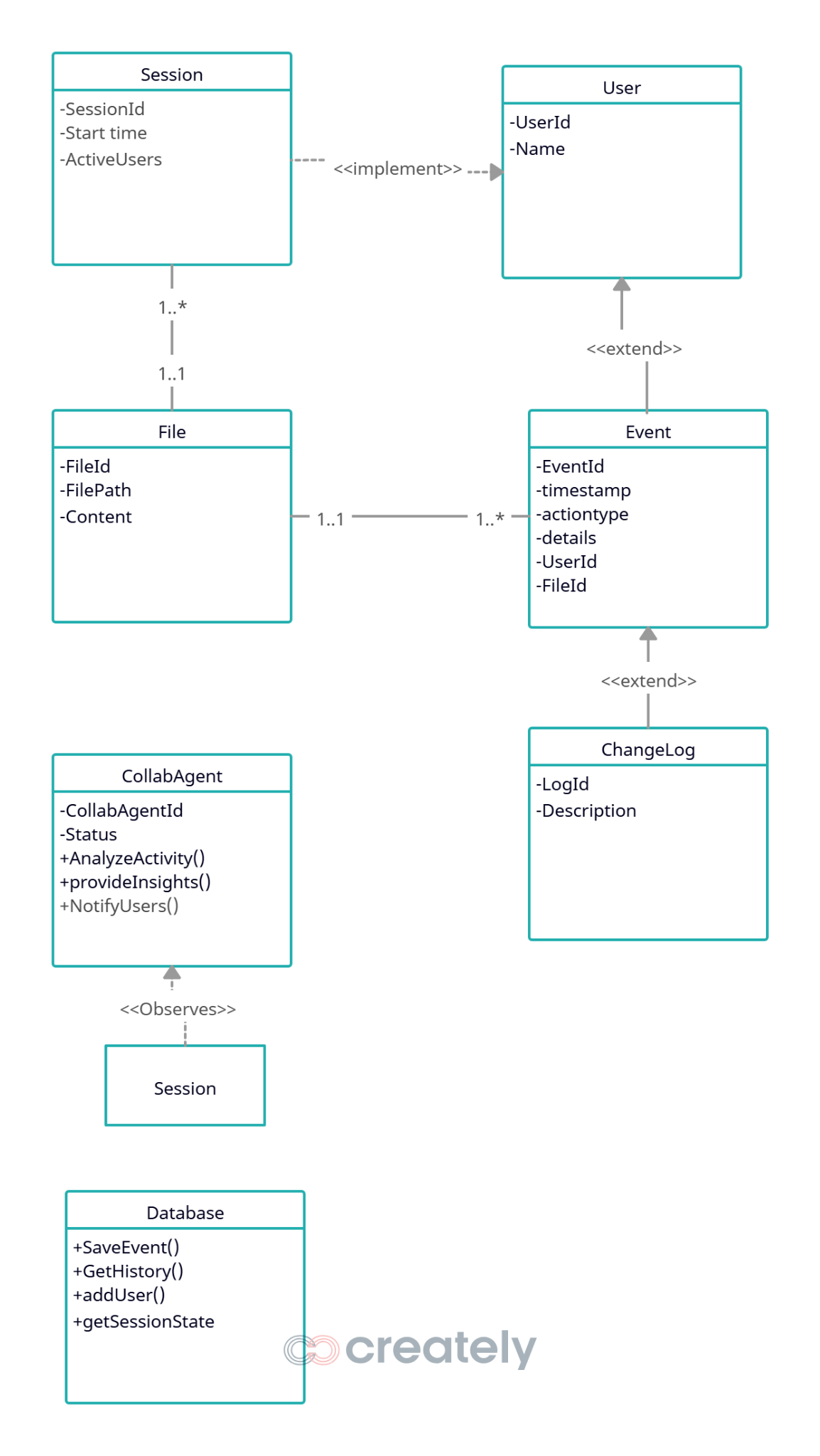
**CollabAgent01  
Benjamin O'Neill, Andrew Rush, Jaryn Hernandez, Nicholas Phillips, Alphin Shajan  
Component Overview** The AI - Assisted Collaboration assistant project will contain components, and technologies, including making use of a **VSCode extension ‘Live Share’** for in real time coding collaboration. We will also be making use of a **Database** to store information for its users.

## **Main Application**

**VS Code Extension (TypeScript, Node.js)** The core of the system is a VS Code extension that enables real-time collaborative editing across teams.

