Colab

Collaborative Live Share Editor

Implemented by

Mourad Mohamed Al-Sheriey
Youssuf Amr Abdelgalil
Abdelrahman Tarek Ali
Ahmed Amr Al-Akwah
Mohammed Fadl Al-Hdrmi

Brief Idea Documentation

TABLE OF CONTENTS

1. Abstract

Colab is a Collaborative Live Share Editor, an editor connecting both tutors and students and going as far as offering a sanctuary to those seeking a shared development environment.

The need for Colab emerges from day to day educational experiences of students, mentors and instructors in several learning environments, especially that of a programming lab.

Traditionally, in order to provide some form of personalized experience, students attend having mentors directly and inefficiently supervising them.

The inefficiency spouts from the fact that normally, the number of mentors is small, and that they've to be physically available, jumping around from a student to the other.

Colab enables both parties to be connected in a single session where students can write their code, complete their tasks, and have it remotely reviewed by their tutors who can help and evaluate them, monitor their progress and efficiently provide them with a higher quality of personalized learning experience.

Moreover it motivates software development techniques like pair programming aimed at providing a better software quality and increasing programmers productivity, team morale and satisfaction.

And then there are interviews where the best measure for an individual's skill is for it to be shown off on a live share with the interviewer.

Colab, powered by its utility, comes as a powerful and adaptable solution to all presented use cases and more…

2. Background

2.1. Introduction

With various technologies coming into the picture, we've reached an era where digital transformation is a need rather than a choice and it's not necessarily about the technology itself but using it in contexts where it can people solve their traditional problems more easily and efficiently.

Whether it is in academia or industry, collaboration strikes as a problem that needs to be handled to achieve a faster progress.

Colab, as a collaborative live share editor, comes to provide an effective and efficient solution for different use cases of collaborative editing.

2.2. Motivation

Colab is motivated by the need for collaboration whether it's the day to day educational experiences of students, mentors and instructors in several learning environments or the application of modern software development techniques like pair programming aimed at providing a better software quality and increasing programmers productivity, team morale and satisfaction.

2.3. Beneficiary

The following are the parties that will benefit from the project upon completion:

- **Universities**
- **Tutors**
- **Learners**
- **Developers**
- **Interviewers**

2.4. Main Development Techniques

The following are the main development techniques used throughout the project:

- **Operational Transformation**

  To support lock free and non-blocking collaboration functionalities.

- **Socket Programming**

  To create the underlying communication protocols.

- **Event-driven Programming**

  To conform to the programming style used by socket.io as the underlying library on which socket programming is done.

2.5. Main Development Applications

The following are the main development applications used throughout the project:

- **WebStorm**

  A powerful IDE published by Jetbrains and used for modern JavaScript development.

- **Adobe XD**

  A powerful tool published by Adobe XD and used for prototyping web UX.

2.6. Main Development Libraries

The following are the main development libraries used throughout the project:

- **NodeJS**

  As the backend runtime environment.

- **ReactJS**

  As the main frontend development library.

- **Socket.IO**

  To handle communication between web clients and servers through a client side library that runs in the browser and a server side library for NodeJS.

3. Problem Definition

The problem can be viewed from different angles, one of which is that of a programming lab, traditionally, in order to provide some form of personalized experience, students attend having mentors directly and inefficiently supervising them.

The inefficiency spouts from the fact that normally, the number of mentors is small, and that they've to be physically available, jumping around from a student to the other.

Another arises from the application of modern software development techniques such as pair programming and its physical limitation requiring both programmers to be in the same place.

And then there are interviews where the best measure for an individual’s skill is for it to be shown off on a live share with the interviewer.

4. Related Work

4.1. Codeshare

Codeshare is an online code editor that can be used for interviews, pair programming, and teaching.

It provides a limited number of features including:

- Editing sessions expiring within 24 hours.
- Voice chat among session users.
- Syntax highlighting for a huge variety of programming languages.

