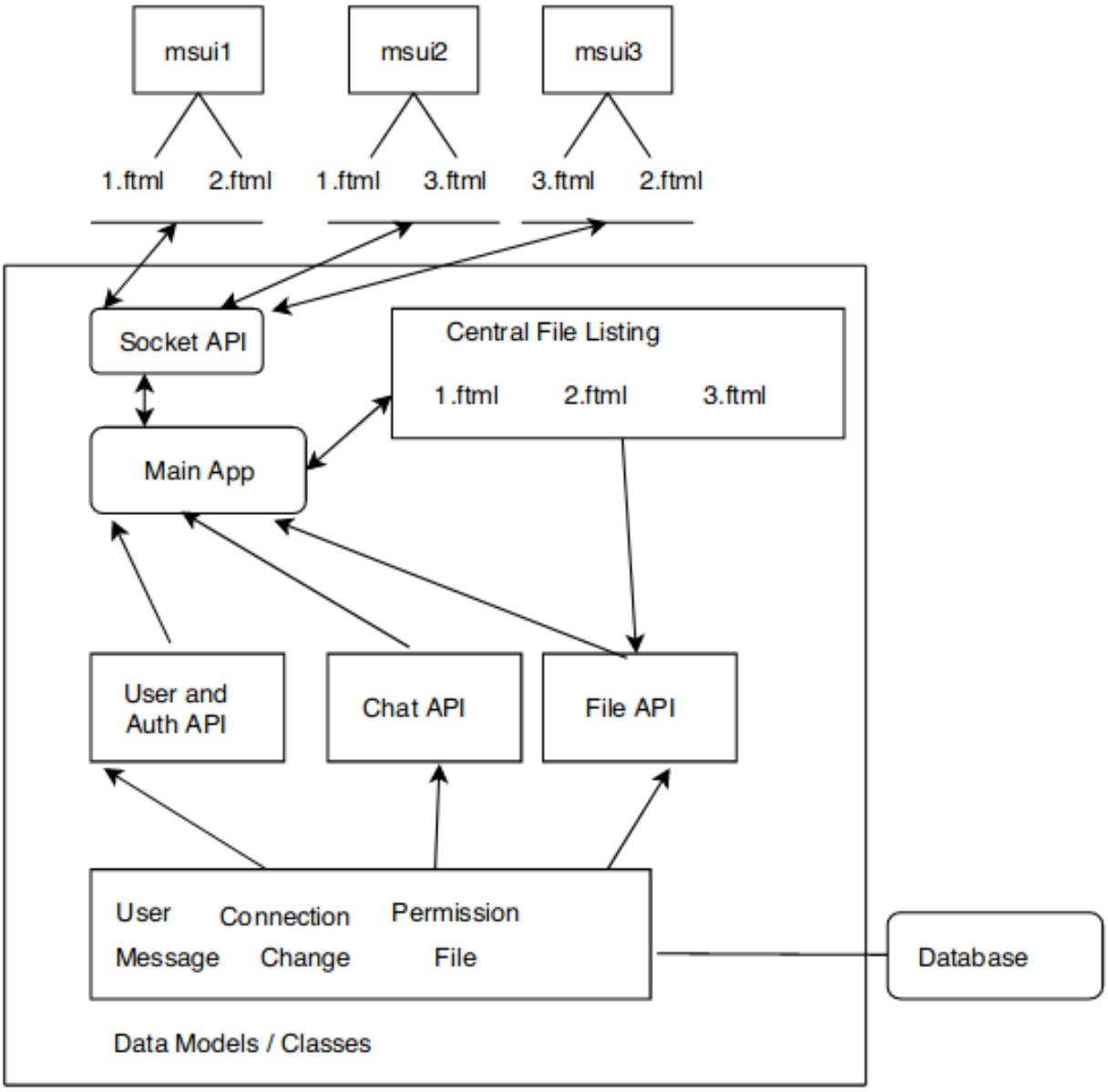
Developer documentation - Mscolab(development version)

Note: Connection class and database model was scrapped because it wasn't necessary to keep track of connections.

