Google Summer of Code 2020
Project Proposal

OpenMF
Android Forensics Tool

OpenMF

CORE
Sustainable Computing Research

- Create web client for OpenMF

Basic Details:

- **Name:** Shivanshu Raj Shrivastava
- **Email:** shivanshu1333@gmail.com
- **GitHub:** shivanshu1333
- **LinkedIn:** Profile Link
- **Contact:** +91 9425646127
- **GPG Fingerprints:** 42B2 92EC F055 51E7 1CF0 9C84 F120 B48B 2677 FF0B

Project Introduction:

A. Problem

OpenMF is an open-source forensic tool for Android smartphones that helps digital forensic investigators throughout the life cycle of digital forensic investigation.

For e.g. Let us say we have a crime scene in which we have captured some suspects and we have their mobile phones. If we want to extract all the data from their phones and see which of them are actually involved in the crime scene then we require software to perform this task and produce meaningful evidence and analysis reports for every phone (Digital forensic case).

The OpenMF project is a dedicated software to:

1. Extract the relevant data
2. Manage all the cases separately
3. Produce Analysis report

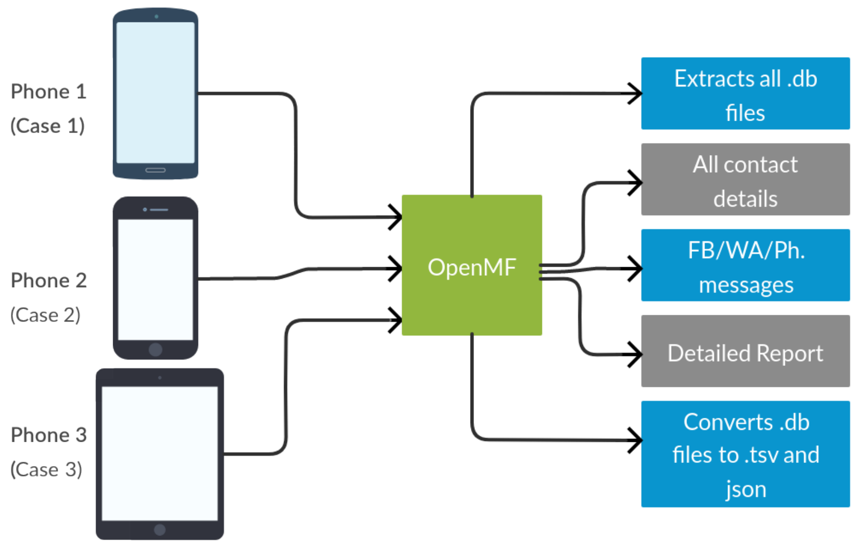


Fig. 1- OpenMF is used from the command line to extract data from different phones (cases)

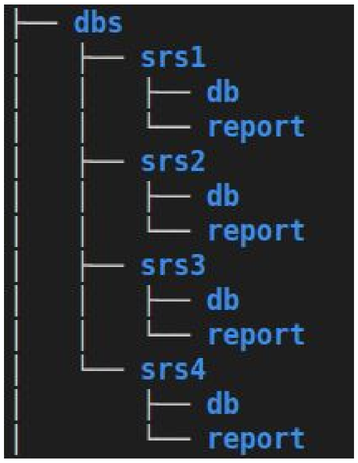
B. Current State

At present OpenMF is a command-line tool capable of useful data extraction and very basic report generation of the given cases (phones). That is if you attach a rooted android device and run `python3 collector.py --option all --session_name CaseNo1`, all extracted databases (.db files) of the phone will get stored in a folder name `CaseNo1`, where `CaseNo1` is a name given to that particular phone (case).

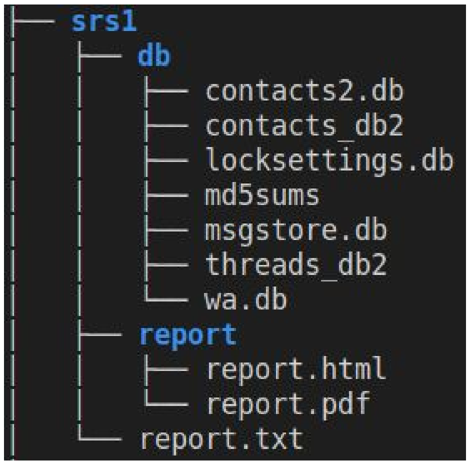
Here we can provide the following options:-

1) **all** : Collect all common dbs
2) **general_info** : To collect general device information
3) **phone** : To collect mobile contacts, call logs and messages information
4) **whatsapp** : To collect WhatsApp message information and contact details
5) **facebook** : To collect facebook messages and profile information

We can also generate a case report by running `python3 collector.py --report CaseNo1`
Additionally `.db` files can also be converted into `.tsv` files to easily read the data. This is done by running `python3 converter.py 'path to case files' 'path to destination folder'`.



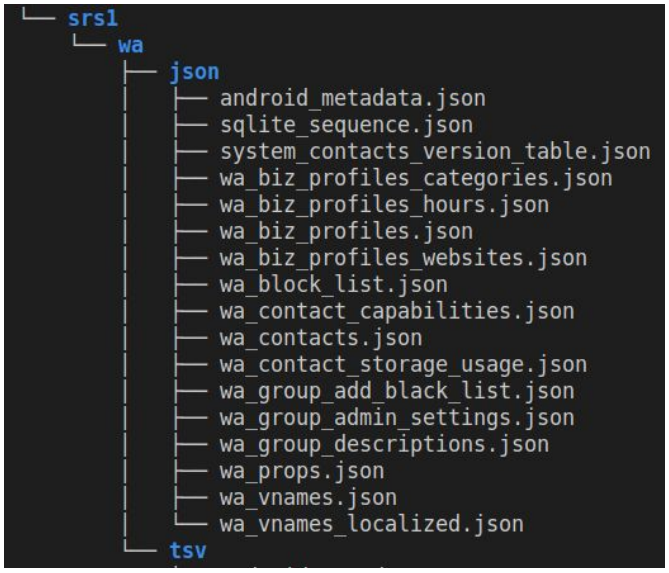
```
├── dbs
│   ├── srs1
│   │   ├── db
│   │   └── report
│   ├── srs2
│   │   ├── db
│   │   └── report
│   ├── srs3
│   │   ├── db
│   │   └── report
│   └── srs4
│       ├── db
│       └── report
```



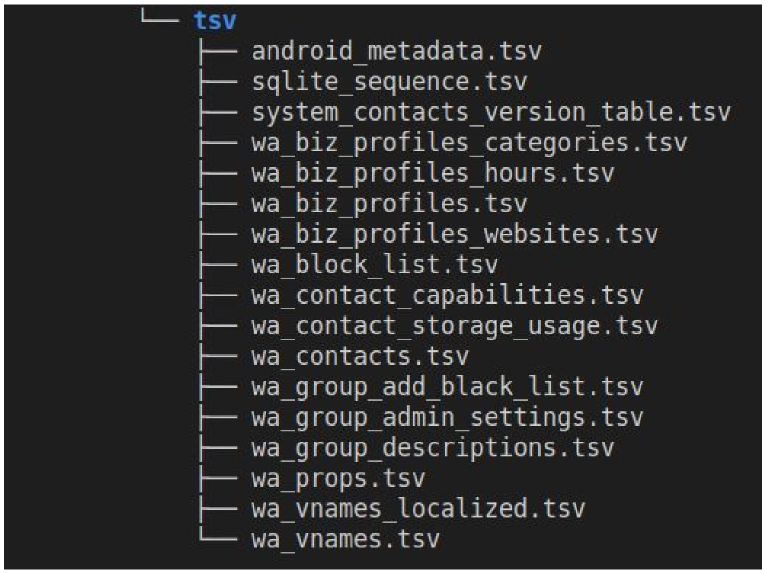
```
├── srs1
│   ├── db
│   │   ├── contacts2.db
│   │   ├── contacts_db2
│   │   ├── locksettings.db
│   │   ├── md5sums
│   │   ├── msgstore.db
│   │   ├── threads_db2
│   │   └── wa.db
│   ├── report
│   │   ├── report.html
│   │   └── report.pdf
│   └── report.txt
```

Fig 2- a) Shows extracted dbs from different phones(cases with name srs1, srs2 etc)

Fig. 2- b) Shows detailed structure of extracted data (if option all is used, 260+ databases are extracted)



```
└── srs1
    └── wa
        ├── json
        │   ├── android_metadata.json
        │   ├── sqlite_sequence.json
        │   ├── system_contacts_version_table.json
        │   ├── wa_biz_profiles_categories.json
        │   ├── wa_biz_profiles_hours.json
        │   ├── wa_biz_profiles.json
        │   ├── wa_biz_profiles_websites.json
        │   ├── wa_block_list.json
        │   ├── wa_contact_capabilities.json
        │   ├── wa_contacts.json
        │   ├── wa_contact_storage_usage.json
        │   ├── wa_group_add_black_list.json
        │   ├── wa_group_admin_settings.json
        │   ├── wa_group_descriptions.json
        │   ├── wa_props.json
        │   ├── wa_vnames.json
        │   └── wa_vnames_localized.json
        └── tsv
```



```
└── tsv
    ├── android_metadata.tsv
    ├── sqlite_sequence.tsv
    ├── system_contacts_version_table.tsv
    ├── wa_biz_profiles_categories.tsv
    ├── wa_biz_profiles_hours.tsv
    ├── wa_biz_profiles.tsv
    ├── wa_biz_profiles_websites.tsv
    ├── wa_block_list.tsv
    ├── wa_contact_capabilities.tsv
    ├── wa_contact_storage_usage.tsv
    ├── wa_contacts.tsv
    ├── wa_group_add_black_list.tsv
    ├── wa_group_admin_settings.tsv
    ├── wa_group_descriptions.tsv
    ├── wa_props.tsv
    ├── wa_vnames_localized.tsv
    └── wa_vnames.tsv
```

Fig. 2- C) Shows extracted WhatsApp data in JSON and tsv format (all option can also be used)

C. Issues to be resolved

The command-line tool in general is not user friendly to perform any task. Web client allows an alternative easier way. Thus a web client is required for OpenMF which will also help in adding some more functionalities to the existing tool.
The issue which will be covered during GSoC 2020 is to **add support for web client** for this command-line tool with additional features like **Admin, Management** and **Extractor** roles. These roles will have different privileges to the tool and as a whole, it will become a complete open-source forensic tool for Android smartphones which then can be used by any organization.
The web client should be capable of providing services like:

- Digital forensic case and evidence management
- Raw data acquisition – physical acquisition and logical – file system level acquisition
- Meaningful evidence extraction and analysis support
- Evidence presentation