In contrast to Colab, it lacks the following:

- Dynamic editing privileges as it allows either one or all to edit at any time.
- Code compilation and automated task evaluation.

4.2. Cloud9

Cloud9 is an integrated development environment supporting a variety of programming languages.

It includes most of the features required by an editor but it lacks in terms of usability and collaboration utility.

4.3. Plunker

Plunker is a known real-time collaboration tool to prototype, experiment, share and debug ideas.

Its main features include:

- Desktop integration enabling basic drag and drop between their editor and the client's file system.
- Github imports.

In contrast to Colab, the key difference is that it lacks:

- Flexibility in regards of the project's type as it only supports certain technologies and development environments such as Angular and React.
- Actual utility to support other use cases such as teaching and interviews.

5. Project Specifications

5.1. System Architecture

Colab's general system architecture can be viewed as a client-server architecture where a server connects two major parties, the session master and the session users.

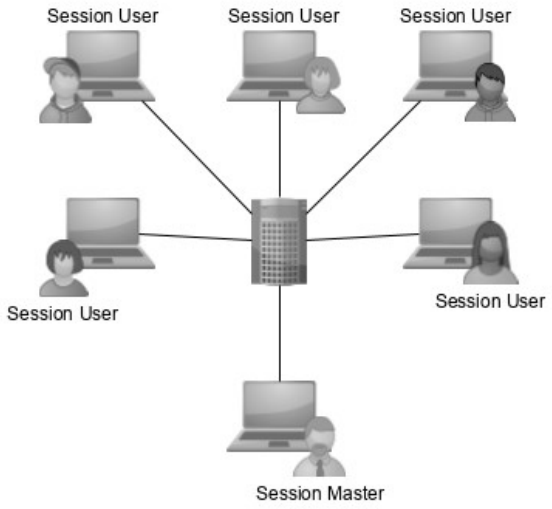

Figure 1: Colab's General System Architecture

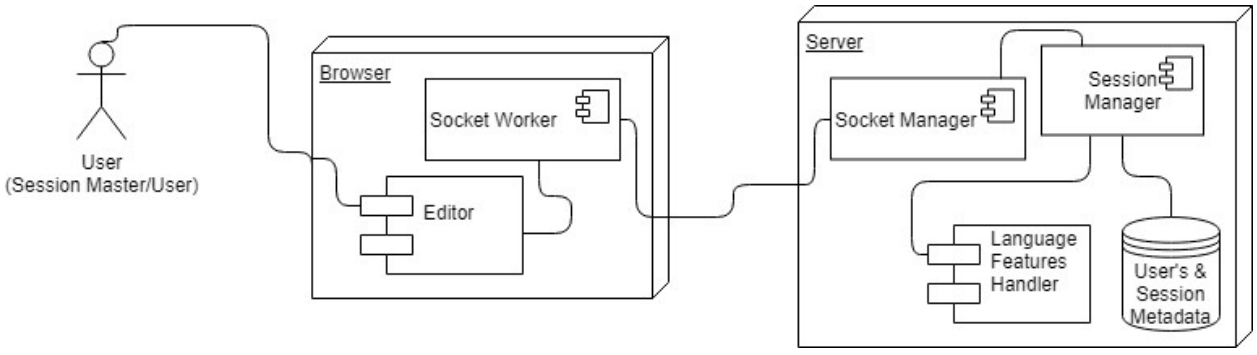

Figure 2: Colab's System Architecture

5.2. Block Diagram

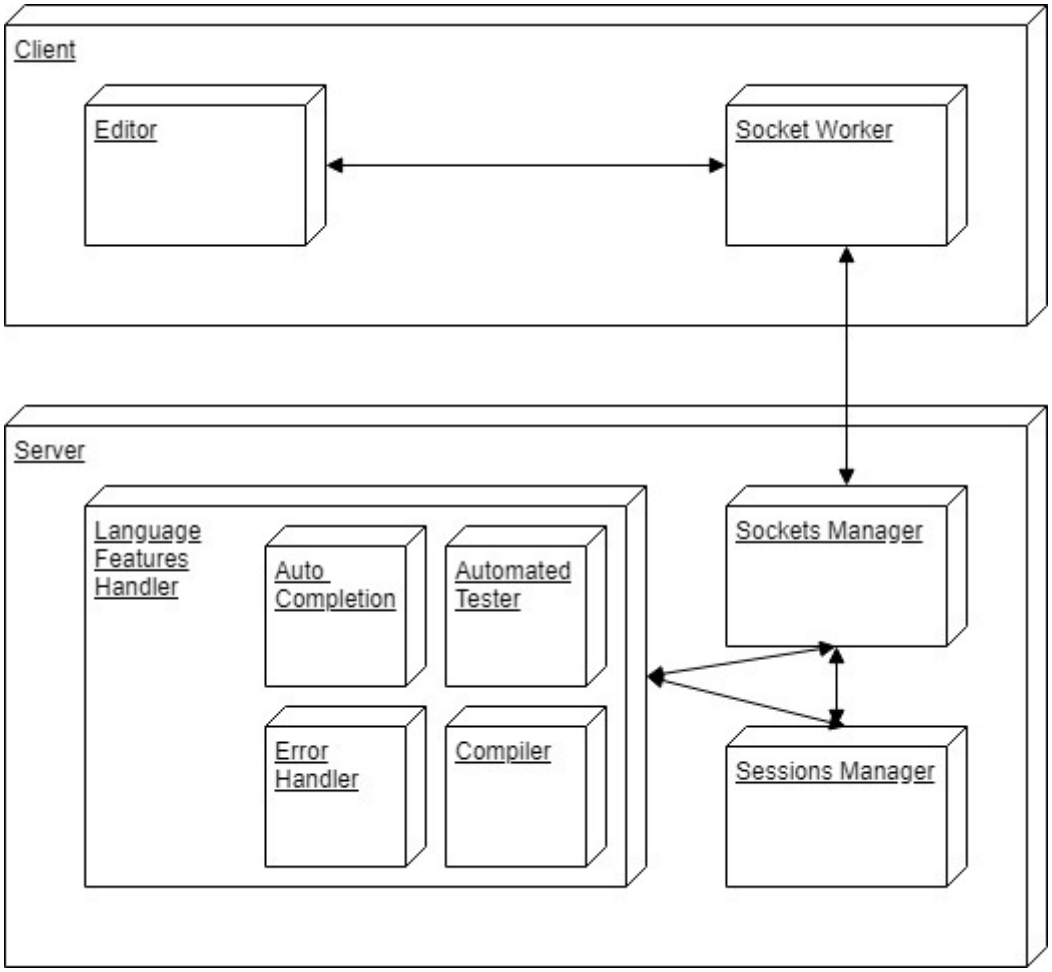

Figure 3: Colab's Block Diagram

5.3. Functional Requirements

The following represent Colab's core functional requirements:

- Guest can sign up as user.
- Guest can login as guest or user.

- User can create a new session.
- User can search for sessions by name.
- User can join sessions through invitation links.
- User can log out.

- Sessions are referenced by name.
- Sessions include text chat.

- Session master can invite users.
- Session master can remove session users.
- Session master can manage session permissions.
- Session master can publish session tasks.
- Session master can view session task solutions.
- Session master can view session task status.
- Session master can view session task help requests.
- Session master can answer session task help requests.
- Session master can terminate mastered session.

- Session user can access editor's module.
- Session user can run editor's module code.
- Session user can preview compilation output.
- Session user can request help on a session task.
- Session user can submit a session task solution.

- Editor's module supports collaborative editing.
- Editor's module can query language features handler.

- Language features handler supports auto-completion.
- Language features handler includes an error handler.
- Language features handler includes an automated tester.
- Language features handler provides access to a different set of compilers.

5.3. Non-functional Requirements

The following represent Colab's non-functional requirements:

- **Data Integrity**

  Collaborative editing must ensure the accuracy and consistency of data between collaborators.