Dataflow Model



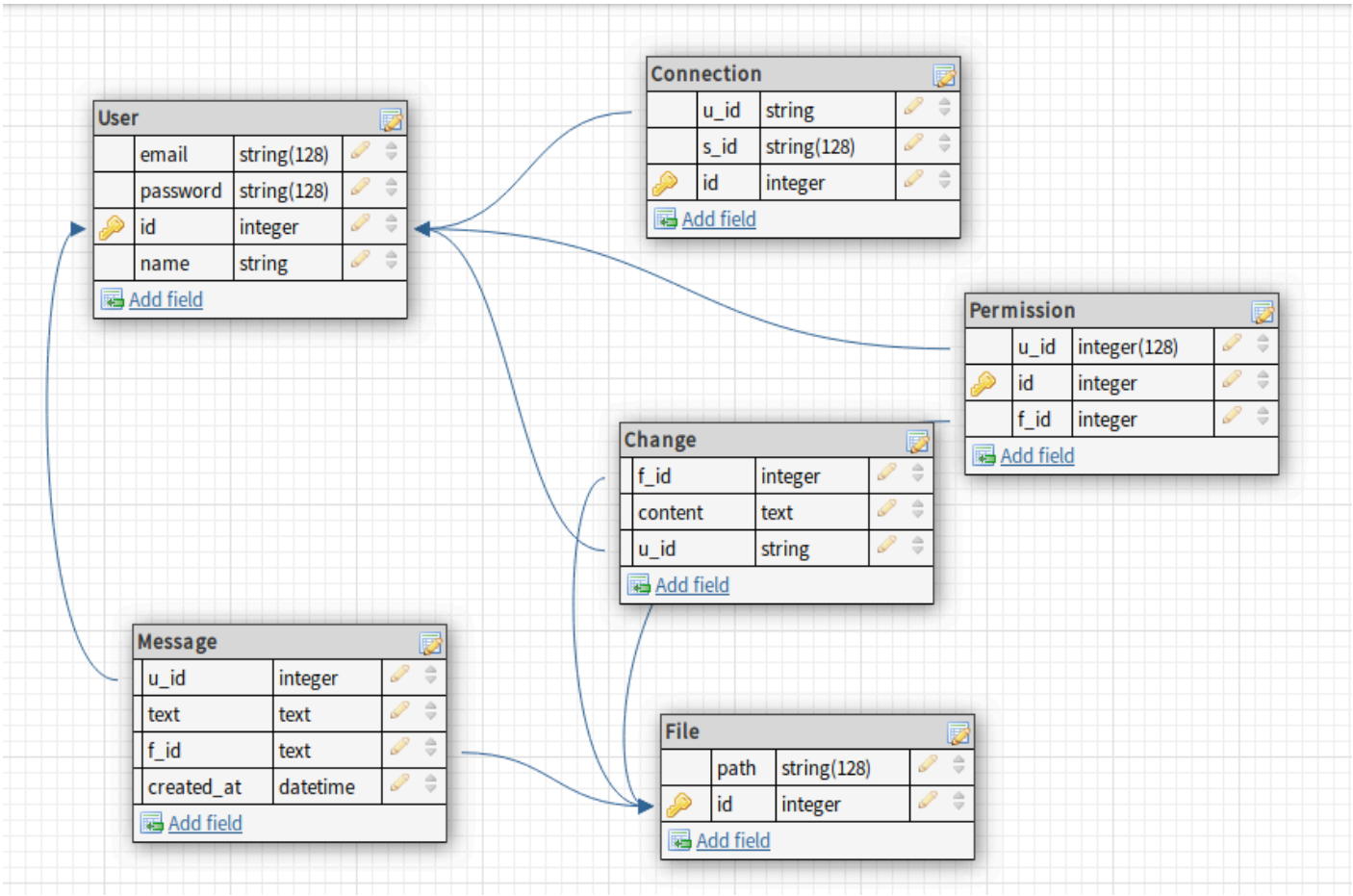
`mscolab` has four major components:

- Data Models / Classes
- Main App

- APIs
- UI

Among these, the first three modules would be developed within `mscolab` server. UI of this software would be integrated with msui module of mss software.

