**Why do we need a Sewa Management System ?**

We have an Attendance machine which gives us data in the form of a log/csv file. It is difficult to get reports out of it and we need to calculate the count of attendance everytime we need to. Hence we need a system where anyone can see the tenant’s attendance online and generate reports.

**Requirements and goal of the system**

Our system should meet the following requirement :

**Functional Requirements**

1. Users should be able to upload the file received from the Attendance Machine/Sewa Samiti and the system should process the attendance and save data in respective tables.
2. Users should be able to generate reports out of it.
3. API to add/update/delete the individual sewardar Attendance.
4. RFID to Sewadar Code mapping
5. Pre-validation of CSV files.
6. Multiple files upload
7. Saving file in multiple storage
8. Processing CSV files and time out on web apps

**Non Functional Requirements**

1. Data should not take much time to process and in case of any error user need not to re-process the file again.

**Extended Requirements (To be decided)**

1. Observe how much time a file is taking to process based on the size and how many times a user uploads a file monthly/yearly etc.
2. How much time reports take to generate.

**Load in the system**

Our system will be read and write heavy as we need to save the bulk attendance into the tables and also to generate the reports.

**Database Design**

**Terminology used**

RFID : It's a unique batch id written on the back of the card.

1. **user\_attendance\_records :** Contains attendance records of all users.

| user\_attendance\_records | |
| --- | --- |
| id | Unique id |
| user\_id | String |
| In\_time | datetime |
| out\_time | datetime |
| duration | Integer |
| attendance\_process\_status\_id | Integer(fk to attendace\_process\_status**)** |
| attendace\_date | date |
| updated\_by | SYSTEM/User |
| updated\_at | datetime |

**Index on attendance\_date for faster retrieval.**

Attendance\_data if we think it will reduce our time in reports as we have to use sql function to grab the date part and get reports.

Updated\_by : User : Means it’s a manual addition/update.

1. **user\_rfid\_mapping :** this is the active userid and rfid mapping table

| user\_rfid\_mapping | |
| --- | --- |
| user\_id | String |
| rfid | String |

1. **User\_rfid\_history\_mapping :** Contains records of rfid which were updated.

| user\_rfid\_history\_mapping | |
| --- | --- |
| id | Unique id |
| user\_id | String |
| rfid | String |
| updated\_by | User id |
| updated\_at | datetime |

1. **attendace\_process\_status :** Once file is uploaded and initial checks are done , we will start asynch process.

| attendace\_process\_status | |
| --- | --- |
| id | Integer |
| file\_path | String |
| file\_hash | String |
| uploaded \_by | User id |
| uploaded\_at | datetime |
| status | UPLOADED/IN\_PROCESS/PARTIAL\_SUCCESS/FAILED |
| pending\_records\_file\_path | String |

File\_hash : Unique file be uploaded always.

Pending\_records\_file\_path : Error records will be copied to this table and given to user.