- **Performance**

  Real-time collaborative editing must be done within a maximum of 1 second of delay.

- **Usability**

  User actions should be performed with the minimum number of clicks.

- **Portability**

  Colab as a web application must ensure portability across different platforms and browsing environments.

- **Adaptability**

  UX must be designed to allow for different use cases such as teaching, pair programming and interviews.

5.4. Use Case Diagram

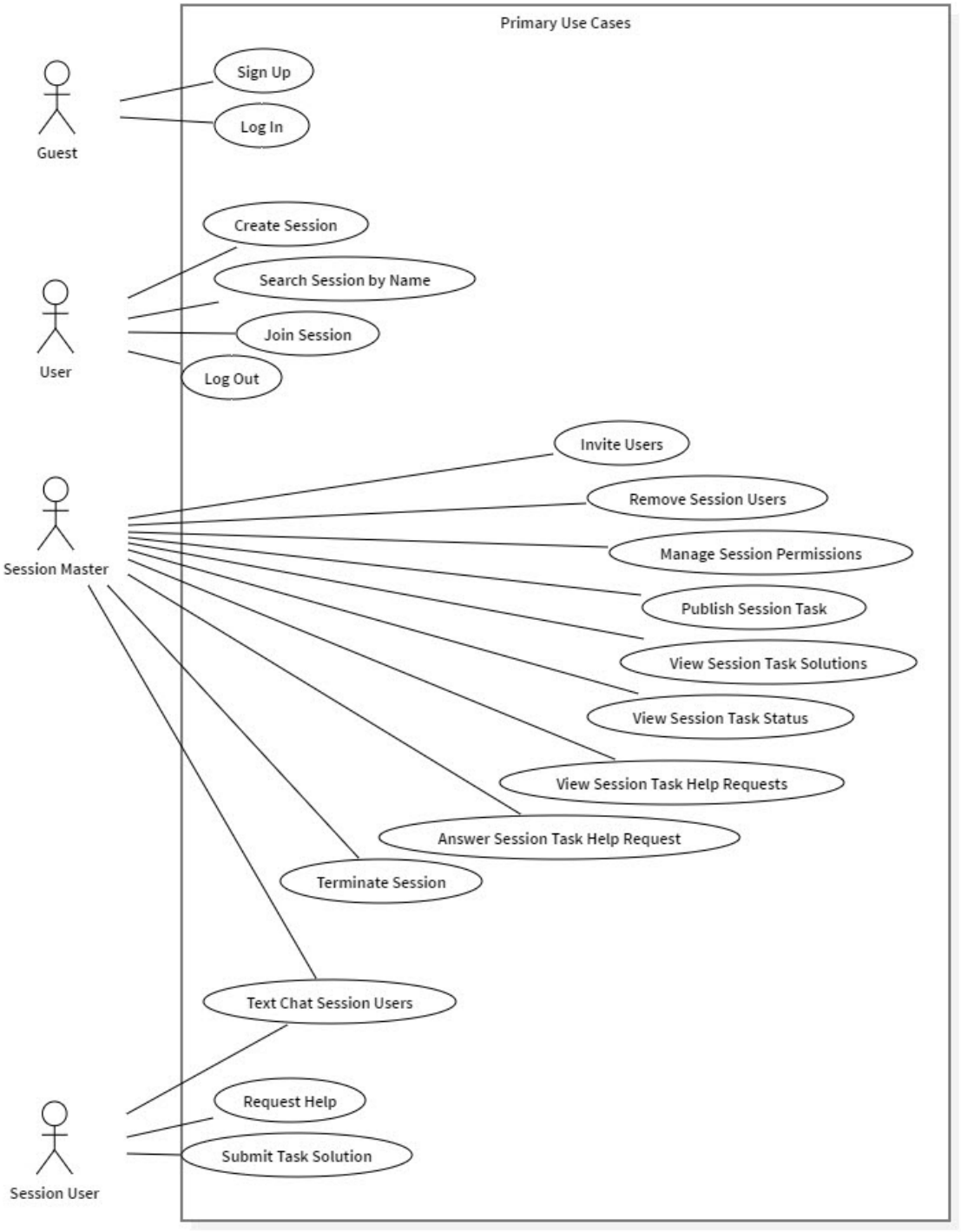

Figure 4: Colab's Primary Use Case Diagram

5.5. Sample Use Cases

<table>
<tr><td>Use Case ID</td><td colspan="2">USER_ACTION_0</td></tr>
<tr><td>Use Case Name</td><td colspan="2">User Creates Session</td></tr>
<tr><td>Actors</td><td colspan="2">Session User</td></tr>
<tr><td>Pre-conditions</td><td colspan="2">User is logged in with an account or as guest.</td></tr>
<tr><td>Post-conditions</td><td colspan="2">A session is created within the system.</td></tr>
<tr><td rowspan="6">Flow of Events</td><td>User</td><td>System</td></tr>
<tr><td>Types the session name to be created.</td><td></td></tr>
<tr><td>Presses “Create Session” button.</td><td></td></tr>
<tr><td></td><td>Creates session with the specified name.</td></tr>
<tr><td></td><td>Prompts the success of session creation.</td></tr>
<tr><td></td><td>Joins the user into the session as “Session Master".</td></tr>
<tr><td rowspan="3">Exceptions</td><td colspan="2">System</td></tr>
<tr><td colspan="2">Session couldn’t be created.</td></tr>
<tr><td colspan="2">Prompt user with the failure message.</td></tr>
</table>

Use Case ID	USER_ACTION_1	
Use Case Name	User Joins Session	
Actors	Session User	
Pre-conditions	User is logged in with an account or as guest.	
Post-conditions	The user is joined into the session.	
Flow of Events	User	System
	Selects the session to be joined.	
		Joins the user into the session as “Session User”.
Exceptions		

Use Case ID	SESSION_MASTER_ACTION_0	
Use Case Name	Session Master Updates Code	
Actors	Session Master	
Pre-conditions	User is joined into the session.	
Post-conditions	Content/Code of users connected to the session is updated with the new code.	
Flow of Events	User	System
	Types new code.	
		Receives “Updates Message”.
		Broadcasts “Update Message” to the other connected users.
	Receive “Update Message”	
	Update content with the updated segment.	
Exceptions		