Data Models



The project needs six basic models named:

- User
  This class represents an user of `mscolab`.
  id: user-id
  email: email-id of the user
  name: Screen name of the user
  password: bcrypt hashed password

- Permission
  This data represents the authorization of an user to collaborate on a file.
  u_id: user-id
  f_id: file-id
  access_level: enum[“admin”, “collaborator”, “viewer”]

- Message
  Message represents a single message unit, linked with a chat(further linked to a file).
  u_id: user-id
  f_id: file-id
  text: message content
  created_at: used to order the messages in a conversation

- Change
  Change represents a change in .ftml file submitted by an authorized user.
  u_id: user-id
  f_id: file-id
  content: 'diff' of file made by the user
  created_at: used to sort the changes w.r.t time

- File
  File represents an .ftml file.
  id: to be used as fileId
  path: filepath - unique
  description: description of the file, stating its purpose of creation.

APIs

Some of the major APIs and their functionalities are listed as follows:

- Socket API
  - connect_user
    Checks email-id and password, if authorized saves user-id and socket-id to Connection table.
  - disconnect_user
    Removes connection associated to the socket
  - message
    Notifies main app about incoming message from one of connected socket clients.
  - emit
    Emits 'file change' or 'message' events to other users to change their local data in real-time, sent by main-app.
  - is_online
    Used to check if an user is online by looking for an entry in connection
- User and Authentication API
  - add_user
    Add email-id, name, and password to 'User'
  - remove_user
    Remove user from 'User' table.
  - change_password
    Change password of user

- authenticate
  Checks if an email-id and password passed as parameters are valid, and matching with one in the database.
- user_exists
  Checks if an email-id exists in 'User'

- File API
  File API will be based on fs library, and the storage options can be modified as per need. A configuration file would be used to control the same.
  - file_save
    Adds an entry in the 'Change' table.
    '$name.ftml' is tweaked to append 'Change-Id' to change-log attribute inside each <Waypoint/> tag. This would help to display information specific to this waypoint in UI.
    Saves the new file atomically, (this process will be made efficient by saving only the 'diff' at the right file cursor)
  - get_file
    Returns file as ASCII string or buffer, as instructed in the parameter.
  - get_authorized_users
    Returns a list of users who are authorized to collaborate on file identified by f_id/f_name.
  - get_change_log
    Returns a list of changes by collaborating users sorted by timestamps from Change table with f_id as key.
  - exists
    Returns boolean value if the file with 'file-name' exists or not.
  - list
    Returns an array of 'File' data with permission level for each file. This data can be used to view projects dashboard in client's side, as a list of projects the user is admin of and another list of projects the user is collaborating on.
  - delete_file (access_level = admin)
    Deletes file from file-path and 'File' table, preferentially delete entries related to this file from 'Message', 'Permission', 'Change'.
  - add_permission (access_level = admin)
    Checks if user with user-id exists. If yes, add user-id and file-id to 'Permission' table.
  - remove_permission (access_level = admin)
    Checks if user with user-id exists. If yes, remove entries of 'user-id and file-id together' in 'Permission' table.
  - rename_file (access_level = admin)
    Rename a file, would basically change 'path' in 'File' table corresponding to file-id.
- Chat API
  - message_save
    Adds an entry in the 'Message' table.

- get_messages
  Return an array of messages corresponding to a file_id (chat_id as each file can have one ‘Chat’ entry), sorted by ‘created_at’ values.

