InstaOrbit Collaborative Module Documentation

Collaborative Features

The Collaborative Module is a core feature of InstaOrbit, enabling team collaboration and project sharing. This module allows users to create, manage, and share projects efficiently in a web-based environment. Key collaborative functionalities include:

- **User Login & Credentials**
- **Project Creation and Progress Saving**
- **Team collaborative working**
- **Public Catalog**

Workflow:

How it works:

InstaOrbit leverages a Redux store to manage the app's state, including:

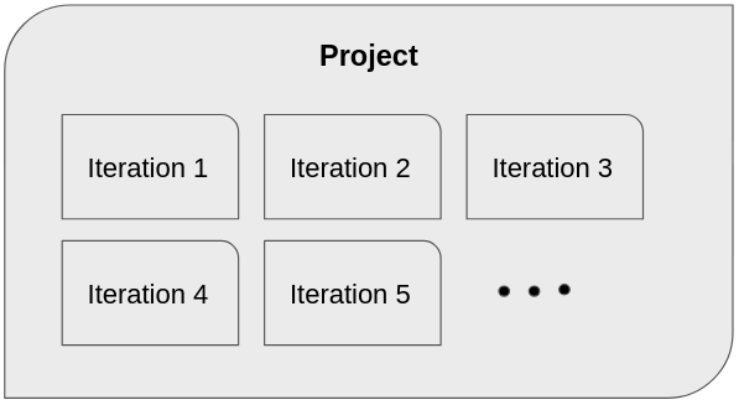
- **Object Definitions** (e.g., satellites, ground stations)
- **Configuration Settings** (e.g., simulator, viewport options)
- **Simulated Results** (e.g., time series of simulations)
- **User Credentials & Project Information** (e.g., user details, project ID)

Data Workflow:

- **Saving Data**: The current state of the Redux store is saved to Firebase NoSQL Database when a user saves their work.
- **Retrieving Data**: Saved states from Firebase are loaded back into the Redux store when a user retrieves a project.

Key terminologies:

Figure 1. File Structure



Iteration:
An Iteration represents a single saved state of the Redux store. It acts as a discrete data packet and is stored as an individual document in the database. Each iteration has:

- A unique name
- Creation date
- Assigned user

Project (Trajectory):
A Project is a higher-level organizational structure akin to a folder. It contains multiple iterations and is associated with a specific user. Projects allow for the storage and management of several iterations under one umbrella.

User Workflow:

The Collaborative Module is accessed through the **Top Bar** and functions independently from other features. This means all other features in InstaOrbit remain fully operational, even if the collaborative features are disabled. Below are the key workflows for users within the module.

For new users, the top bar provides login and signup options. Users can either create an InstaOrbit account or use Google for authentication. Once logged in, additional buttons will appear in the interface, enabling collaborative actions:

- Open
- Versions
- new
- Save
- Save As

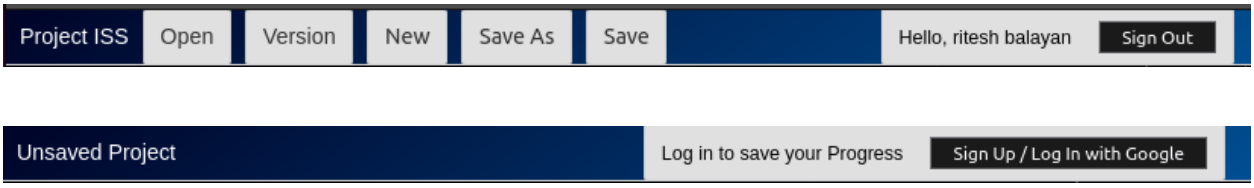


Figure 2. Top Bar for Authenticated User and new user

- **Open/Versions**
  - **Open**: Clicking "Open" brings up a pop-up window displaying a list of the user's saved projects.
  - **Versions**: Clicking "Versions" displays a list of iterations within an existing project. Selecting a trajectory (project) will load the corresponding state from Firebase into the Redux store.

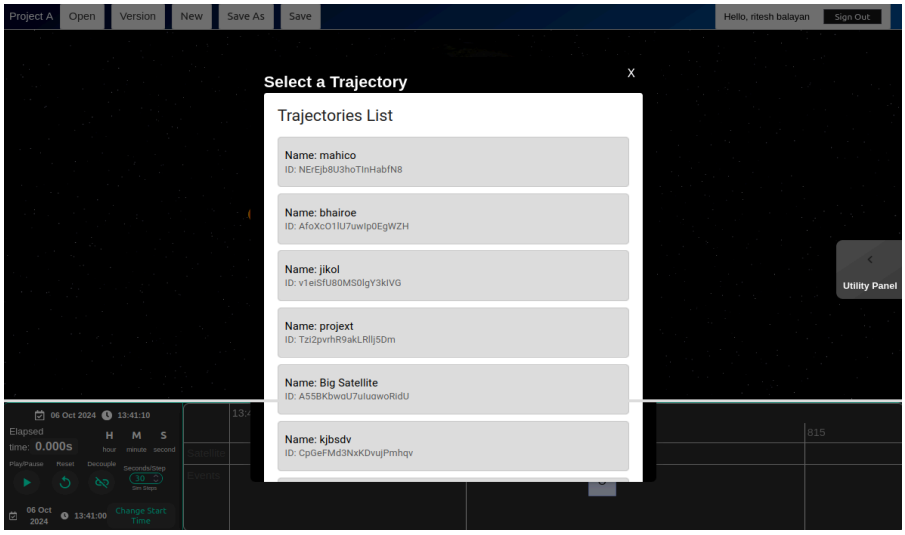


Figure 2. Pop up window to see list of projects

- **New**: A new user can create a project.
- **Save & Save As**
  - **Save**: Saves the current progress in the same iteration, updating the existing data.
  - **Save As**: Creates a new iteration within the same project, allowing users to save progress separately while keeping the last iteration unchanged.