🔹 **Key Features:**

* **Collaboration Sessions** – Users can start or join named sessions with unique invite codes.
* **Live Editing Sync** – Every keystroke, cursor movement, and file open event is synced instantly across all team members.
* **Colored Cursors & Highlights** – Each user gets a unique color to identify their edits.
* **Session Management** – Sidebar shows active members and connected session status.

## **Backend Services**

**Flask (Python)** The backend manages communication, synchronization, and activity monitoring between extension clients and the Agent Bot.

🔹 **Key Features:**

* **Session Control** – Creates, joins, and maintains collaboration sessions.
* **File/Change Broadcasting** – Distributes edits and events in real time.
* **Activity Logging** – Records user actions such as file opens, edits, and saves.
* **Bot Integration** – Relays activity updates and status changes to the Agent Bot.

## **Agent Bot**

**Collaboration Assistant** The Agent Bot helps teams stay coordinated by monitoring activity and providing intelligent feedback inside the extension sidebar.

🔹 **Key Features:**

* **Activity Monitoring** – Posts updates like “Ben opened auth.js” or “Nick is editing Login.jsx.”
* **Manual Status Updates** – Users can broadcast what task they’re working on with one click.
* **Smart Task Detection** – Suggests likely tasks based on recent file activity and editing patterns.
* **Team Feed** – Maintains a scrollable history of team actions for easy reference.

## **Database & Authentication**

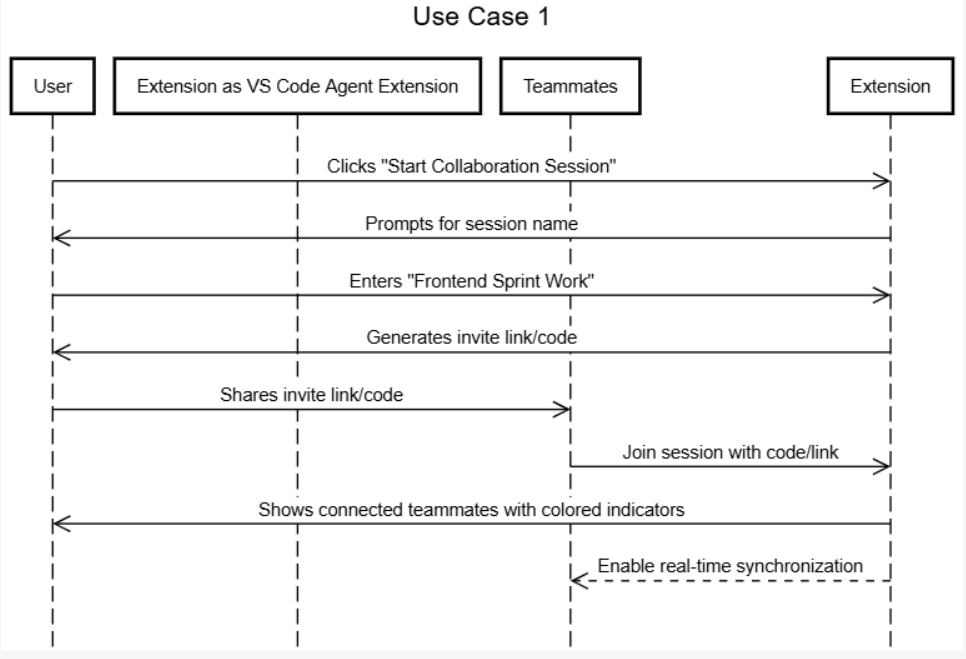
**(Planned – PostgreSQL/Supabase)** The system will use a database service to store and manage collaborative data.