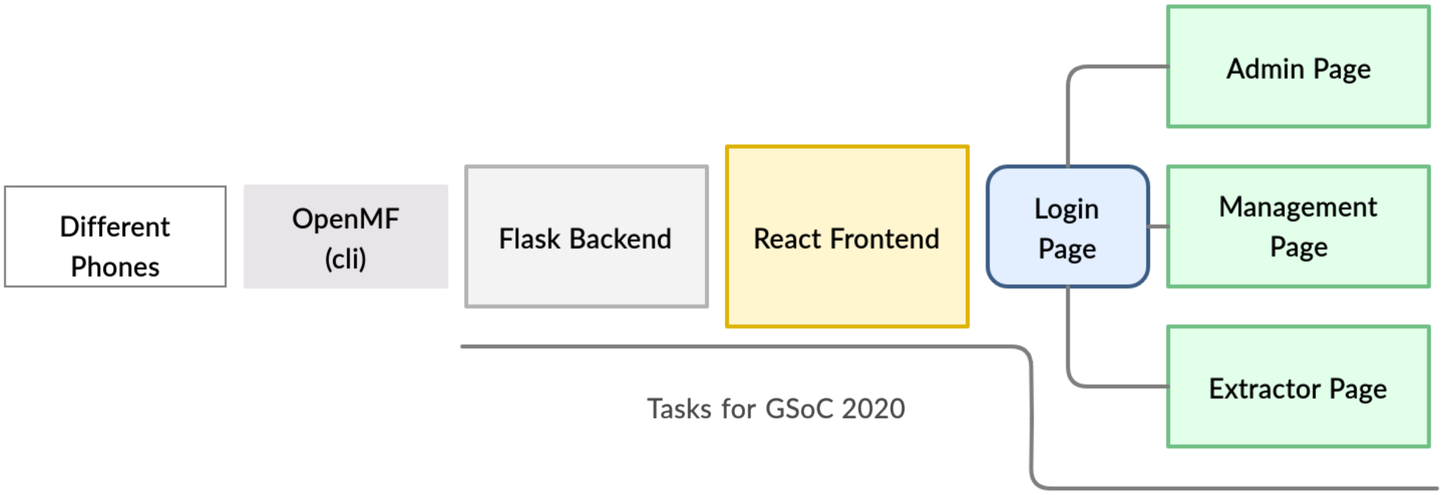


Fig. 3- Flow chart of OpenMF after implementing Web Client

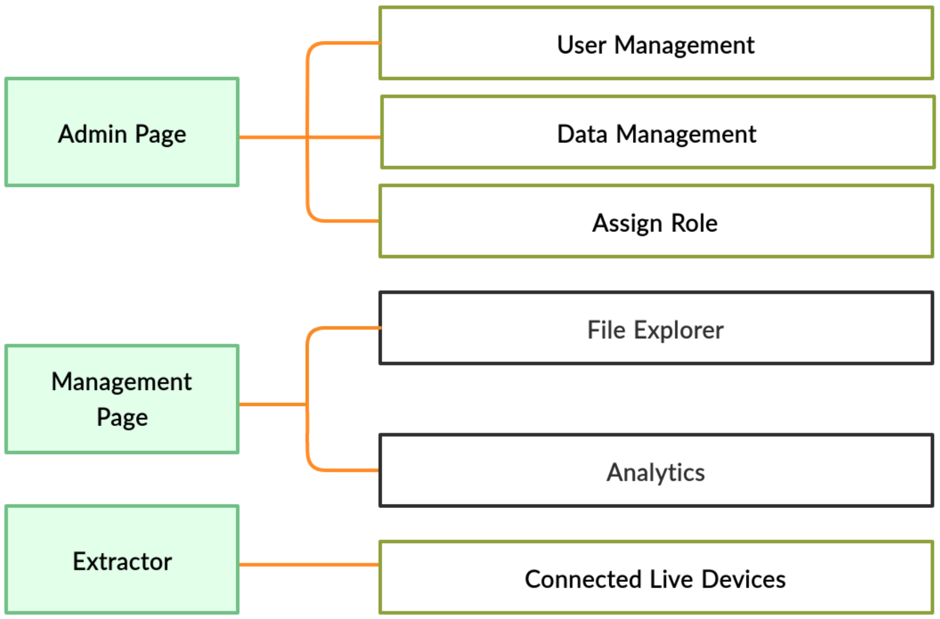


Fig. 4- Description of Different Roles in Web Client

Project Goals:

Deliverables:-

- Modify the current code base to make a strong robust command-line tool for OpenMF.
- Design and Implement a Client-side Framework for Web Client.
- Design and Implement Login, Admin, Management and Extractor pages (refer Fig. 3-4).
- Implementation of backend from scratch and writing different APIs in the flask framework.
- Implementation of different roles (Admin, Management, Extractor) from scratch.
- Extending the functionality of OpenMF to include download, upload, and management of files on the server.
- Implementation Access Control Lists (ACL) to manage file permissions among users by Admin.
- Design and implement a Database model, relation mapping (ORM) in the database to incorporate the above functionalities in MySQL using the SQLAlchemy toolkit.
- Write a project wiki and update the user and developer documentation.
- Weekly blogs on medium.

Implementation:

I have divided the coding phase in 3 major phases.

- Phase one involves the implementation of the frontend for Web Client.
- Phase two involves the Implementation of a flask server, writing different APIs, defining different routes.
- Phase three involves the Implementation of different roles, file management on the server, ACL implementation and implementation of the Database model.

Description of different phases:

Phase 0 (4 May - 1 June):

- Community bonding and finalize project milestones.
- Modification in the current code base to make a strong robust command-line tool for OpenMF
- Documentation of command-line tool for OpenMF

Phase 1 (1 June - 3 July):

- Design and Implement frontend for Web Client.
- Design frontend as a single page application in React.
- Design and Implement Login, Admin, Management and Extractor pages (refer Fig. 3-4).

Designs of different pages are shown below (Phase 1) :-

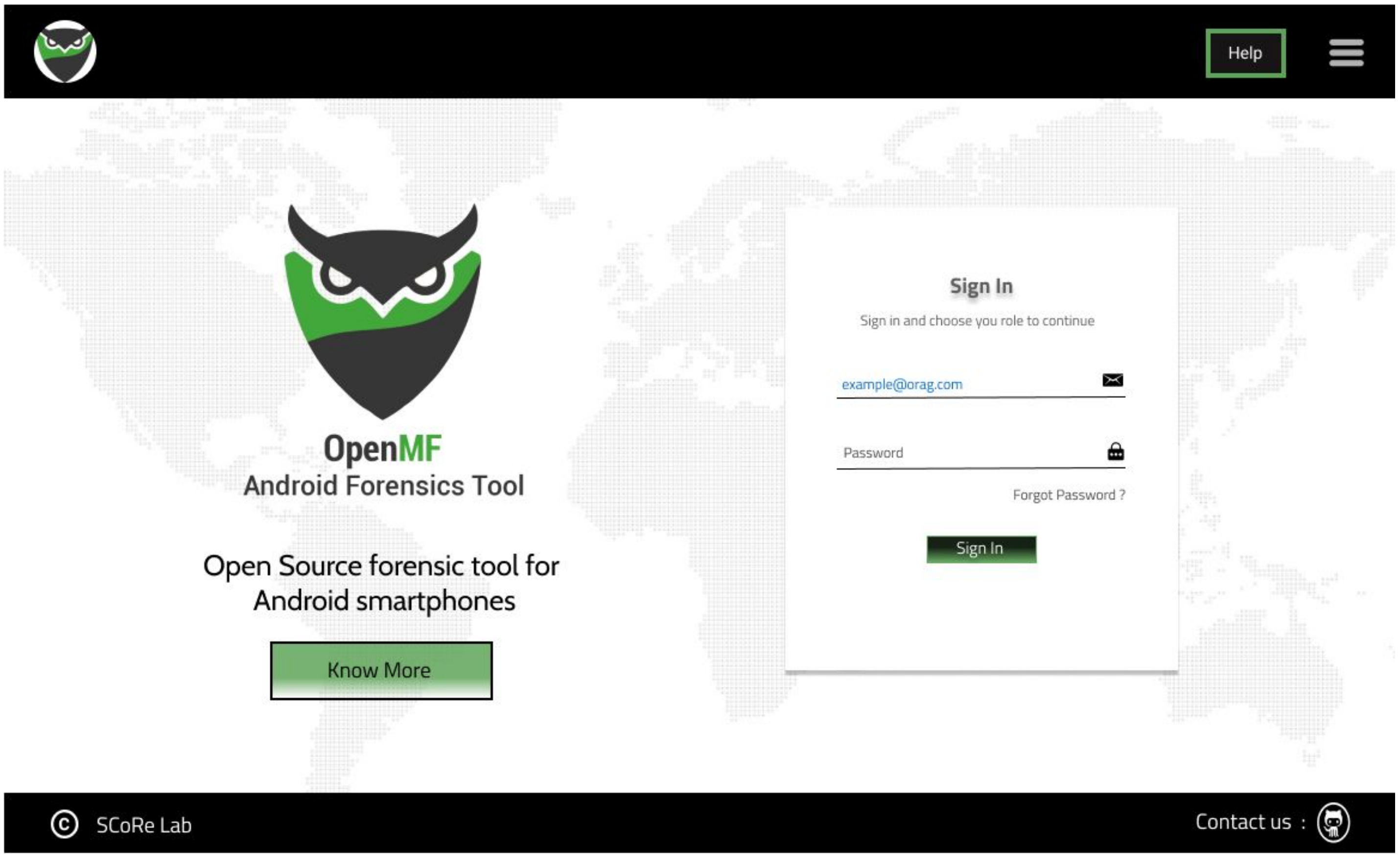


Fig. 5- Landing page (Login page) design for OpenMF

Description of this page:-

When a user login using their Email-Id and Password, a pop-up will open which will ask them to choose a role (Roles are assigned by admin).
According to their selection, the corresponding page will open.

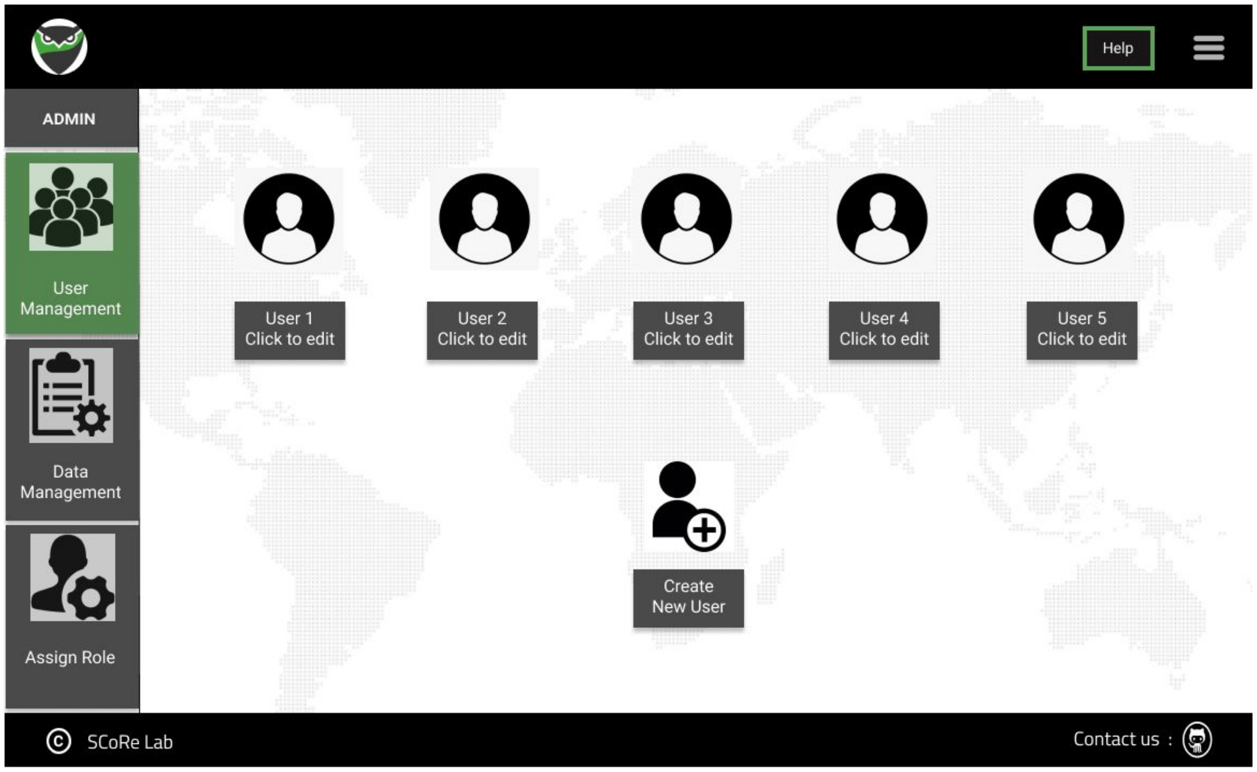


Fig. 6- Admin page design for OpenMF, (a) User Management

Description of this page:-

An admin has a privilege to Manage existing users, Create new users and Delete existing users.