Features to be Added:

Team Collaboration: Adding other users to the project. Here the key challenge is managing conflicts when two users make changes in the same file.

Public Store/Wall: Where all the public projects are available, Also add getting started projects here.

Social Sharing: Sharing Iteration in a public catalog, where every user can try design. And even save design in their own project and make modifications.

Important Distinction

- ***Team Collaboration****: When working in a team, the entire project is shared with members to allow collaborative design and iteration management.*

- ***Social Sharing****: When sharing socially, only an iteration is shared publicly. This allows others to experiment with a snapshot of your project without accessing the whole project.*

Understanding CodeBase:

Code base of this module is in mostly independent folders.

Functional Components:

FireBase Functions are in Folder firebase inside src:

```
src
  |---firebase
  |       |----firebase.jsx
  |       |----firebaseUtils.jsx
  |       |----googleauth.jsx
  |       |----login.jsx
  |       |----signout.jsx
  |       |----signup.jsx
  |
  |---Store
  |       |----store.jsx
  |       |----timeSlice.jsx
  |       |----satelliteSlice.jsx
  |       |----viewSlice.jsx
  |       |---- …
  |
```

firebase.jsx : It is a configuration file of Firestore
googleauth.jsx , login.jsx, signup.jsx, signin.jsx : Are pretty much Self Defined

firebaseUtils.jsx:

It is most important file where most important functions are defined:

- ***newTrajectory*** *(name)*
  Create a new project for authenticated user with project name (name)

- ***fetchTrajectories*** *()*
  Gives list of projects of authenticated user as output.
- ***fetchIterations*** *(projectId)*
  Gives list of trajectory in given project as output.
- ***uploadIteration*** *(projectId, stateSlice, ItterationName)*
  Add a new document in Firebase, In a project. with the name "ItterationName". where stateSlice is a data packet of selected Slices from Redux.
- ***updateIteration*** *(projectId, stateSlice, ItterationName)*
  Override the document (Iteration) file.
- ***uploadAutoSave*** *(projectId, stateSlice)*
  This function is supposed to save the file periodically in a new folder "AutoSave" (TBD to over ride of make new file for each autosave of hybrid approach)
- ***downloadIterationState*** *(projectId, IterationId, fieldName)*
  It will download a given field from document Iteration of folder project to the Redux store and replace existing one.

Render Components:

Render Components of collaborative features are only accessible in Top Bar in view port and are configured in Folder:

```
src
  |---TrajectoryPlanner
  |       |----Windows
  |       |       |---Topbar
  |       |       |       |---TopBar.jsx
  |       |       |       |---TrajectoryList.jsx
  |       |       |       |---ItterationList.jsx
```

- **TopBar.jsx:** *Have Render element of top bar*
- **TrajectoryList.jsx:** *Handle popup of Project List*

- **ItterationList.jsx:** *Handles popup of Iteration List*

Some Dependency in Store (redux):

Every state in the redux store needs to be added an extra reducer to be used in firebase.

```
import { createSlice } from '@reduxjs/toolkit';
const authSlice = createSlice({
 name: 'auth',
 initialState: {
   user: null,
 },
 reducers: {
   setUser: (state, action) => {
     state.user = action.payload;
   },
   clearUser: (state) => {
     state.user = null;
   },
 },
 extraReducers: (builder) => {
   builder.addCase('SET_AUTH', (state, action) => {
     return action.payload;
   });
 },
});
export const { setUser, clearUser } = authSlice.actions;
export default authSlice.reducer;
```

An arbitrary example of a redux Slice, Highlighting underline part required for this module

ExtraReducer will communicate by unloading or loading the entire Slice on call. Make sure extraReducer is defined while creating Slice to be used in a collaborative module.

Note: Redux Function are Defined in Store Folder in src. Each Slice is defined in a different file. Also Note Slice in redux can be seen as an independent state with its own actions. Redux Configuration is in file Store.jsx where all slices are registered to make a global redux state.

Note: One more Important part of the code base to look into is Slice called workingProject in Redux. It is to store the current project and iteration ID of the project in the local Redux store so when calling firebase for actions we know what project data we need.

Firebase File Schema

trajectories [Collection]
- trajectoryID [doc]*
  - *createdOn [timestamp]*
  - *name [string]*
  - *users [array]*
    - *user ID [String]**
  - autosave [Collection]
  - iterations [Collection]
    - IterationID [doc]*
      - *createdOn [timestamp]*
      - *name [string]*
      - *State [Map]*

users [Collection]
- *userID* [doc]*
  - *trajectory [array]*
    - *Trajectory ID [string]**

Legend
(Colour) = Fields
* = multiple Entries