Main App

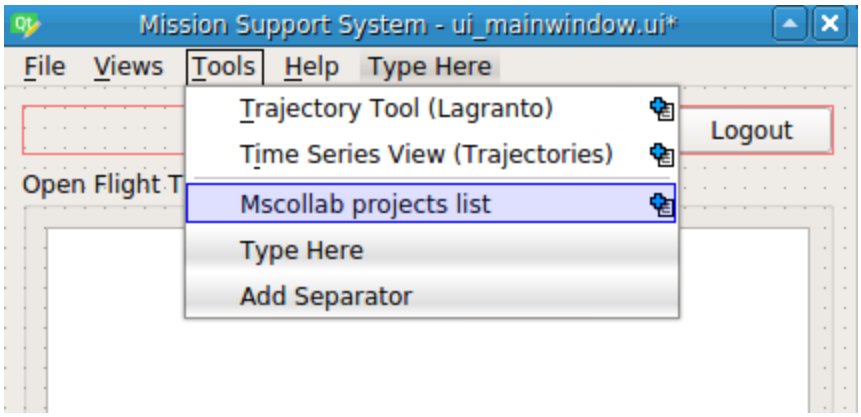
This module orchestrates all other services and APIs and regulates the data-flow.
A pseudo-code of main app is as follows:

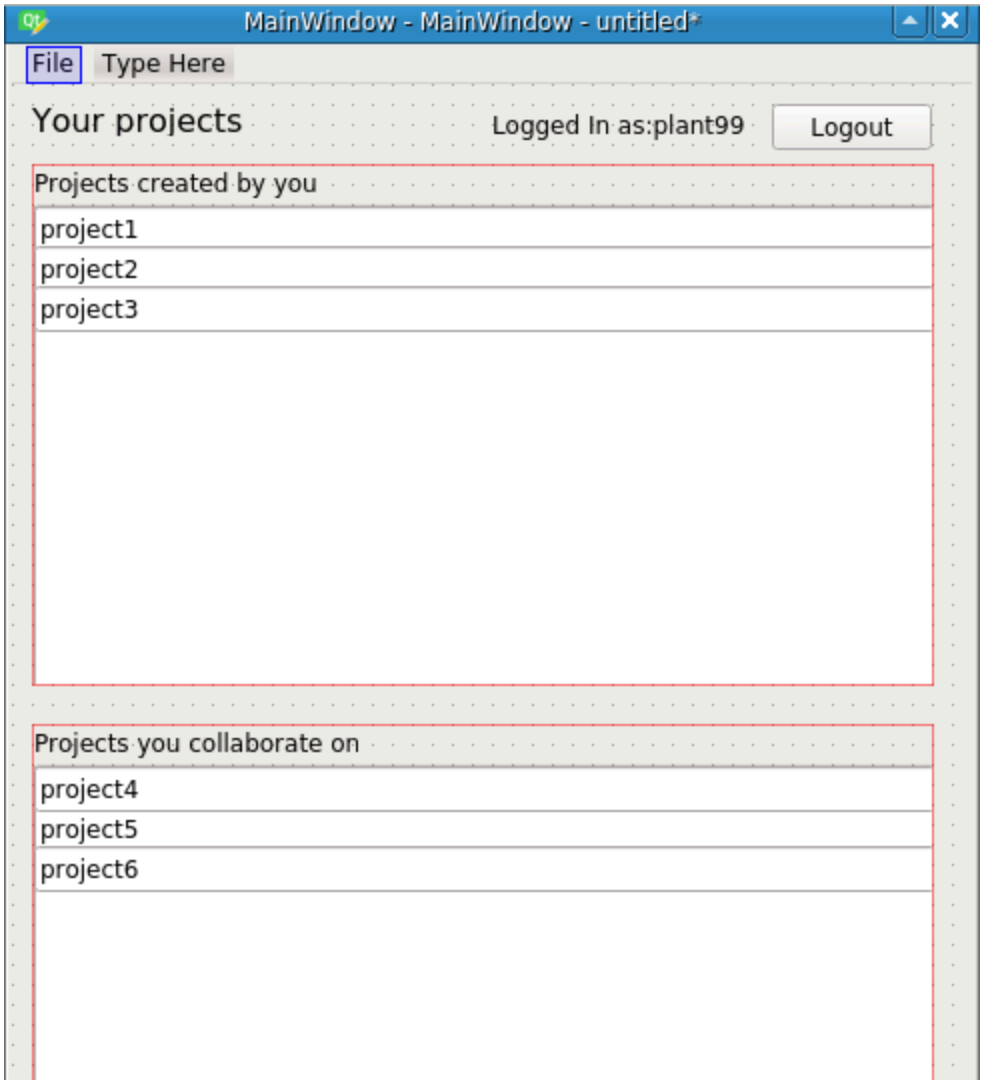
```
import socketManager
import fileManager
import authManager
import chatManager
import Flask
app = Flask(__name__)

@socketManager.sio.on('connect')
def connect(sid, env):
       # check auth here by authmanager
      socketManager.connect_user()


@socketManager.sio.on('disconnect')
def connect(sid, env):
      socketManager.disconnect_user()

#  use decorator to check auth
@sio.on('message')
def message(sid, data):
      if data.type == "file":
            # fileManager handles file here
             pass
      else:
             # data.type = message
            # chatManager handles messages here
            pass

app.route('/user')
def user_handler():
      args = request.args
      # authManager handles the rest
      # would be used to add remove user, etc
```