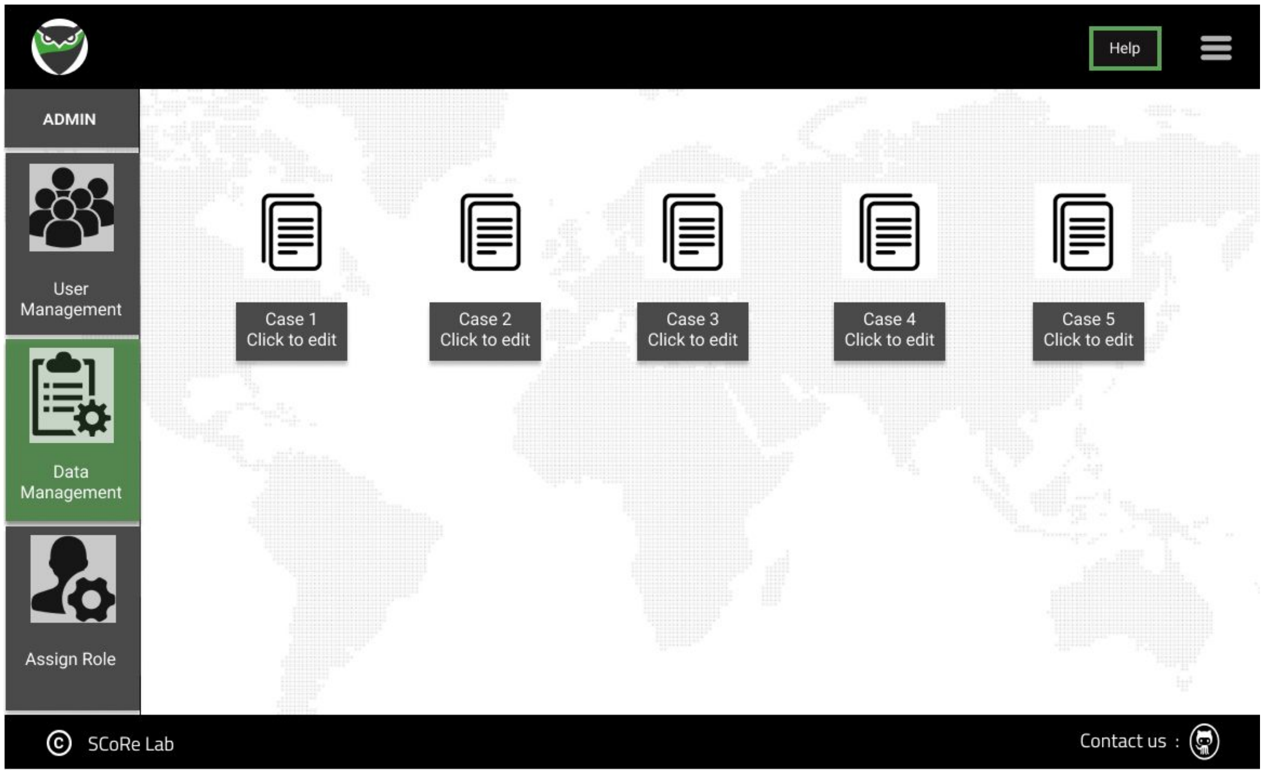


Fig. 7- Admin page design for OpenMF, (b) Data Management

Description of this page:-

An admin has a privilege to View and Delete existing data on the server.

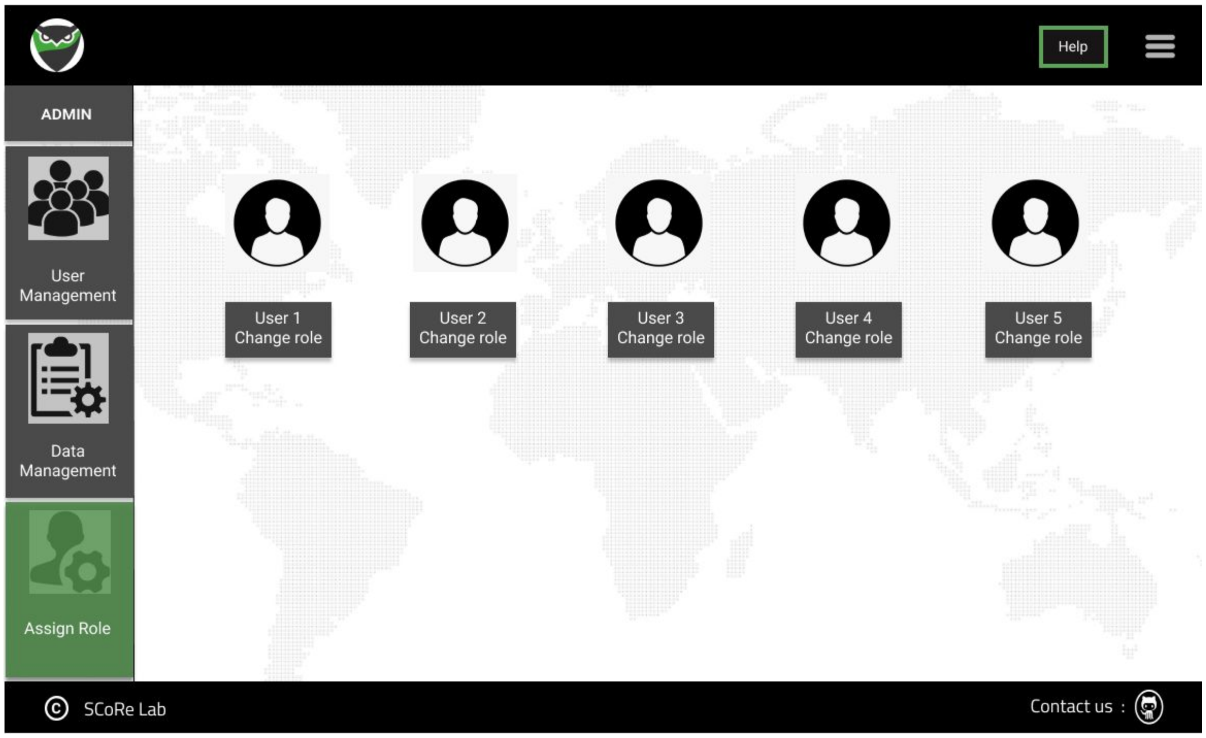


Fig. 8- Admin page design for OpenMF, (C) Assign Role

Description of this page:-

An admin has a privilege to Add a role to any user as well as Delete the role of any user.

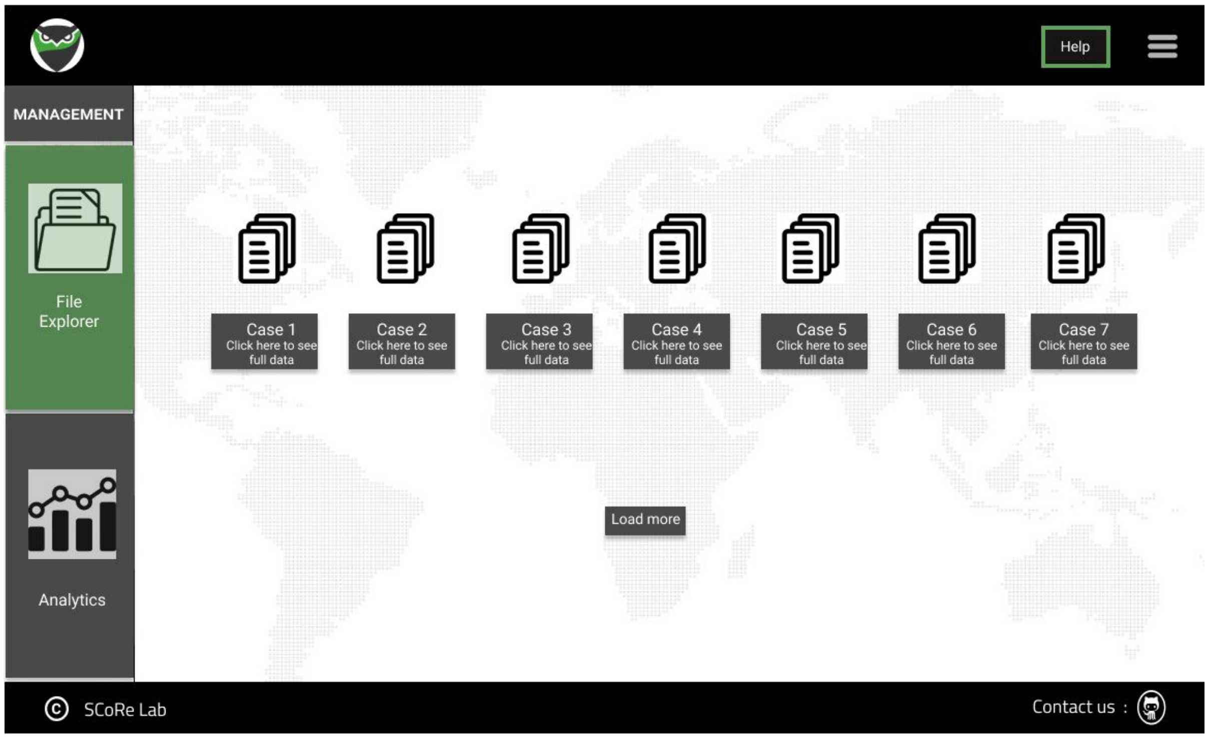

Help
MANAGEMENT
File Explorer
Analytics
Case 1 Click here to see full data
Case 2 Click here to see full data
Case 3 Click here to see full data
Case 4 Click here to see full data
Case 5 Click here to see full data
Case 6 Click here to see full data
Case 7 Click here to see full data
Load more
SCoRe Lab
Contact us :

Fig. 9- Management page design for OpenMF, (a) File Explorer

Description of this page:-

A user with Management credentials has the ability to see the detailed data of all the cases. The file explorer will have ability to see particular **.db files** in **.tsv format**, which will be displayed in WebClient.