5.6. Sequence Diagrams

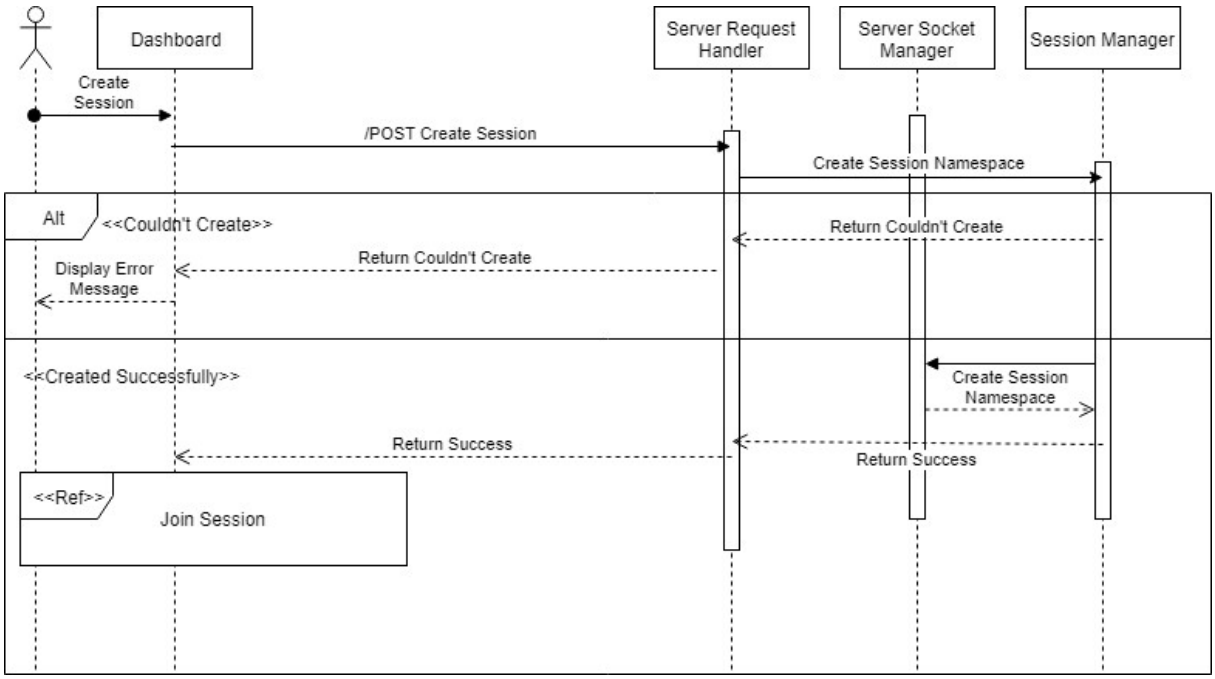

Figure 5: Colab's Create Session Sequence Diagram

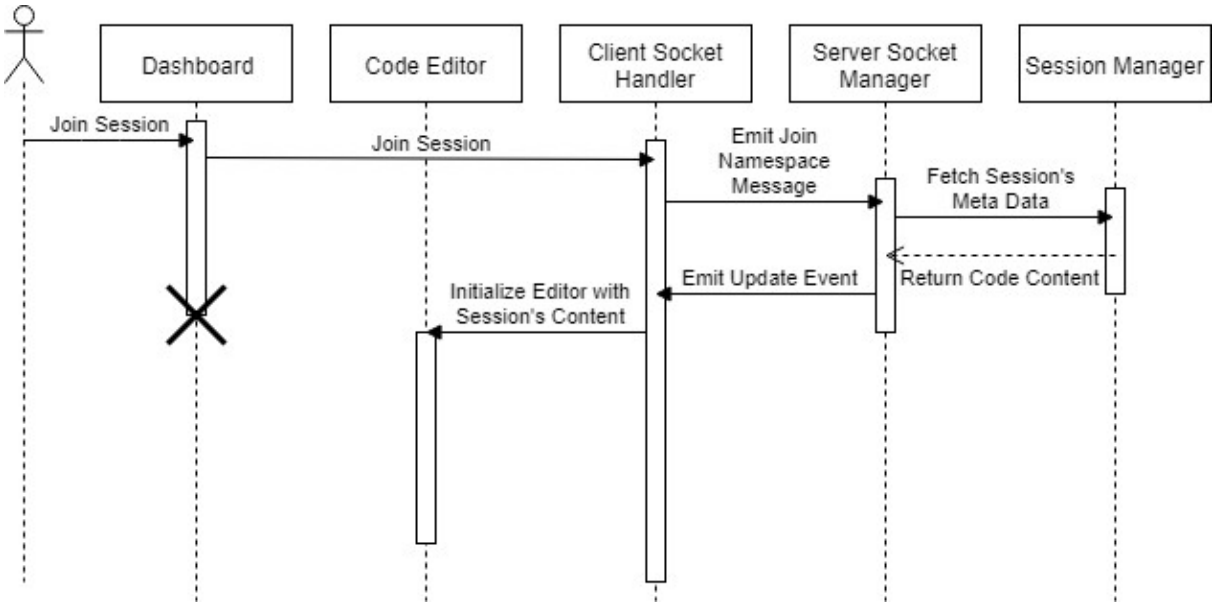

Figure 6: Colab's Join Session Sequence Diagram