UI

UI module of `mscolab` will be integrated with msui, the core UI module of mss software. The temporary files would be stored in ‘~/mssdata’ directory or as configured in ‘mss_wms_settings.py’.

A list of projects which the user is working on can be displayed as illustrated below, which can be activate by clicking on Tools->mscolab projects list, on MSS' main window.



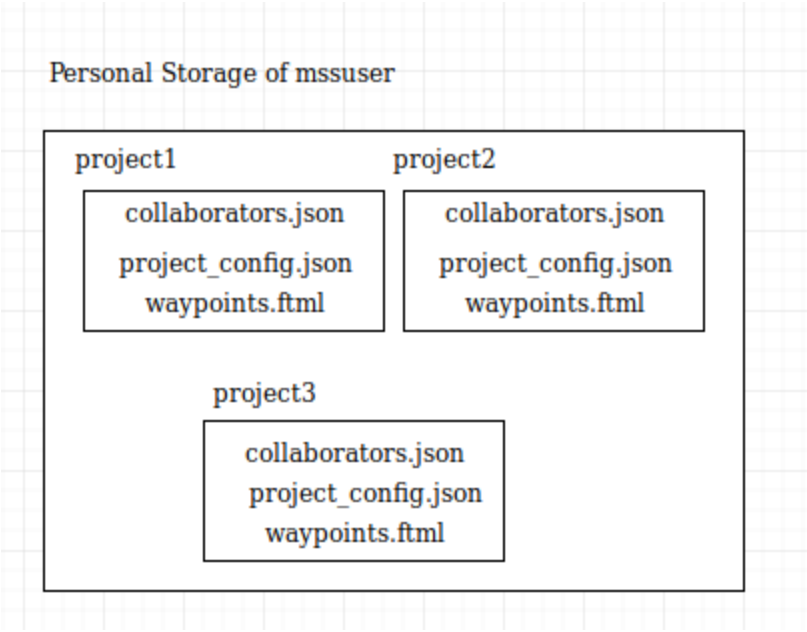


Double clicking on a project opens the mscolab-ui window as shown on page 11.