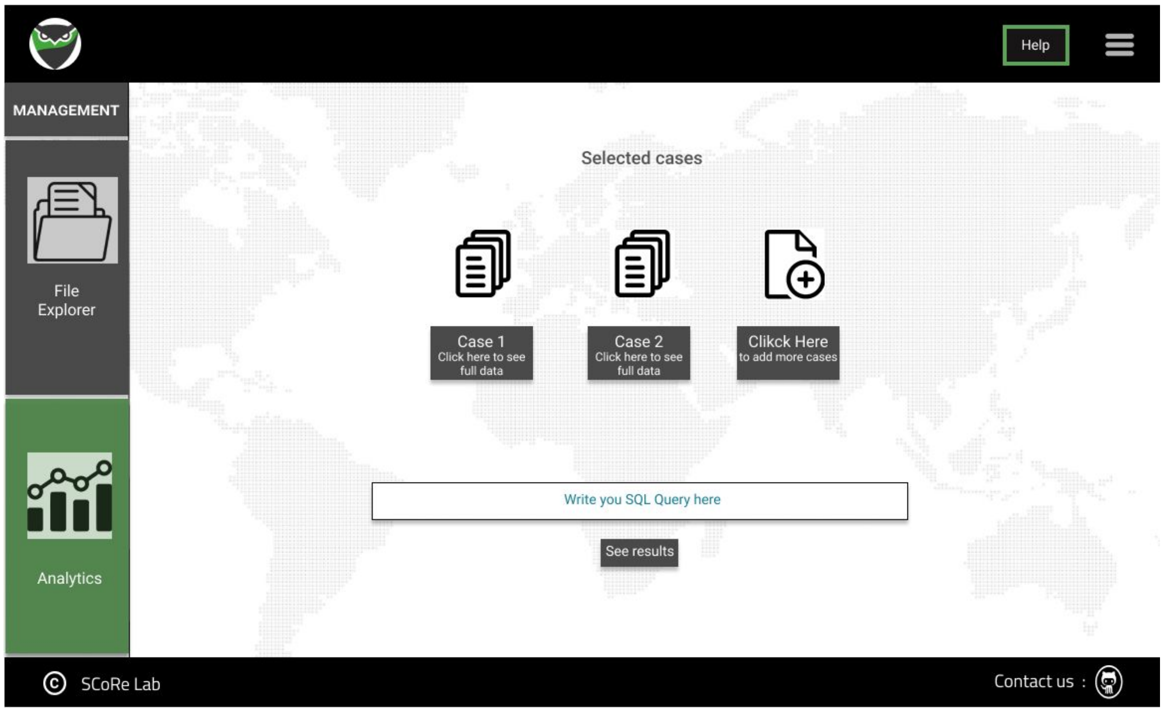


Fig. 10- Management page design for OpenMF, (b) Analytics

Description of this page:-

Analytics can be performed on cases by selecting and writing a SQL Query on them

---

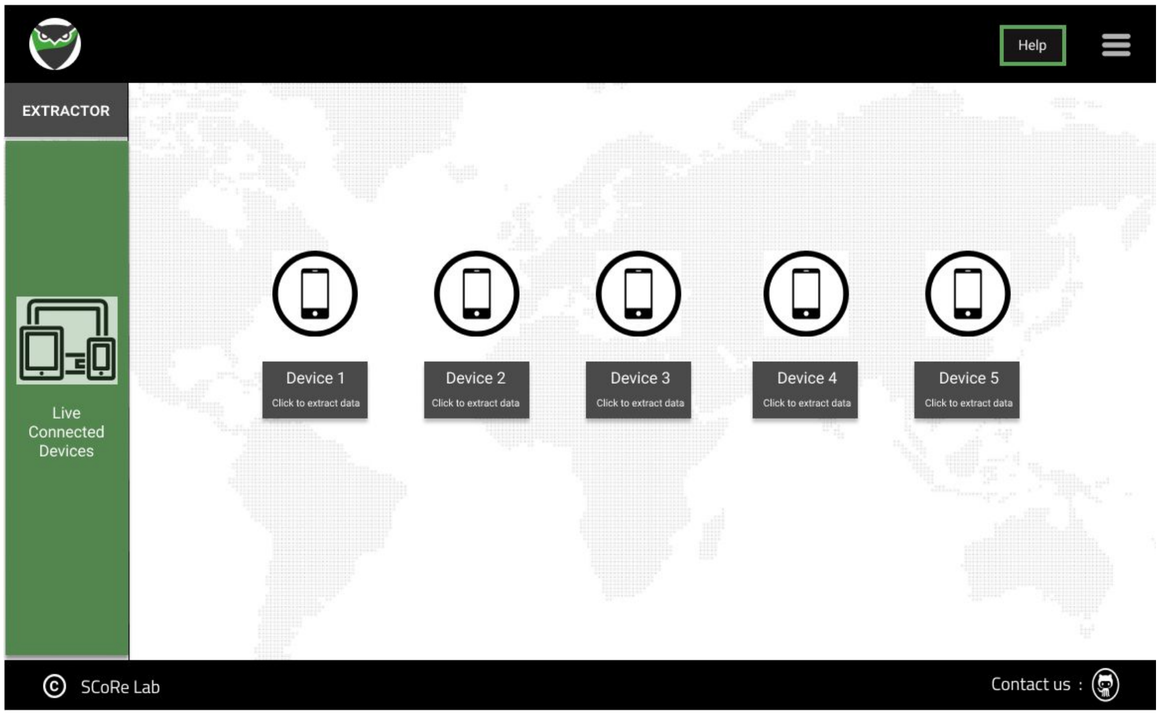


Fig. 11- Extractor page design for OpenMF, (a) Connected Live Devices

Description of this page:-

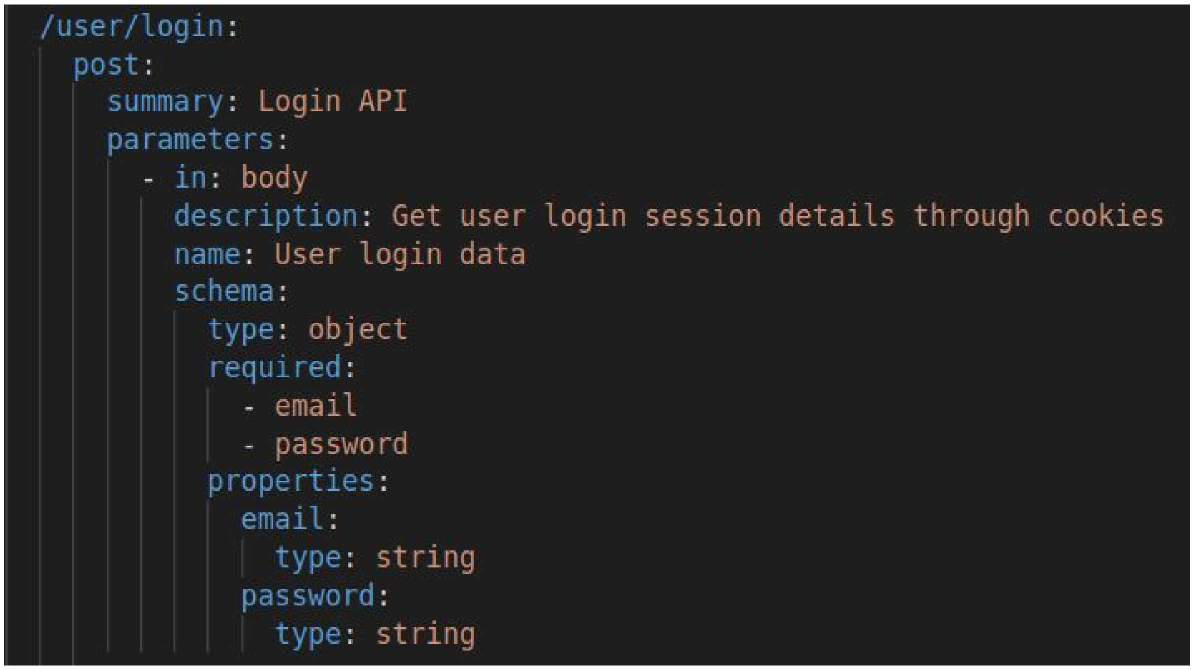
A user with Extractor credentials has the ability to see all the live devices connected. Extractors can then choose an option by clicking on the button to extract various data just like the command line and can save them on the server.

Phase 2 (3 July - 31 July):

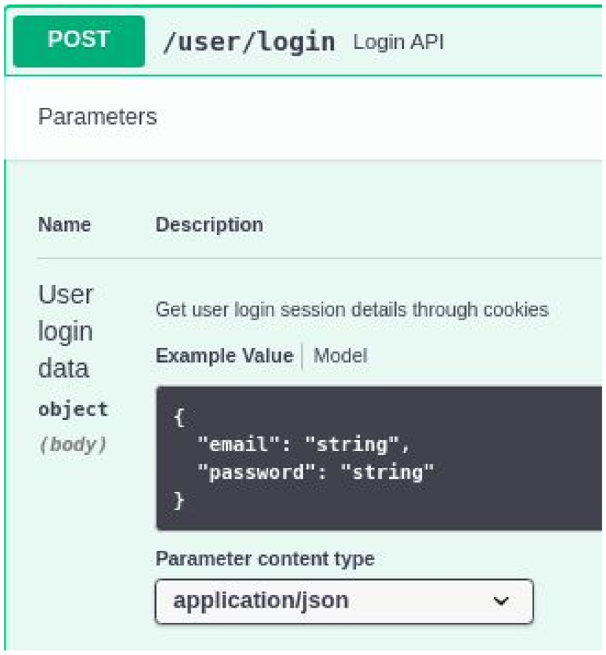
- Implementation of backend from scratch and writing different APIs in the flask framework.
- Implementation of different roles (Admin, Management, Extractor) from scratch.
- Extending the functionality of OpenMF to include download, upload, and management of files on the server.
- Implementation Access Control Lists (ACL) to manage file permissions among users by Admin.

<u>Designs of different APIs are written below (Phase 2) :-</u>

- **User related APIs -**



```
/user/login:
  post:
    summary: Login API
    parameters:
      - in: body
        description: Get user login session details through cookies
        name: User login data
        schema:
          type: object
          required:
            - email
            - password
          properties:
            email:
              type: string
            password:
              type: string
```



POST /user/login Login API

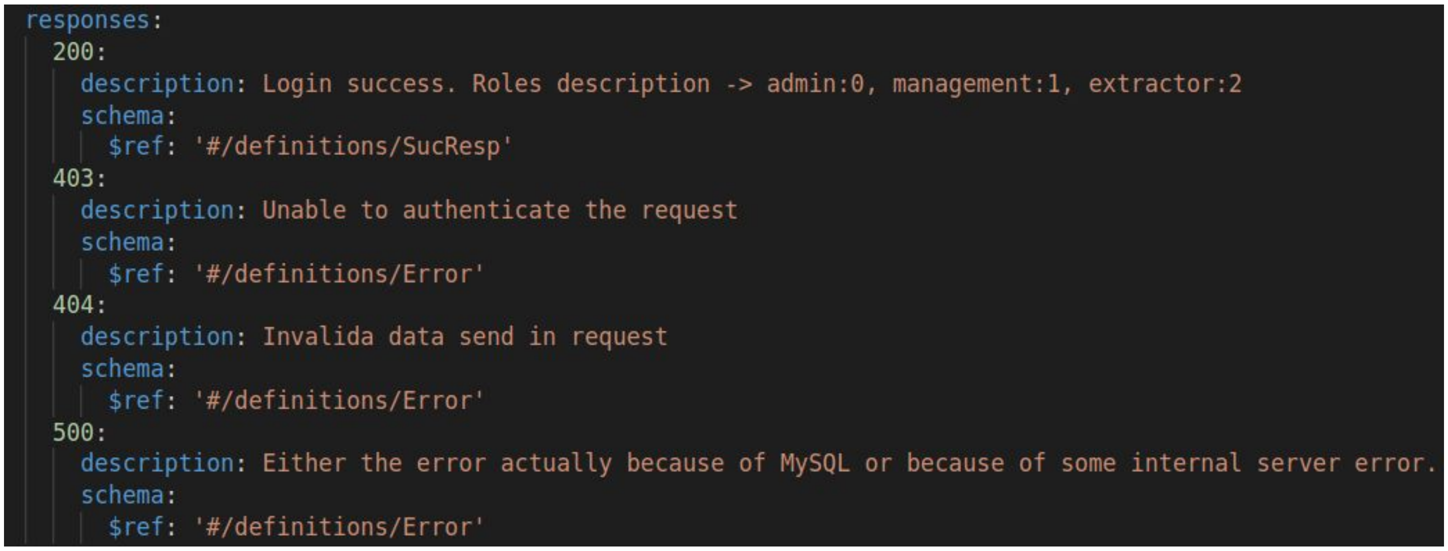
Parameters

Name	Description
User login data object *(body)*	Get user login session details through cookies Example Value | Model