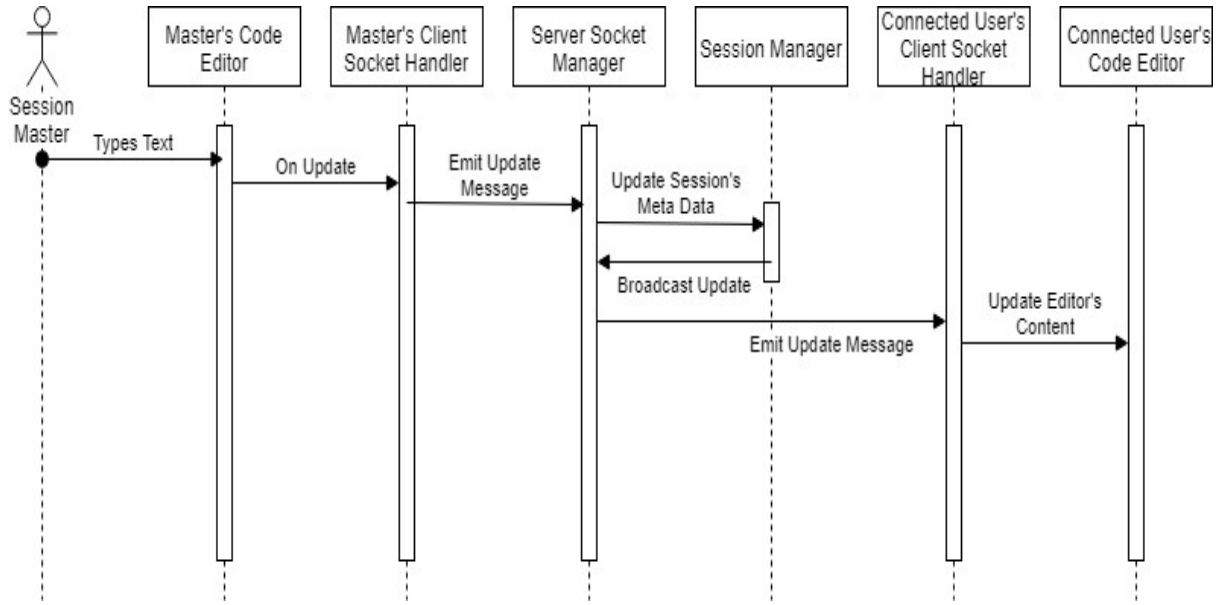

Figure 7: Colab's Session Master Updates Code Sequence Diagram

5.7. Entity Relationship Diagram

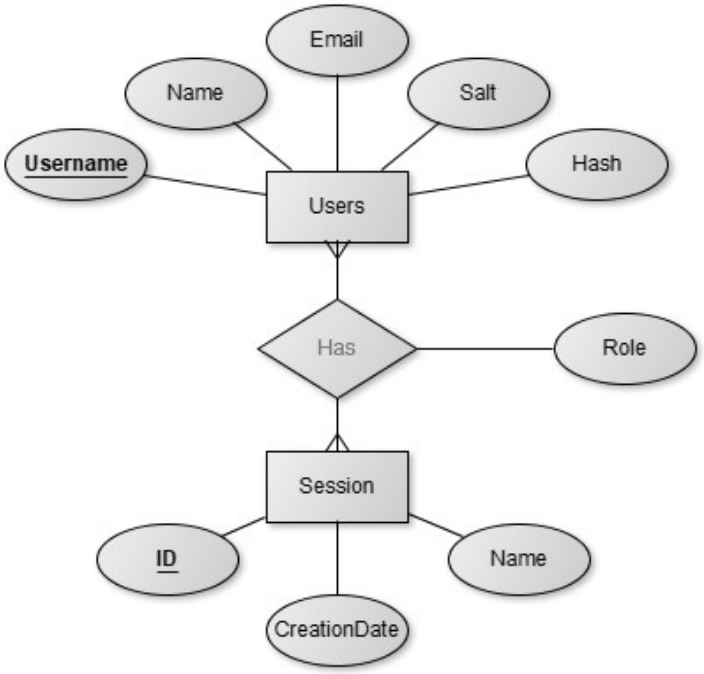

Figure 8: Colab's Entity Relationship Diagram