Project

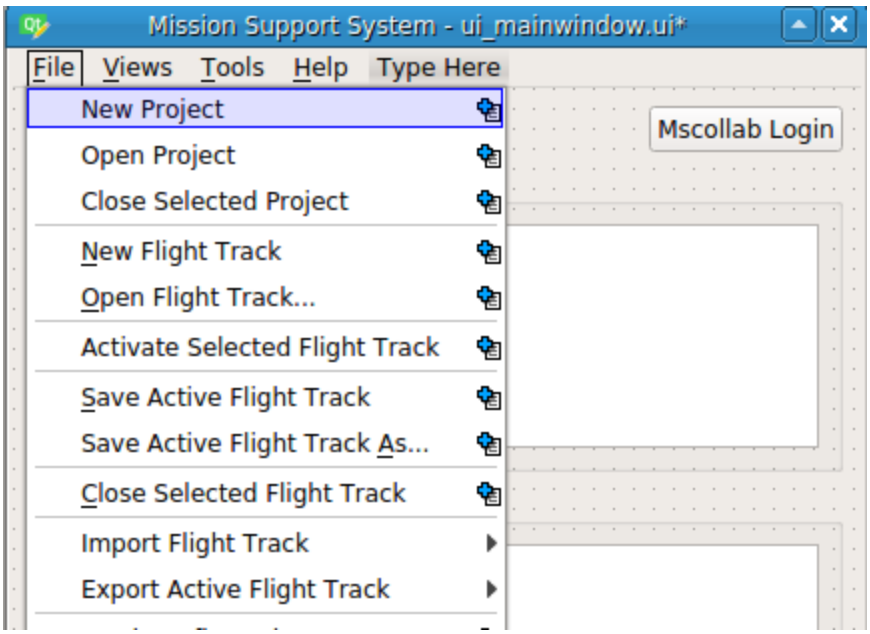
- A 'project' is a data storage model implemented by popular code IDEs, like vscode,pycharm etc.

- In this case, instead of treating a '$filename.ftml' as a project, since it won't be aesthetic to store configuration data in ftml file, it would be better if we introduce a project as a collection of some files.
- To start with, it will have a flightpath related file, and a configuration file, and a contributors file showing waypoint details and collaborators who contributed to change of the waypoint.
- An illustration is shown below.

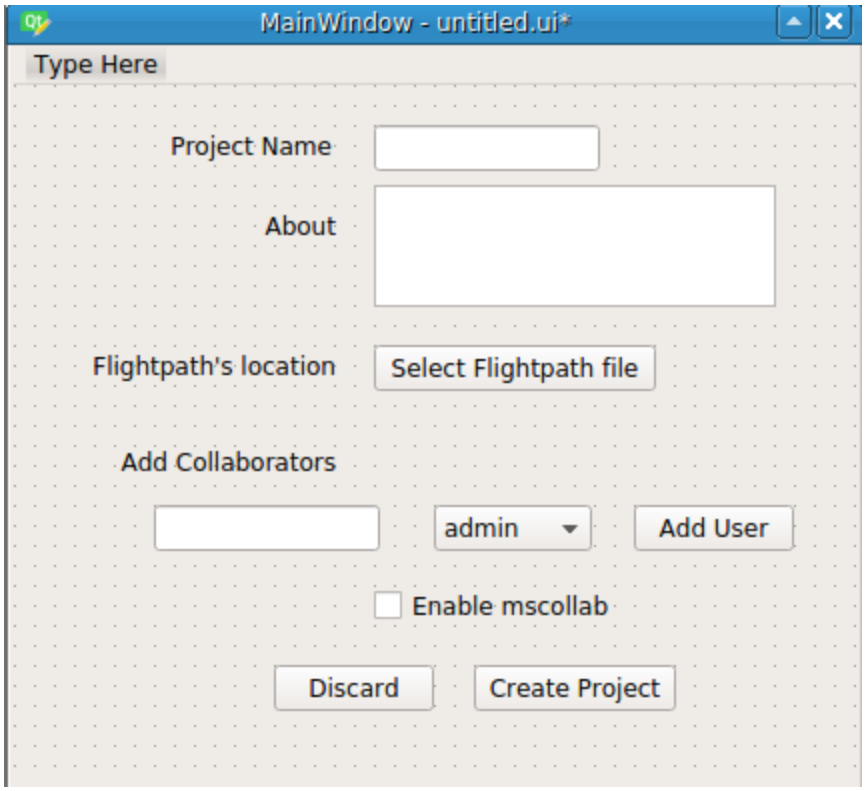


- To integrate this with mss, create_new_view function can be modified to open a flightpath in a particular view mode. The attribute `self.active_flight_track` can be changed to `self.active_project` . And each window opened would have a flightpath with some configuration obtained from project_config.py, and contribution details of each waypoint from project_config.py.

Projects can also be opened/created directly from main-window.