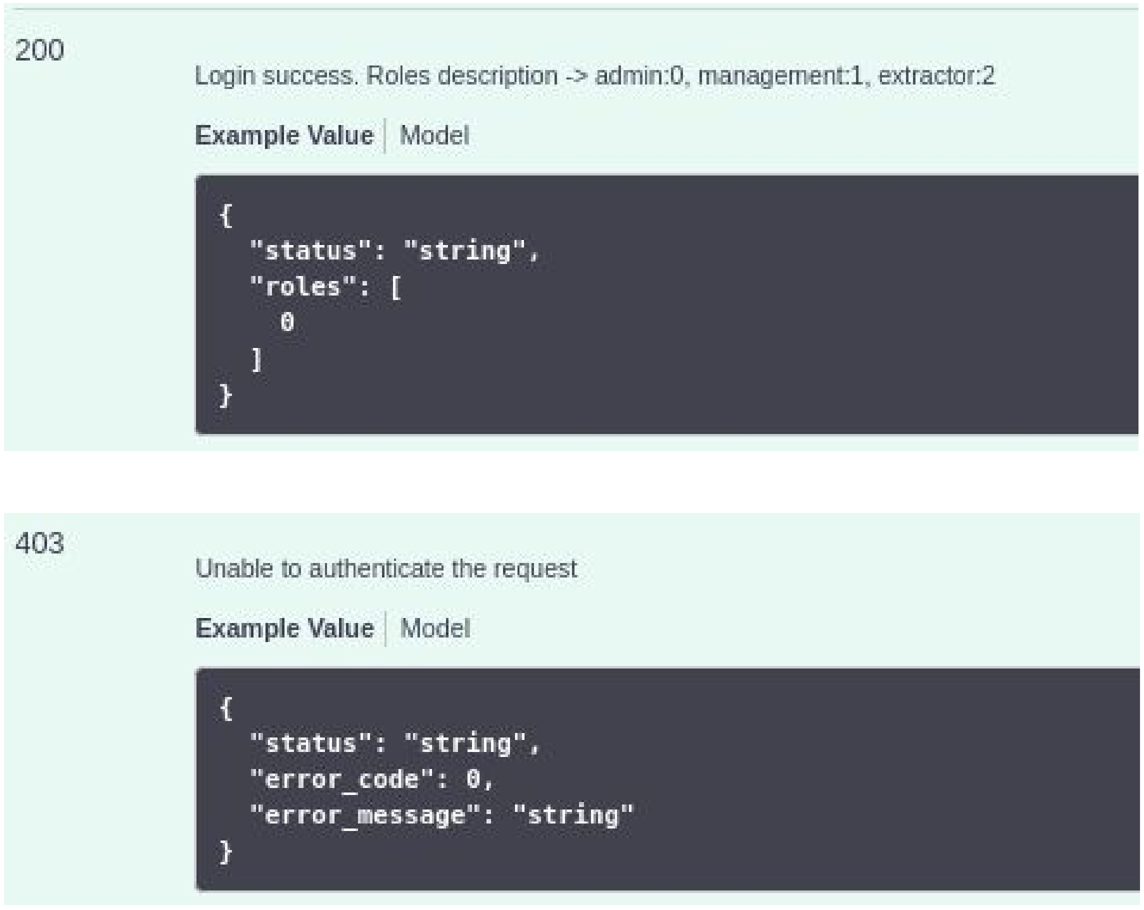
```
{
  "email": "string",
  "password": "string"
}
```

Parameter content type: application/json

<u>Fig. 12</u>- user-login post request API



```yaml
responses:
  200:
    description: Login success. Roles description -> admin:0, management:1, extractor:2
    schema:
      $ref: '#/definitions/SucResp'
  403:
    description: Unable to authenticate the request
    schema:
      $ref: '#/definitions/Error'
  404:
    description: Invalida data send in request
    schema:
      $ref: '#/definitions/Error'
  500:
    description: Either the error actually because of MySQL or because of some internal server error.
    schema:
      $ref: '#/definitions/Error'
```



200

Login success. Roles description -> admin:0, management:1, extractor:2

Example Value | Model

```
{
  "status": "string",
  "roles": [
    0
  ]
}
```

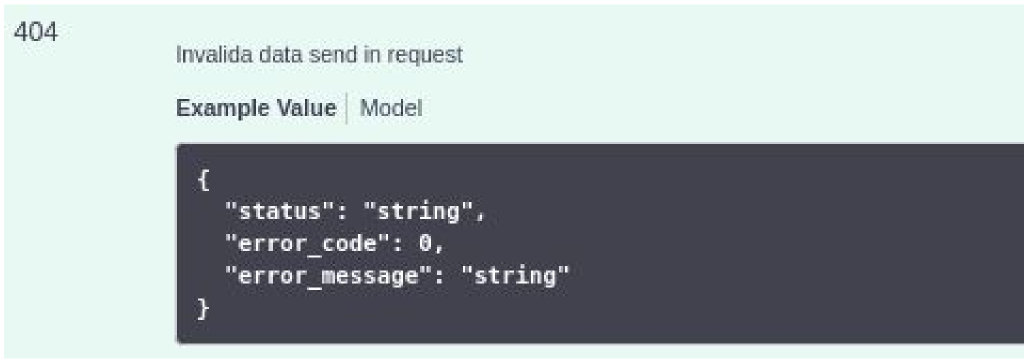
403

Unable to authenticate the request

Example Value | Model

```
{
  "status": "string",
  "error_code": 0,
  "error_message": "string"
}
```

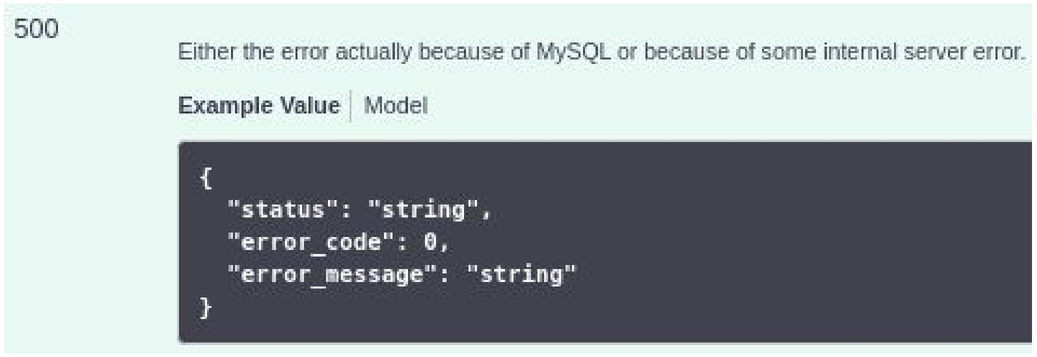
Fig. 13 **(a)**- user-login post request API responses


404

Invalida data send in request

Example Value | Model

```
{
  "status": "string",
  "error_code": 0,
  "error_message": "string"
}
```

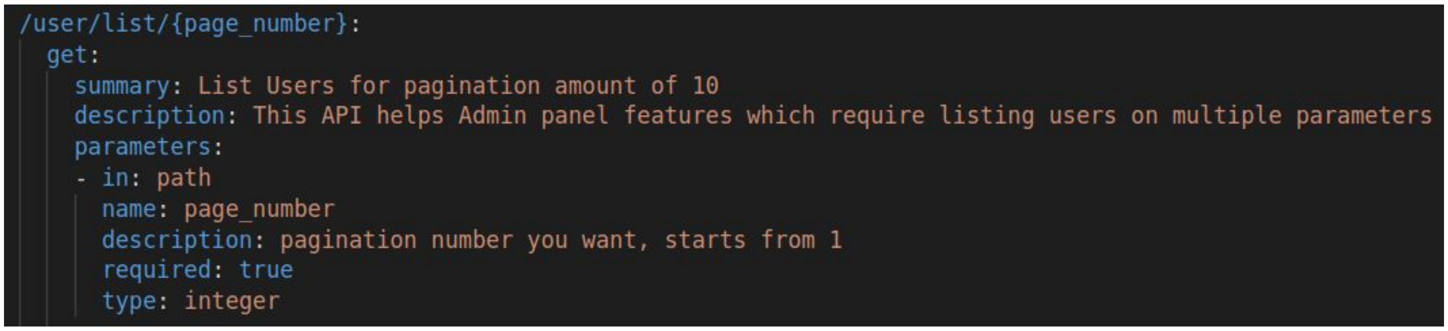

500

Either the error actually because of MySQL or because of some internal server error.

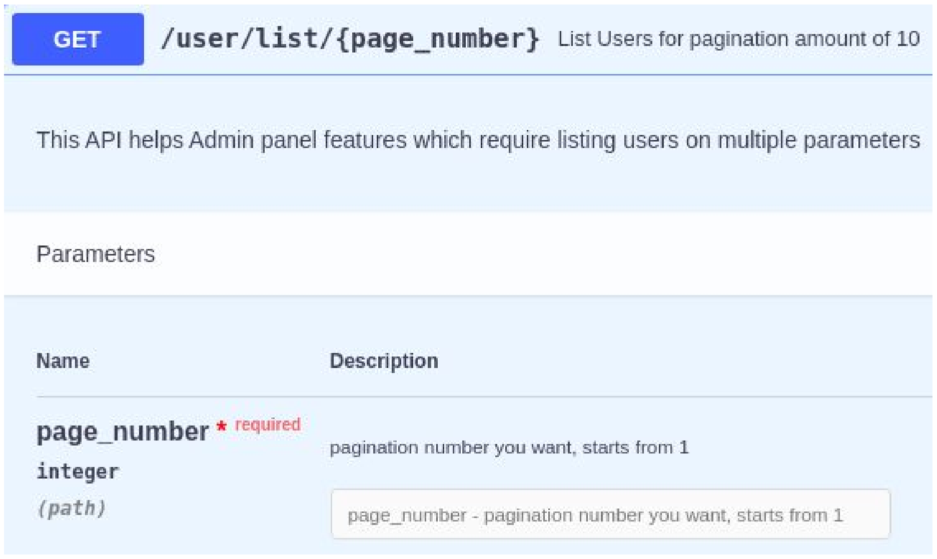
Example Value | Model

```
{
  "status": "string",
  "error_code": 0,
  "error_message": "string"
}
```

