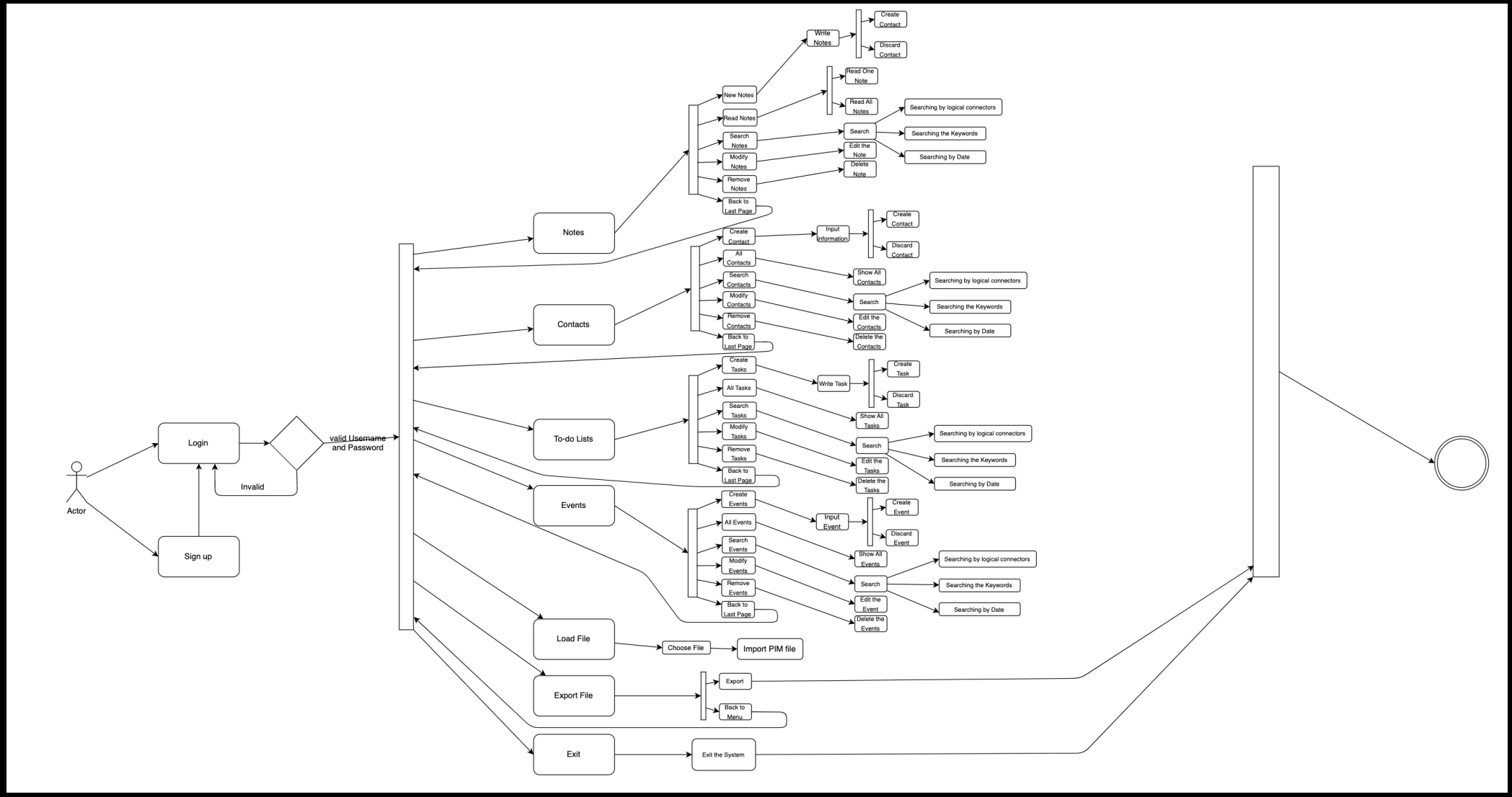
```
Searching with logic connector(s): Hello && > 2023-11-25 09:00:00
```