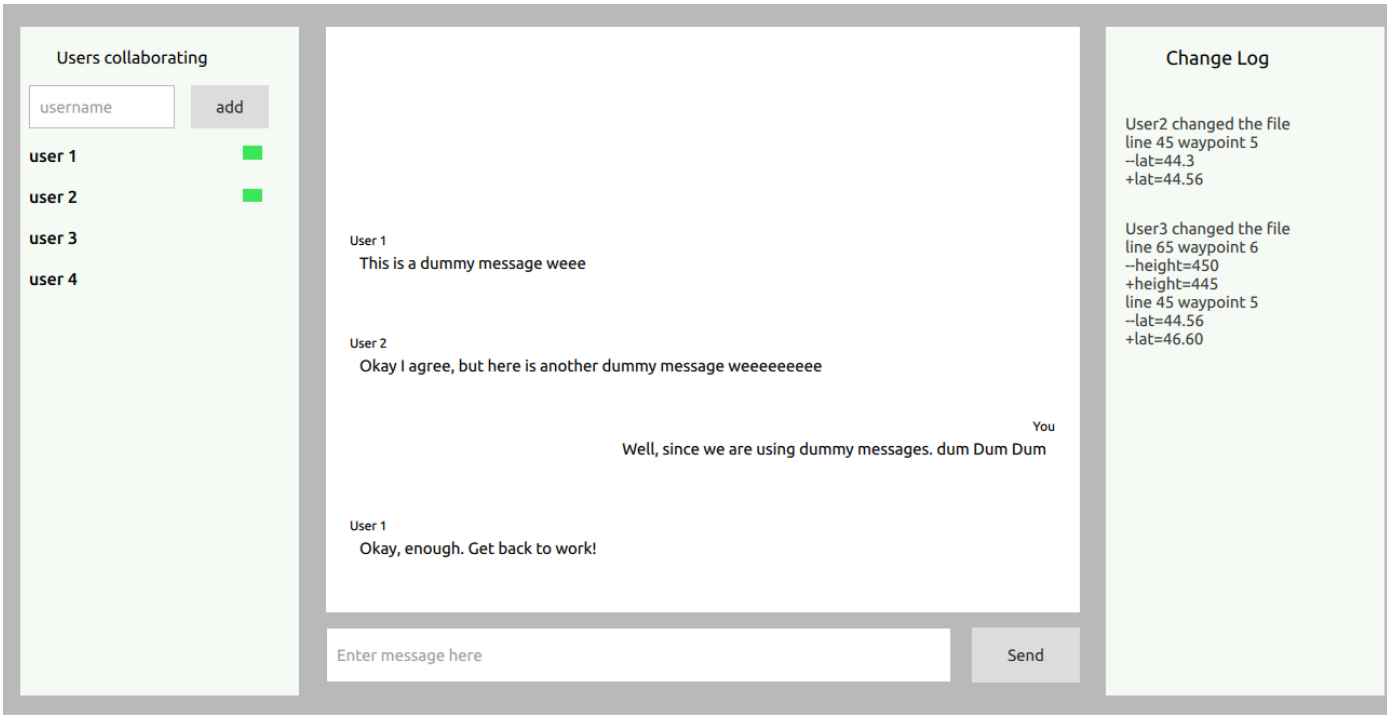
The configuration for a new project can be input by the user in a graphical manner.



This creates a project on mscolab server with a single administrator.