Fig. 13 **(b)**- user-login post request API responses



```
/user/list/{page_number}:
  get:
    summary: List Users for pagination amount of 10
    description: This API helps Admin panel features which require listing users on multiple parameters
    parameters:
    - in: path
      name: page_number
      description: pagination number you want, starts from 1
      required: true
      type: integer
```

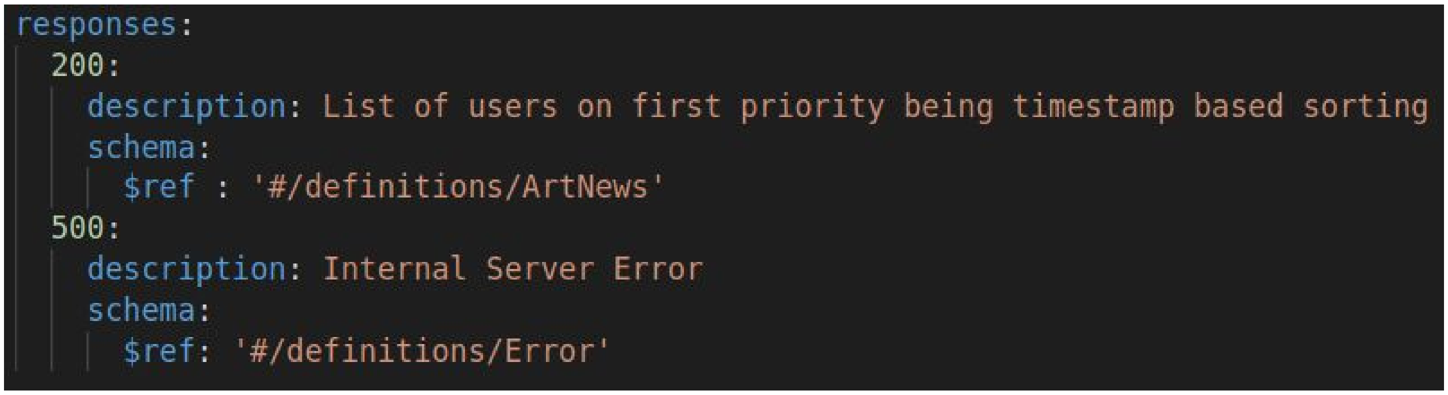

GET `/user/list/{page_number}` List Users for pagination amount of 10

This API helps Admin panel features which require listing users on multiple parameters

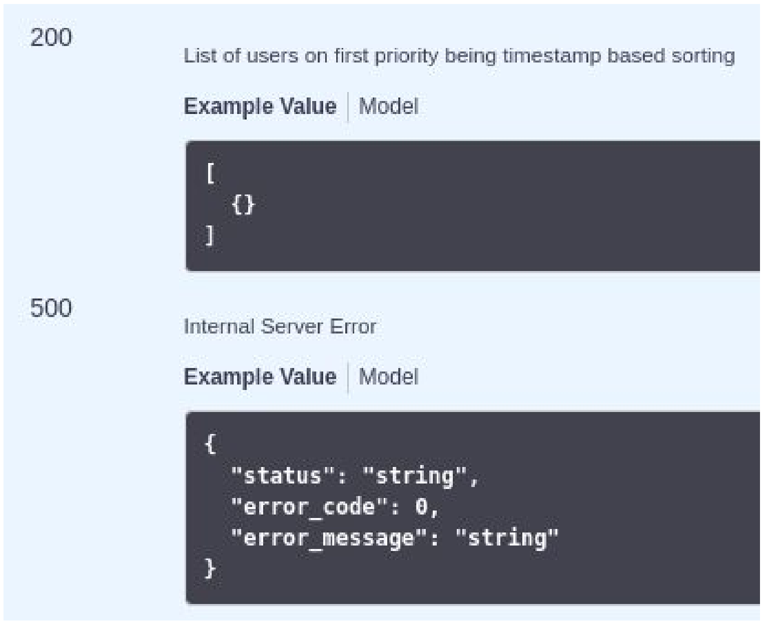
Parameters

Name	Description
page_number * required integer *(path)*	pagination number you want, starts from 1 page_number - pagination number you want, starts from 1

Fig. 14- user-list get request API

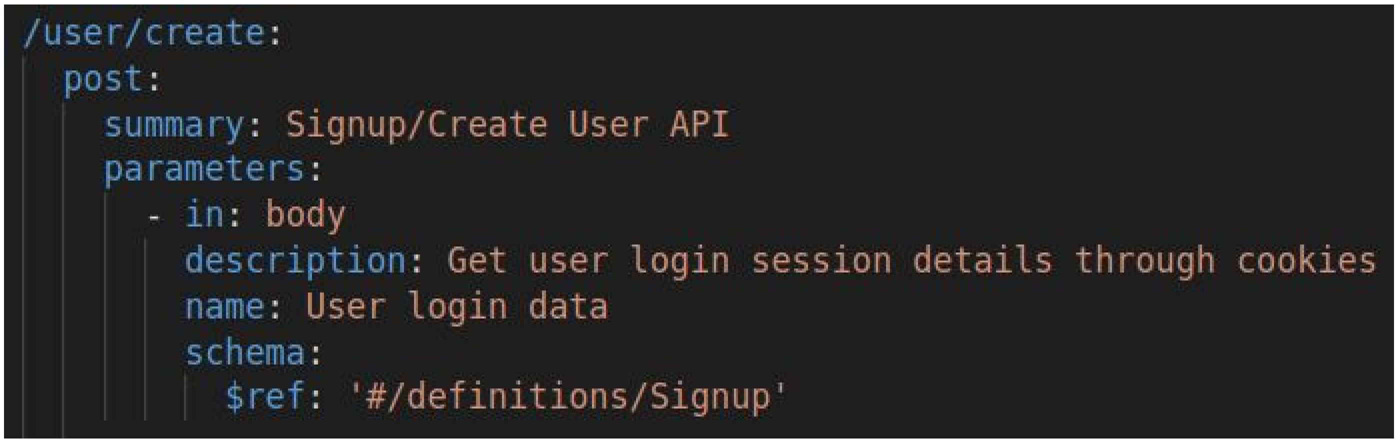


```
responses:
  200:
    description: List of users on first priority being timestamp based sorting
    schema:
      $ref : '#/definitions/ArtNews'
  500:
    description: Internal Server Error
    schema:
      $ref: '#/definitions/Error'
```

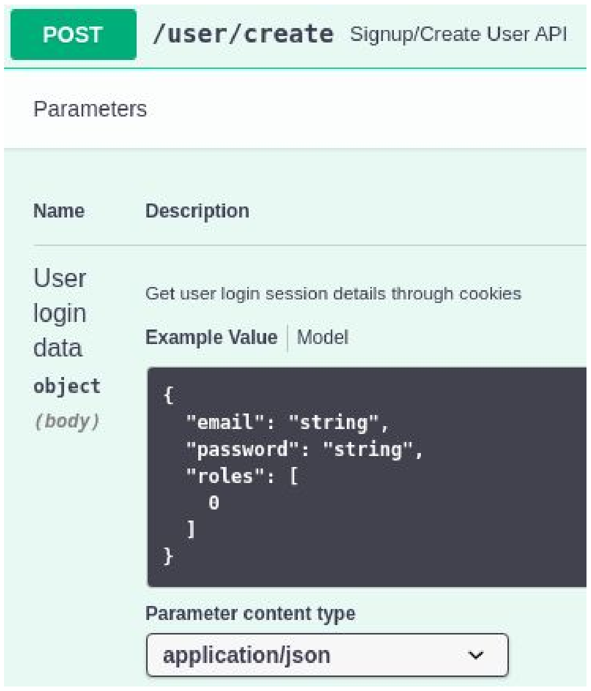


```
200
        List of users on first priority being timestamp based sorting
        Example Value | Model
        [
          {}
        ]
500
        Internal Server Error
        Example Value | Model
        {
          "status": "string",
          "error_code": 0,
          "error_message": "string"
        }
```

Fig. 15- user-list get request API responses



```
/user/create:
  post:
    summary: Signup/Create User API
    parameters:
      - in: body
        description: Get user login session details through cookies
        name: User login data
        schema:
          $ref: '#/definitions/Signup'
```


POST /user/create Signup/Create User API

Parameters

Name	Description
User login data object (body)	Get user login session details through cookies

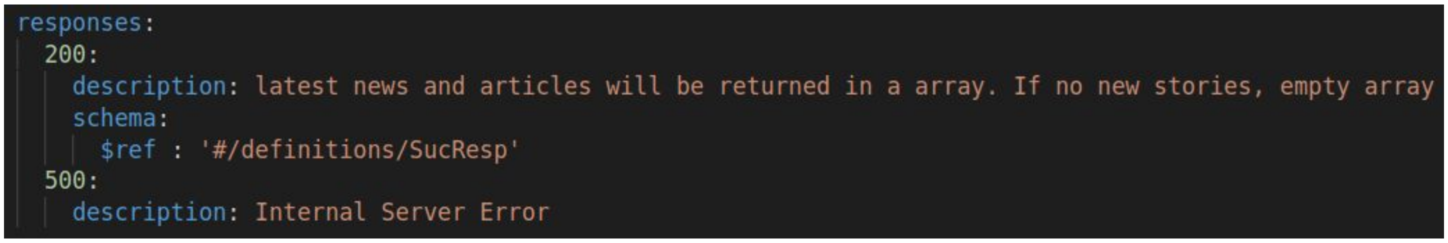
Example Value | Model

```
{
  "email": "string",
  "password": "string",
  "roles": [
    0
  ]
}
```

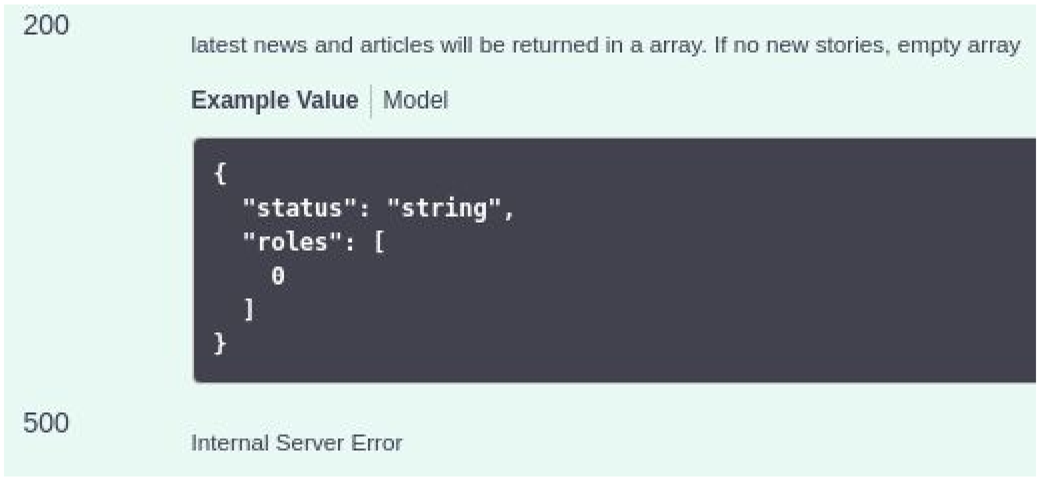
Parameter content type

application/json

<u>Fig. 16</u>- user-create post request API



```
responses:
  200:
    description: latest news and articles will be returned in a array. If no new stories, empty array
    schema:
      $ref : '#/definitions/SucResp'
  500:
    description: Internal Server Error
```



200

latest news and articles will be returned in a array. If no new stories, empty array