Overall Design & Architecture

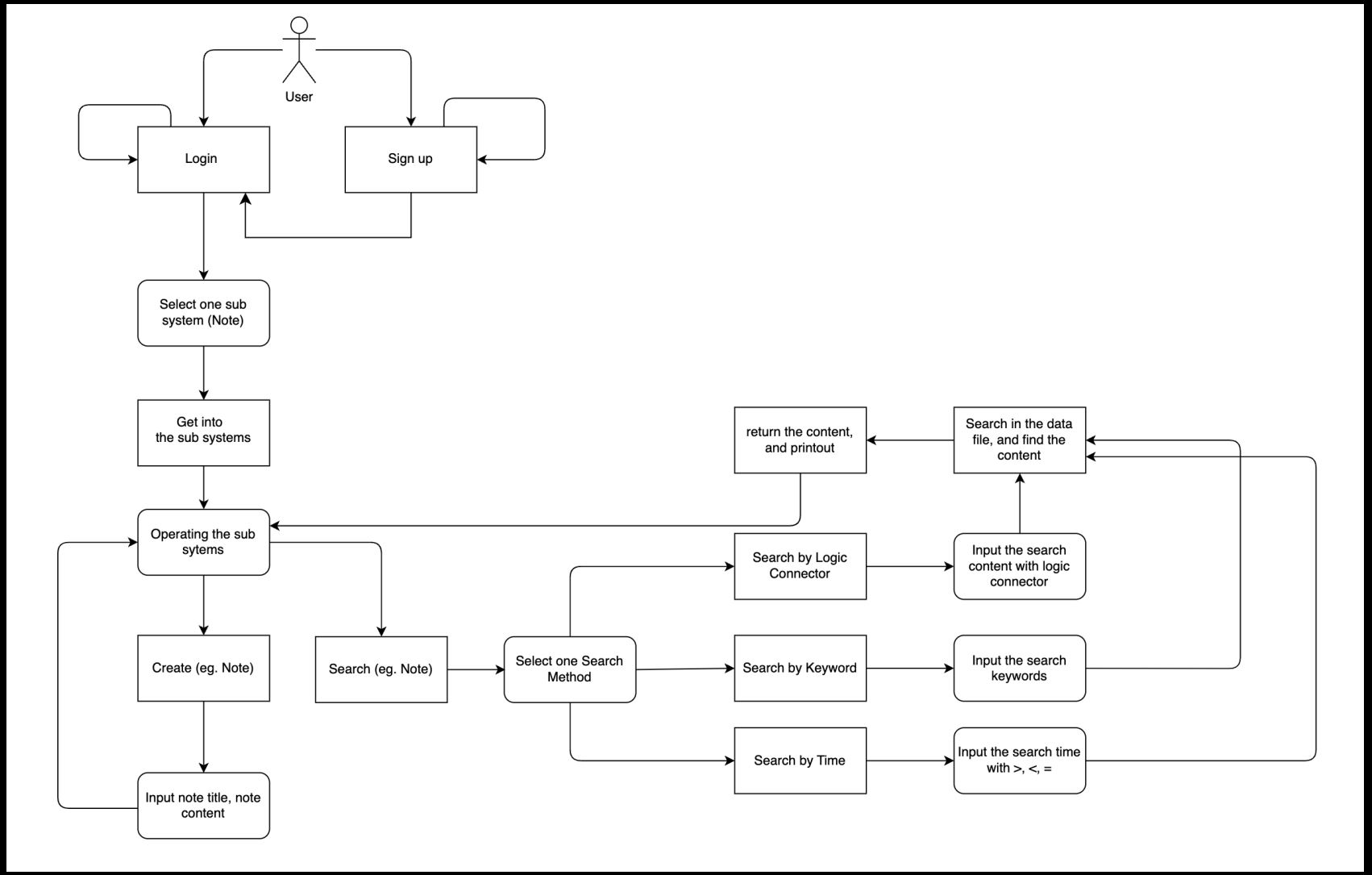
UML Class Diagram



UML Activity Diagram



UML Activity Diagram (Searching)



Data Store Design => CSV File (Table Form)

[User]

userID	userName	password

[Events]

eventID	userID	eventTitle	eventDescription	eventStartTime	eventAlarm

[Contacts]

contactID	userID	firstName	lastName	phoneNumber	address

[Notes]

noteID	userID	noteTitle	noteContent	lastModifyTime

[Tasks]

taskID	userID	taskTitle	taskDescription	taskDDL

“Database” API Interface

[Insert]

```
new SimpleDatabase("insert", FILE_NAME, DATA_STRING_2D_ARRAY[][]);
```

[Update]

```
new SimpleDatabase("update", FILE_NAME, USER_ID, DATA_STRING_ARRAY[]);
```

[Remove]

```
new SimpleDatabase("remove", FILE_NAME, USER_ID, CLASS_ID);
```

[Get]

```
get(String fileName);
```

[Get New ID]

```
getNewID(String fileName);
```

“Search” API Interface

[Search by Keyword]

search(String keyword, String fileType);

[Search by Time]

