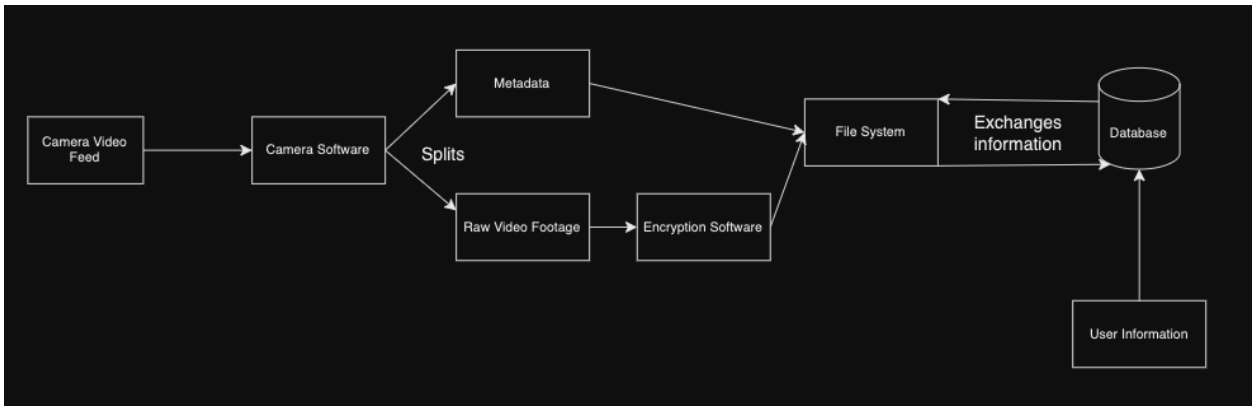
Design Choice

For this project, we have decided to use SQL database to store all of the encrypted and nonencrypted data. It is generally considered bad practice to store files, particularly video ones, directly in the table as they take up large amounts of data. To allow the system to run much more efficiently, the main camera networks' computers will contain a proprietary file system, where the encrypted video feed, associated metadata, and account information will be stored. The file path of each video will be fed into the database, and when a user attempts to call the file, they will be prompted by the software to enter the encryption password. This design choice replaces our previous plan to contain two separate databases and allows for encryption to be handled purely by the file system and its software.

SQL is the best database language for this purpose because there is a large amount of documentation and examples for such a simple process. Using the existing documentation, the development team will be able to make systems that will require as little maintenance as possible. Our developmental process for this will focus on reliability and sustainability. Each camera system will be local, so there is no need for a separate database for account information. Instead, the database will have a small part of it partitioned off purely for account information. Separating the data this way increases security in the case of a breach, while still allowing the system to act efficiently. Each entry will store account ID, username, passwords, and authorization level. Those with high enough authorization will be permitted to access the encrypted video data.



Database: video

video_Recording (video file)	data(String)	time(double)	cameraID(int)
34de3557.mp4	“4531dvfaesfw”	9.30	9821
12dm5571.mp4	“fs3as3dg44gs”	23.15	4521
23re3557.mp4	“32dfg1s4sg34”	4.00	2357
67ma4723.mp4	“2g431gbas3vq”	12.25	4368

Search: data, camera
Description: For our video storage database, we will split up the data between video_recording, data, time, and cameraID. The video recording will store the encrypted file of the video recording from one of the security cameras. The data variable will hold the metadata of the video including file type, size, and resolution. The time variable will store the time at which the video was recorded and the cameraID variable will store the ID of the camera that the video was recorded from.

Database: Account Information

Account ID(int)	Username(String)	Password(String)	Authorizations (String)
91912321	“joesif”	“gO0lish”	“Company”
82345164	“randall”	“1234”	“Customer”
31256721	“joseph”	“password	“Customer”
49287615	“dan”	“@s1f1t”	“Admin”

Search: Username, Account ID
Description: For our account information database, we split up the data between accountID, username, password, and authorizations. The accountIID variable will hold the ID of the account of a user. The username variable will hold the username of the user associated with the account ID and the password variable will hold the password for the account. The authorizations variable will hold the type of account the user has so that the types of functions they can use are properly assigned to them.

Possible tradeoffs

Both SQL and noSQL frameworks could be used for storage in this system. We have decided on SQL because it has years of documentation from Microsoft and its users on how to safely transmit and store information for video files and meta data. Due to our focus on reliability, we think the SQL choice will be the safest.