Example Value | Model

```
{
  "status": "string",
  "roles": [
    0
  ]
}
```

500

Internal Server Error

<u>Fig. 17</u>- user-list get request API responses

- **Roles management APIs -**
  1. **/user/role-update/add -** Useful for assigning roles to users in Admin Dashboard
     a. Method : POST
     b. Body : List of roles to add to a user along with user id or email
     c. Response : Success Message
  2. **/user/role-update/delete -** Useful for changing users roles in Admin Dashboard
     a. Method : POST
     b. Body : List of roles to delete from a user along with user id or email
     c. Response: Success Message

- **Data/Cases management APIs-**
  1. **/case/get/{case_id} -** Listing files in file explorer or getting details regarding a case
     a. Method: GET
     b. Param: case_id, the id of the case for which we want details
     c. Response: JSON Object with details like db path, list of dbs, other files etc
  2. **/case/create -** For creating a new dataset/case from extractor page
     a. Method: POST
     b. Body: Case name, extractor id, live device identifier, additional data extraction params
     c. Response: Success Response
  3. **/case/delete/{case_id} -** For deleting a case and it's data from server, useful in Admin Data Dashboard
     a. Method: DELETE
     b. Param: case_id, the id of the case you want to delete
     c. Response: Success Response
  4. **/case/list-files/{case_id} -** For showing files in File Explorer in Data Management
     a. Method : GET
     b. Param: case_id, for which we want to see all data files (.db, .json etc)
     c. Response: metadata of all files and links to explore APIs
  5. **/case/get-file/{case_id}/{file_path} -** For rendering file data into Explorer tab
     a. Method: GET
     b. Param: case_id, file_path: relative path from data directory
     c. Response: File source in suitable rendering format (has to be decided)
  6. **/case/list-cases/{page_number} -** Useful for listing all the cases in Admin panel and Data Management
     a. Method: GET
     b. Param: page_number, pagination of the data with 10 items in each page
     c. Response: Details of cases in a JSON Array
- **Analytics APIs -**
  1. **/analytics/query -** Useful for running queries on a set of data
     a. Method: POST
     b. Body: SQL Query for selected cases, (cases ids in the body as well)
     c. Response: Tabular data for written query or SQL errors

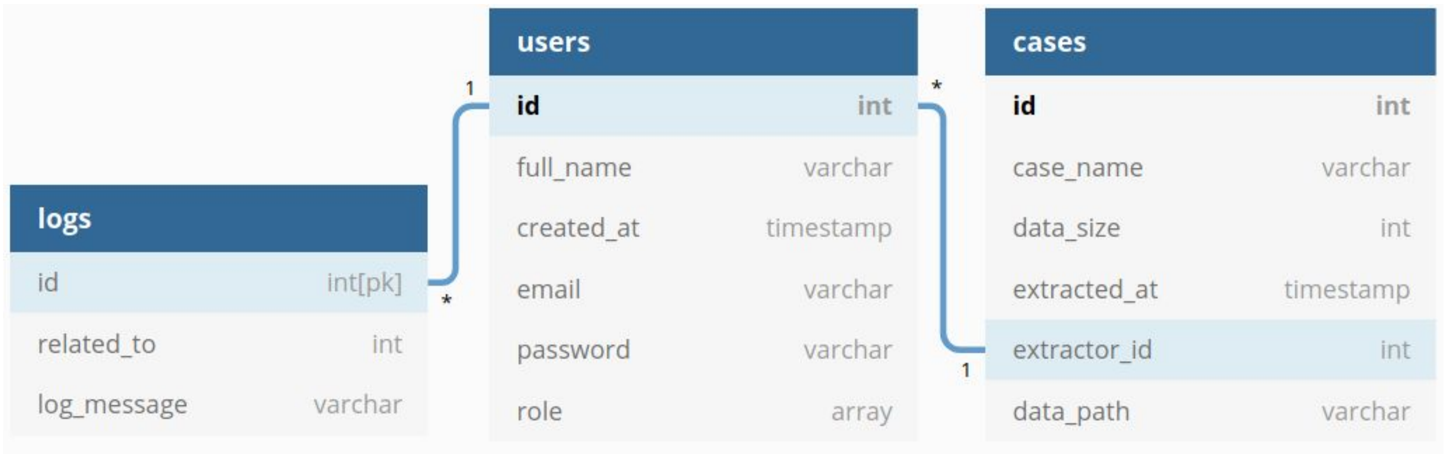
- **Extraction APIs -** Useful for data extraction from live devices
    1. **/extraction/list-devices -** List devices connected to machine on real-time
        a. Method: GET
        b. Response: Details of all devices connected to the machine along with their details
    2. **/extraction/extract-data -** For performing real-time data extraction from a device
        a. Method: POST
        b. Body: device id, case name, and additional metadata like tags etc
        c. Response: Live progress and status of extraction (failed, successful)

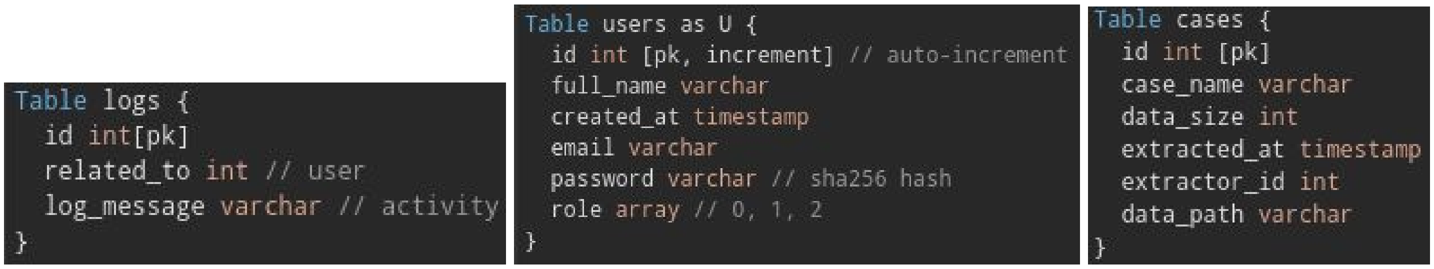
Phase 3 (31 July - 24 Aug):

- Design and implement a Database model, relation mapping (ORM) in the database to incorporate the above functionalities in MySQL using the SQLAlchemy toolkit.
- Compile complete project and fixes existing issues
- Run the project and identify new issues and fix them
- Write a project wiki and update the user and developer documentation.

In this phase, the final database model will be designed and will be integrated with the Flask application. The task in this phase will include configuring Flask - SQLAlchemy, initiating flask - SQLalchemy and Creating Database Models.
Will complete init scripts MySQL data models for new changes of phase 2, modularize Flask application codebase for better understanding.
In this phase, the final production level code will be developed. Cleaning of the codebase, inserting logger, writing swagger and postman docs will be done and will update user and developer documentation along with project wiki.

<u>Designs of Database model is shown below (Phase 3) :-</u>





```
Table logs {
  id int[pk]
  related_to int // user
  log_message varchar // activity
}
```

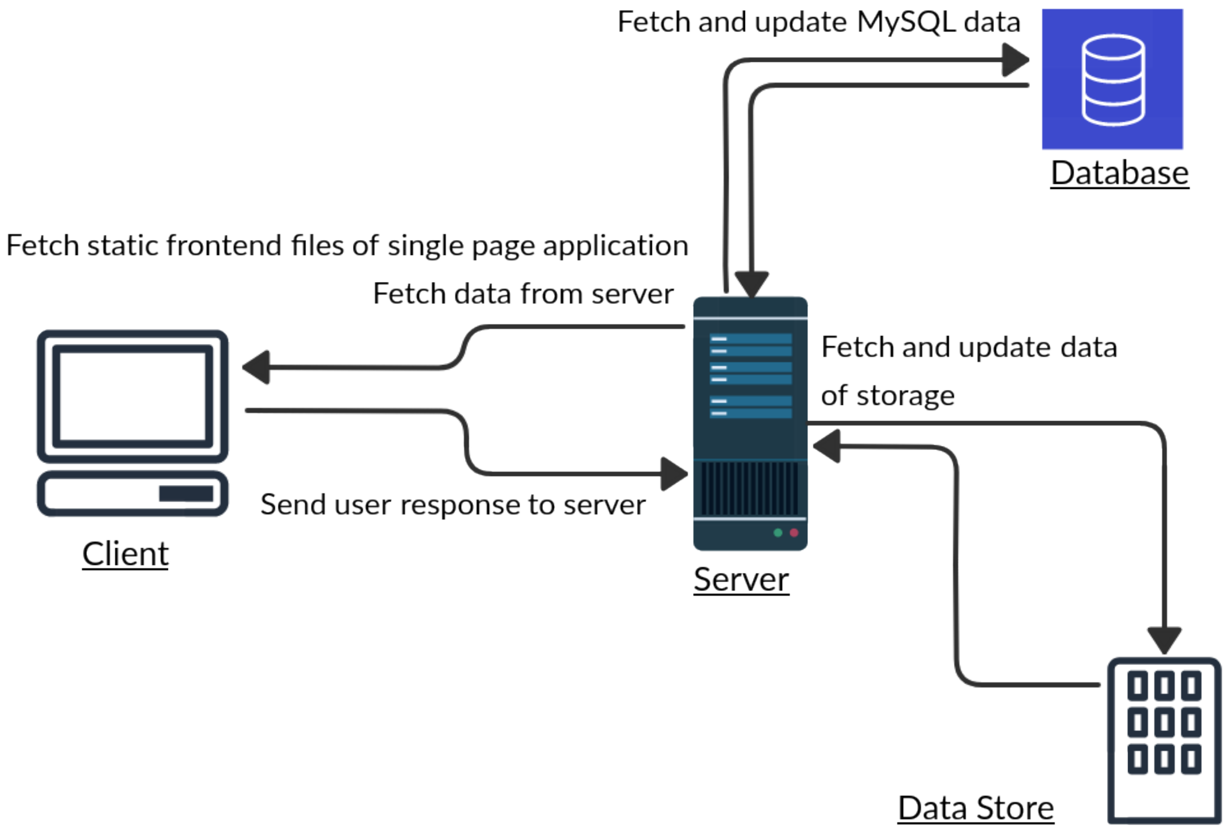
```
Table users as U {
  id int [pk, increment] // auto-increment
  full_name varchar
  created_at timestamp
  email varchar
  password varchar // sha256 hash
  role array // 0, 1, 2
}
```

```
Table cases {
  id int [pk]
  case_name varchar
  data_size int
  extracted_at timestamp
  extractor_id int
  data_path varchar
}
```

Fig. 18- Database design

Tech Stack:-

1. React, webpack, and Babel
2. Redux, React-Router, Redux-saga
3. Flask framework
4. MySQL
5. Python
6. Python-adb and adb_shell



Fetch and update MySQL data
Database
Fetch static frontend files of single page application
Fetch data from server
Fetch and update data of storage
Client
Send user response to server
Server
Data Store

Fig. 19- Data flow in the final design

Timeline:

Note: Each cell contains a time period of 1 week.