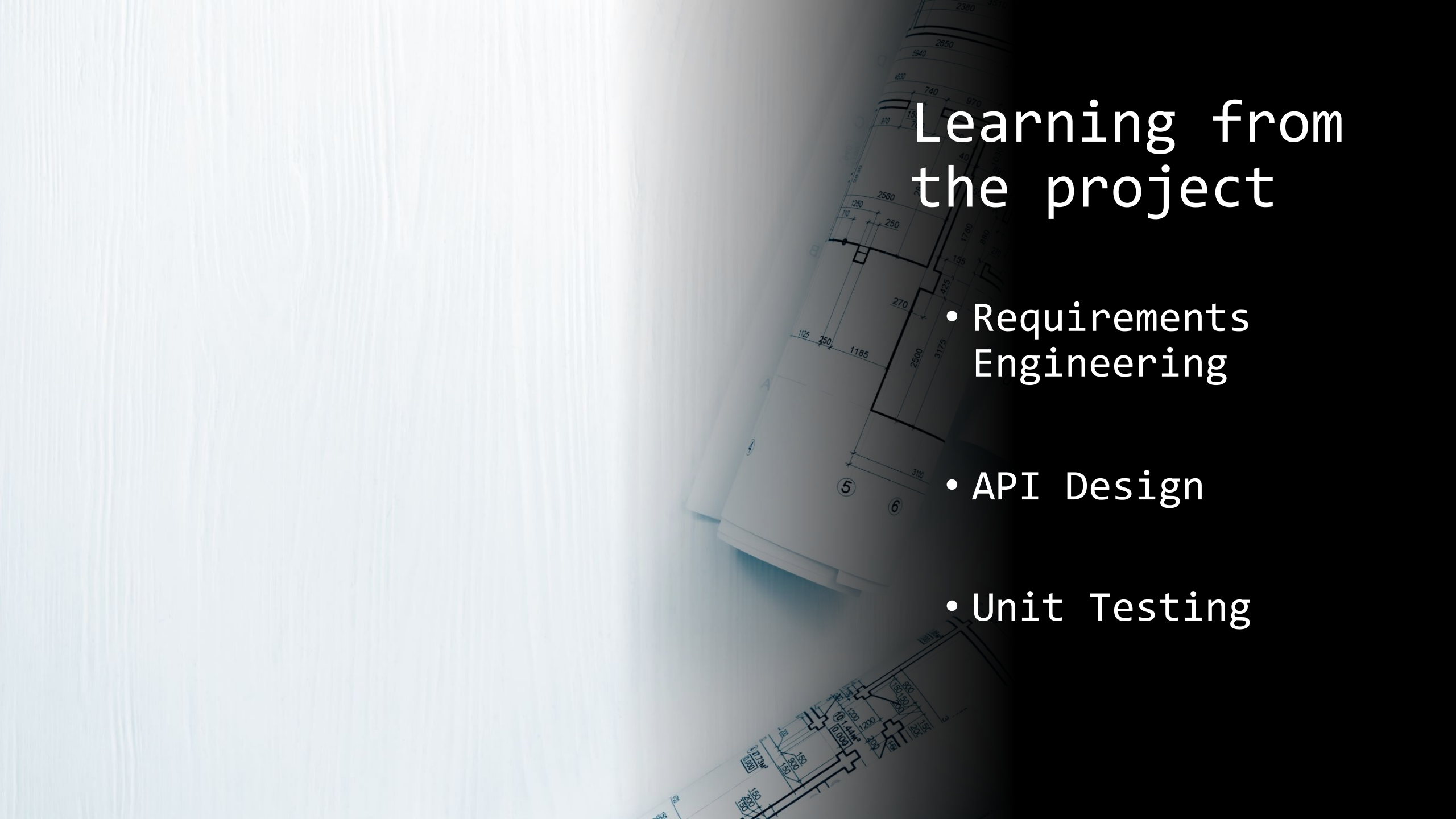
searchByTime(String inputTime, String fileType);

[Search with Logic Connector]

searchWithLogicalConnectors(String expression, String type);

Code Structure

```
├── controller
│   ├── Auth.java
│   ├── Contact.java
│   ├── Event.java
│   ├── Export.java
│   ├── Import.java
│   ├── Input.java
│   ├── Note.java
│   ├── PIM.java
│   ├── Search.java
│   └── Todo.java
├── model
│   └── SimpleDatabase.java
└── view
    └── Pages.java
```



Learning from the project

- Requirements Engineering
- API Design
- Unit Testing

Requirements Engineering

- It is crucial to have Clear and Complete Requirements :
- It plays a crucial role in software development by ensuring that the software meets the needs and expectations of its user.
- By finishing the requirements design, it is clear to separate the total work into 4 project members and gradually accomplish the project according to the plannable timeline.

API Design

- It is very important to keep the API simple and clear
- Good API Style can reduce the workload of code connection

Unit Testing

- By achieving high test coverage and conducting integration testing, we can uncover and resolve errors and logic flaws that may have gone unnoticed during code writing.

- This contributes to improving code quality, enhancing system stability, and reducing the risk of more severe issues.

Q&A

- One-minute for answering question