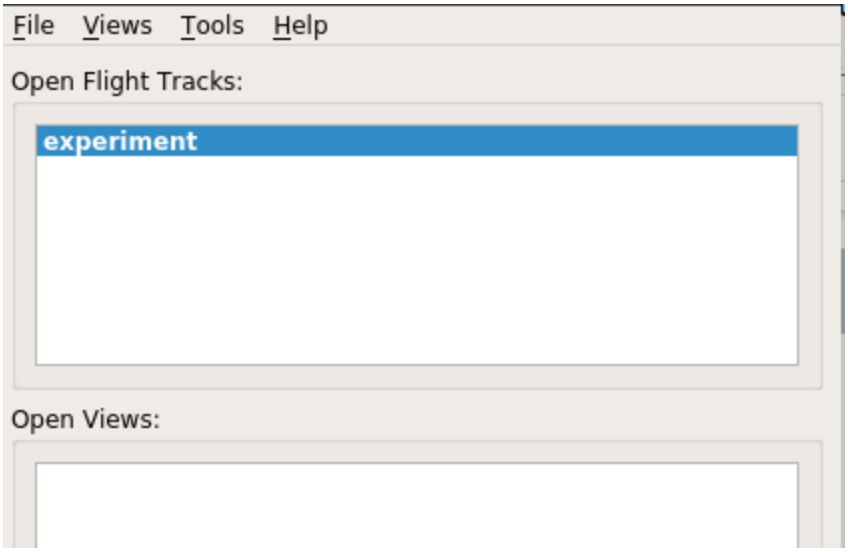
Clicking on ‘mscolab Login’ would show a dialogue-box with email-id and password, for login. If the User with email-id doesn’t exist in the mscolab database, user registration dialogue-box is opened. Once login is completed, login button gets replaced by the following display.

Opening a project which the user is administrator of would open the following window. Clicking on a project which the user is the collaborator of, opens a similar window, without options to add/remove collaborators.



- Left side of the window as seen by the user, has a list of users collaborating on the experiment.
- Right side of the window as seen by the user, has a log of recent history of changes.
- Central space lists the group chat messages which will support important markdown syntax (e.g bold, italics) during editing.

Once a new file is created or a file is opened with `mscolab`, say 'experiment.ftml', msui window gets updated with new file in the listing which can be edited in an usual manner and each change gets saved in `mscolab` server after a certain duration. The continuous backup can be disabled/enabled by the user by an UI element.

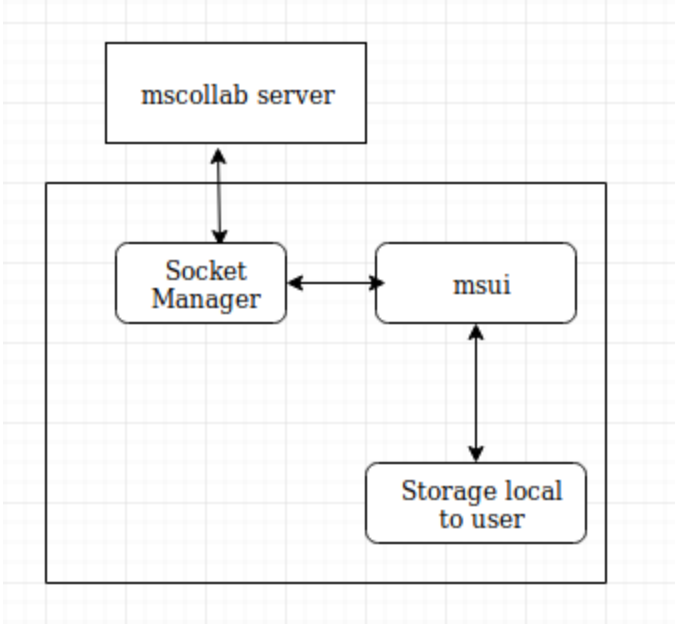


If one opens an old stored file, say 'old-experiment.ftml', following API calls are made with filename/file-id as parameter.

- /get_file handled by FileManager
  If file is not found in $data_dir, it's created in $data_dir as an intermediate save point.
- /get_authorized_users handled by FileManager
- /get_log handled by FileManager

- /get_messages handled by ChatManager

Once this data is received, it is suitably rendered to a new `mscolab-ui` window as shown in the mock-up. The overall data-flow diagram in front-end would resemble the following.



SocketManager class would be a simple class, with two major functions:

- connect()
  Used to start connection after authentication is complete and the client receives a token
- on_message()
  Used to handle messages incoming from `mscolab` server (when SocketManager.emit() is called). Event handlers would be written in '`msui/mscolab_ui.py`'.