S.N.	Date	Work
1)	4 May - 1 June	Community Bonding (take feedback from mentors, Modify current code base)
		Coding officially begins!
2)	1 June - 7 June	Setting up tools for frontend development (React, Redux, ..) Creating a React boilerplate with additional dependencies.
3)	8 June - 14 June	Implementing Login and Admin panel designs in React, following Material UI
4)	15 June - 21 June	Implementing Manager and Extractor panel components in React
5)	22 June - 28 June	After the creation of components, components will be connected to each other with React Router.
		Phase 1 Evaluation (June 29 - July 3)
6)	29 June - 5 July	Taking guidance from the mentors and adopting new changes from feedback Start working on the backend side of the application
7)	6 July - 12 July	Writing down APIs details, metadata, in Flask with additional dependencies Setting up MySQL data models and init SQL scripts
8)	13 July- 19 July	Implementing APIs described above for user and data features
9)	20 July - 26 July	Implementing real-time devices APIs for extraction features Writing helpers for interacting with ADB using python-adb and adb_shell
		Phase 2 Evaluation (July 27- July 31)
10)	27 July - 2 Aug	Taking guidance from the mentors and adopting new changes from feedback
11)	3 Aug - 9 Aug	Leftover work from phase 1 and phase 2
12)	10 Aug - 16 Aug	Completing init scripts MySQL data models for new changes of phase 2 Modularize Flask application codebase for better understanding
13)	17 Aug - 23 Aug	Cleaning codebase, inserting logger, writing swagger and postman docs
		Final Evaluation week starts (Aug 24- Aug 31)
14)	24 Aug - 31 Aug	Writing wiki, user and development documentation
		Final results of Google Summer of Code 2020 announced (September 8)

SCoRe Contributions:

Pull Requests:-

- **[Merged]** Added support for JSON
  https://github.com/scorelab/OpenMF/pull/34
- **[Merged]** Moved additional files to ExtraResources
  https://github.com/scorelab/OpenMF/pull/26
- **[Merged]** Fix redundancy bug.
  https://github.com/scorelab/OpenMF/pull/21
- **[Merged]** Modified converter.py to solve Issue No. 18
  https://github.com/scorelab/OpenMF/pull/19
- **[Merged]** Added readme.md file
  https://github.com/scorelab/OpenMF/pull/33
- **[Merged]** Added favicon for OpenMF
  https://github.com/scorelab/OpenMF/pull/32
- **[Merged]** Added logo for OpenMF
  https://github.com/scorelab/OpenMF/pull/31
- **[Open]** Added developer and user guide in Readme
  https://github.com/scorelab/OpenMF/pull/35
- **[Open]** Added essential files for flask
  https://github.com/scorelab/OpenMF/pull/36
- **[Closed]** Designed and added logo for OpenMF
  https://github.com/scorelab/OpenMF/pull/30
- **[Closed]** Updated readme.md
  https://github.com/scorelab/fact-Bounty/pull/545
- **[Closed]** Added OAuth setup process in detail
  https://github.com/scorelab/fact-Bounty/pull/546
- **[Open]** Fixes localhost:9200 for elastic search server is not working issue
  https://github.com/scorelab/fact-Bounty/pull/536
- **[Open]** Modified OAuth setup guide for easy setup of OAuth service
  https://github.com/scorelab/fact-Bounty/pull/547

Issues Authored:-

- **[Closed]** Update Developer guide
  https://github.com/leopardslab/dunner/issues/203
- **Closed]** converter.py does not store the extracted data in right directory
  https://github.com/scorelab/OpenMF/issues/18
- **[Closed]** Redundant folder creation every time we run collector.py
  https://github.com/scorelab/OpenMF/issues/20

- **[Closed]** Add support for JSON. https://github.com/scorelab/OpenMF/issues/22
- **[Closed]** Add logo for OpenMF https://github.com/scorelab/OpenMF/issues/28
- **[Closed]** Add Favicon to repository https://github.com/scorelab/OpenMF/issues/29
- **[Open]** collector.py fails to store data in the correct directory. https://github.com/scorelab/OpenMF/issues/23
- **[Open]** Add readme.md https://github.com/scorelab/OpenMF/issues/27
- **[Open]** No destination folder for extracted raw data https://github.com/scorelab/OpenMF/issues/24
- **[Open]** Provide proper structure to project https://github.com/scorelab/OpenMF/issues/25
- **[Open]** localhost:9200 for elastic search server is not working https://github.com/scorelab/fact-Bounty/issues/535
- **[Open]** Domain for fact-Bounty has been expired https://github.com/scorelab/fact-Bounty/issues/537
- **[Open]** Vulnerabilities found while setting up fact-bounty-client https://github.com/scorelab/fact-Bounty/issues/538
- **[Open]** Add detailed OAuth Setup instructions for easy installation https://github.com/scorelab/fact-Bounty/issues/544

Personal Information:

- **Name:** Shivanshu Raj Shrivastava
- **Email:** shivanshu1333@gmail.com
- **GitHub:** shivanshu1333
- **LinkedIn:** Profile Link
- **SkypeID:** shivanshu1333
- **Gitter nickname:** shivanshu1333
- **First language:** Hindi, proficient in English
- **Time zone:** Indian Standard Time (GMT +5:30)
- **Contact:** +91 9425646127
- **University:** Indian Institute of Technology, Roorkee
- **Year of study:** 4th year B. Tech.
- **Major:** Electronics and Communication Engineering (Batch of 2020)

Few sentences about the student and why he/she thinks as the best person to do this project.

My contributions to open source have helped me gain experience in understanding the flow of any pre-written code at a rapid pace and enabled me to add/update features. My previous involvement in

OpenMF includes various patches which are mentioned above, solving these issues and creating pull requests gave me a clear cut idea about the project and I'm confident to complete all the tasks mentioned by me. I have sound knowledge of **Python**, **React, Flask, RESTful APIs** and **databases** which are the main technical requirements of **OpenMF.**
Also, I have already completed some projects which require the skills required to complete this project. Further, I don't have any planned commitments during these summers, so I can devote my maximum time to this project.

Reference:

[1] scorelab/OpenMF
[2] scorelab/androphsy

Questions:

1). Are you a SCoRe contributor/ Have you contributed to SCoRe before?
Ans: Yes, I have contributed to the SCoRE organization since January 2020. Yes, My contribution is for **OpenMF, Dunner** and **Fact-Bounty** Projects. At the initial stage, I tried to understand what the project was all about and the architectural structure of the project.
Since this project is implemented with most of the technologies that I'm already familiar with, I started solving open issues so that I could get familiar with the codebase. In solving issues, I found some new issues which I discussed with the community and raised them as a ticket and successfully landed patches for some of them.

2). How can we reach you (eg: email) if we have questions about your application?
Ans: **Email :** shivanshu1333@gmail.com
Gitter : shivanshu1333
Phone : +91 9425646127
+91 8006470279

3). What is your GitHub username(s):
Ans: shivanshu1333

Project Specific Questions:

4). **Which SCoRe GSoC project are you applying for (please submit separate applications for each project):**
Ans: OpenMF - Create web client for OpenMF

5). What do you plan to accomplish over this summer for this project?
Ans:

- Cleanup the current code base of OpenMF and make a robust Command line tool of OpenMF.
- Design and Implement a Client-side Framework for Web Client.

- Design and Implement Login, Admin, Management and Extractor pages (refer Fig. 3-4).
- Implementation of backend from scratch and writing different APIs in flask framework
- Implementation of different roles (Admin, Management, Extractor) from scratch.
- Extending the functionality of OpenMF to include download, upload, and management of files on the server.
- Implementation Access Control Lists (ACL) to manage file permissions among users by Admin.
- Design and implement a Database model, relation mapping (ORM) in the database to incorporate the above functionalities in MySQL using the SQLAlchemy toolkit.
- Write a project wiki and update the user and developer documentation.
- Weekly blogs on medium.

6) If you have your own project to propose, please describe it here:
Ans: NA

7). Projects related details. (Have you tried that project you selected from SCoRe project list? What problems, if any, were presented? What prevented you from getting the entire system up and running?)
Ans: I am an active contributor in OpenMF so I'm already set up. While setting up my system environment, I faced some issues with the project setup as there was no readme. After understanding the code I came to know about the project. Last year's GSoC repository by scorelab helped me a lot to understand the implementation of the project. Initially, I was working with an emulator and then I rooted my phone, which I did for the first time in my life. Also, I discussed it with the community and completely understood the working of the project.

8). List down any plans you have during this summer
Ans: I have no secondary commitments during summer breaks, so I can dedicate my entire day (10-11 hours) to this project. I have my end term exams from 24 April to 4 May so in that time period I will be inactive.

9). Education:
Ans:
- **University :** Indian Institute of Technology, Roorkee
- **Field of study :** Electronics and Communication Engineering (Batch of 2020)
- **Current year of study :** 4th year B. Tech.
- **Programming Courses :**
    - Object-Oriented Programming in Java
    - Computer Architecture
    - Operating Systems
    - Data Structures and Algorithms
    - Embedded System Design
    - Introduction to Machine Learning by Prof. Andrew Ng (Coursera)
    - Introduction to Machine Learning (NPTEL)

- **Group Projects :**
    - Real time, GIS based tracking for water Transportation in India :
        - This project was done under a Hackathon organized by the Indian government.
        - An android app was developed to track the level of water in water tankers using various sensors and Arduino as a microcontroller (see link for more details).
    - Smart-Spell :
        - OCR to read text from books from an Android App.
    - ToDo App :
        - A basic ToDo application built as a beginner level project to learn the basics of web development.
        - Used Flask, Vertabelo, SQLAlchemy and Bootstrap for development.
        - Used Heroku for deployment of the completed project.
    - C.R.U.D.M.E (Create.Read.Update.Delete for Managing Employees) :
        - A simple CRUD employee management web application.
        - Flask was used as the micro web framework.
        - MySQL was used as the relational database to make tables.
        - Learned to make complex models and one-to-many/many-to-many database structures.
        - Learn about ORM and SQL.
    - Humanoid Bot :
        - Worked with my college robotics team (marsiitr) to make a lower body of a Humanoid.
    - Interactive-Led-Display :
        - Developed a Trex game to play over a Led Grid using Arduino as microcontroller.
    - **Technology for Soldier support, Defence Research, and Development Organization(DRDO):**
        - Designed a localization system for real-time position estimation of soldiers with Gesture communication facility,
        - Developed hardware for VHF/UHF transmission of encrypted video from the battlefield to basecamp.
        - Integrated the Health Monitoring, Localization subsystems with the Raspberry Pi based latch on the device on each soldier.

10). Do you have work experience in programming? Tell us about it.
Ans: I have completed many programming-based projects, as mentioned above. Last year I had an internship in Samsung R&D Institute India, in which I established a Node server which takes care of scheduling lock of Samsung smart lock codes of different users and communicates to Samsung SmartThings cloud at the right time to issue commands like set, reset, update and delete lock codes to respective locks from Zigbee hub. I have also completed many institute and course projects, some of which can be found in my Github.

11). Do you have previous open source experience? Briefly describe what you have done.
Ans: I had a summer software development internship with Samsung R&D Institute, in which I worked on Samsung SmartThings cloud which is an open-source community. I am also a member of Fossasia and have made some contributions in the past, I've also contributed to the GFOSS organization in which I worked on the web client of Clio. I am proficient with Flask, NodeJs, frontend frameworks like Angular and React, deploying containers like Docker and Kubernetes, and programming languages like C++, Python, and Javascript.
I am not engaged in any other activity this summer and I am looking forward to wholeheartedly taking this opportunity to build a bigger and better version of OpenMF.

12) Tell one interesting fact about yourself.
Ans: I am an athlete and very concerned about my fitness. I love to keep a balance in my work and social life, and I think that's the key to living a beautiful and happy life.