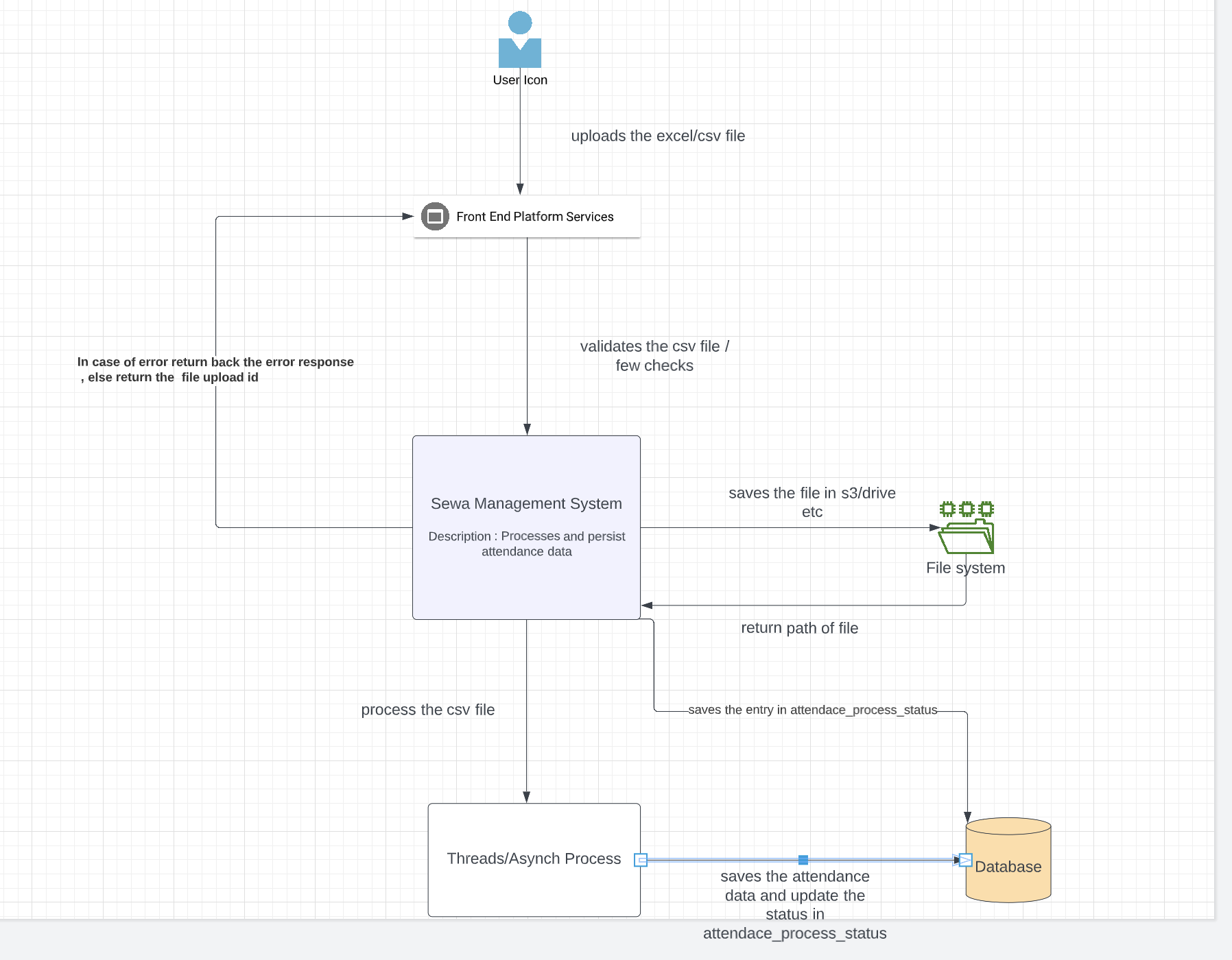
1. **activitiy\_log (To be decided) :** user activity table

| user\_activity\_log | |
| --- | --- |
| id | Integer |
| user\_id | String |
| action | String |
| time | datetime |

1. **User\_attendance\_archive\_data :** Record which will be least used like of last year.

| user\_attendance\_archive\_data | |
| --- | --- |
| id | Unique id |
| user\_id | String |
| In\_time | datetime |
| out\_time | datetime |
| duration | int |
| file\_path | String |
| attendace\_date | date |

**Suggested design**

****

1. Sewa Management system will do initial checks of the format of the file. If there is error return the error response else create an entry in attendance\_process\_status and return the id the user.
2. Management system will start a asynch process where it reads the file and start processing and saves the data in database.
3. Records not being saved will be added to another csv file so that user need not to manually filter the error records and can work on the other csv / rectify it and reupload the pending data.
4. In case same file is reuploaded , checks the hash and sends back error.

**Open question for discussion**

**1. What if the uploaded file has the same records of date which are already in the system ?**

Suggestion : attendance\_date column can be used here , where we have to check if record for the same day exist or not , if it does update the record else send that record to pending\_records\_file

**2. What if system gets stuck and file process hangs in between ?.**

Suggestion : We can save all the data once the whole file process ends else it will be difficult to filter the data.

We can have a scheduler/ task handler which runs every 15 minutes and checks if there is any file with pending status and it’s in same status since more than configurable time , if yes it will pick and start processing,

**3. Shall there be any limit of file size ?**

Depends on how much initial checks we want to do when we upload a file.

**API’s**

| **API** | **Description** | **Type** | **Request Body** | **Response** |
| --- | --- | --- | --- | --- |
| /v1/uploadFile | Upload attendance file, in case of error sends error response else send the attendace\_process\_status\_id for status. | POST | {  “file” : “”  } | {  “Attendace\_process\_status\_id” :  } |
| /v1/attendance | Manual saving of attendance of a user. | POST | {  “User\_id” :  “In\_time”  “out\_time”  } | {  success/failure  } |
| /v1/attendance/{id} | Updating attendance of user | PATCH | {  “User\_id” :  “In\_time”  “out\_time”  } | {  success/failure  } |
| /vi/attendance/{id} | Deleting attendance of user | DELETE |  | {  success/failure  } |
| /v1/attendance | Getting attendance to users with different params. | GET | Request param  Start\_date  End\_date  Attendace\_date  user\_id | List of attendance |
| /v1/attendance-process | To see the file upload status | GET |  | Attendance process data |
| /v1/reports(to be discussed) | Get multiple reports | GET | Param |  |