🔹 **Planned Features:**

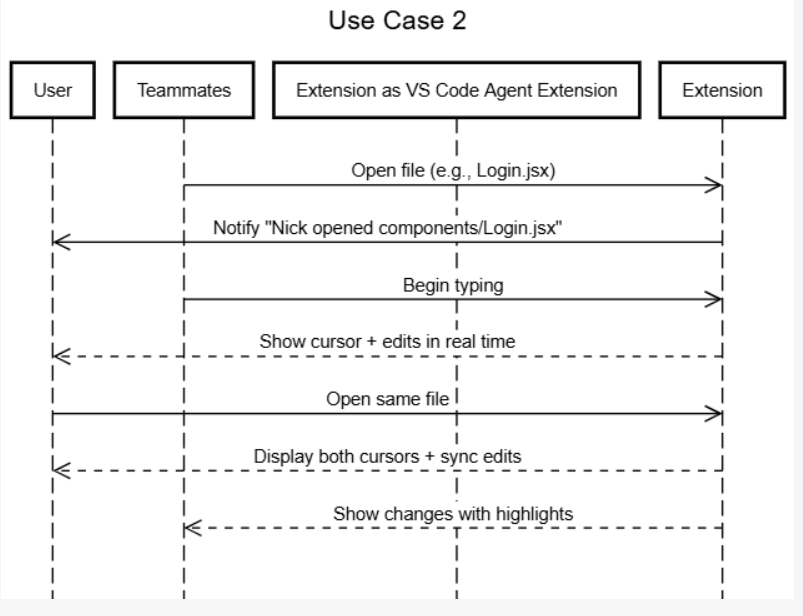
* **User Authentication** – Secure login and session tracking.
* **Session Storage** – Saves collaboration session details and connected members.
* **Activity History** – Stores file edits, bot messages, and status updates for analytics.
* **Real-Time Sync** – Ensures updates propagate instantly across all users.

**Use Case Sequence Diagrams**

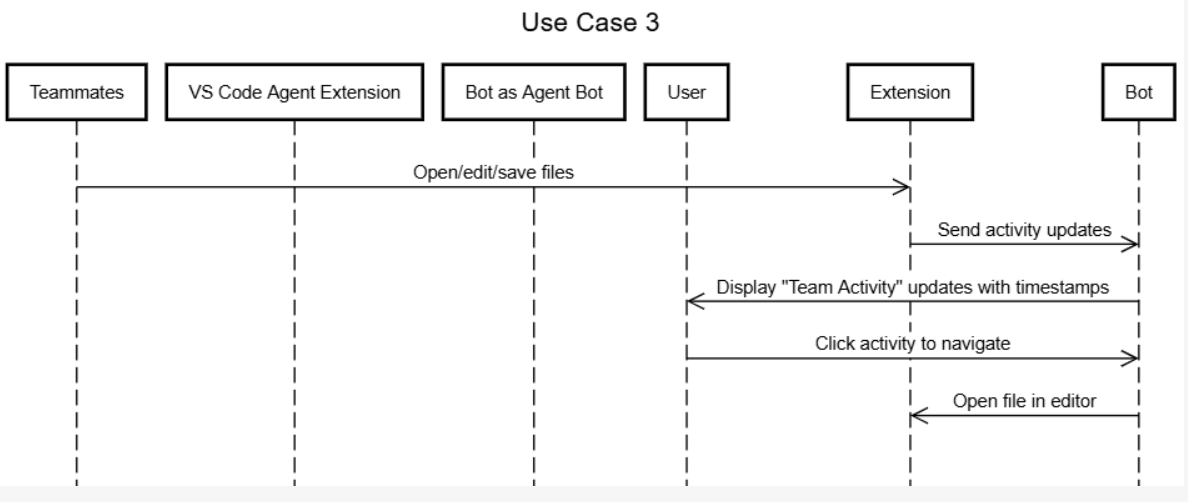
## **Use Case 1 - Real-Time Code Synchronization Setup**



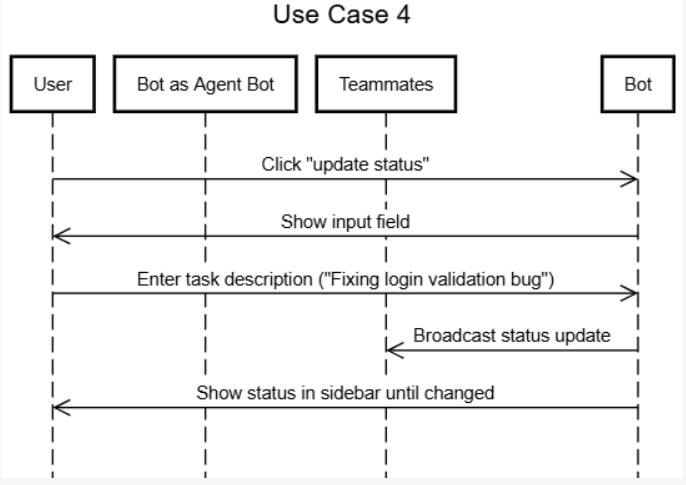
## **Use Case 2 - Live Editing Collaboration**



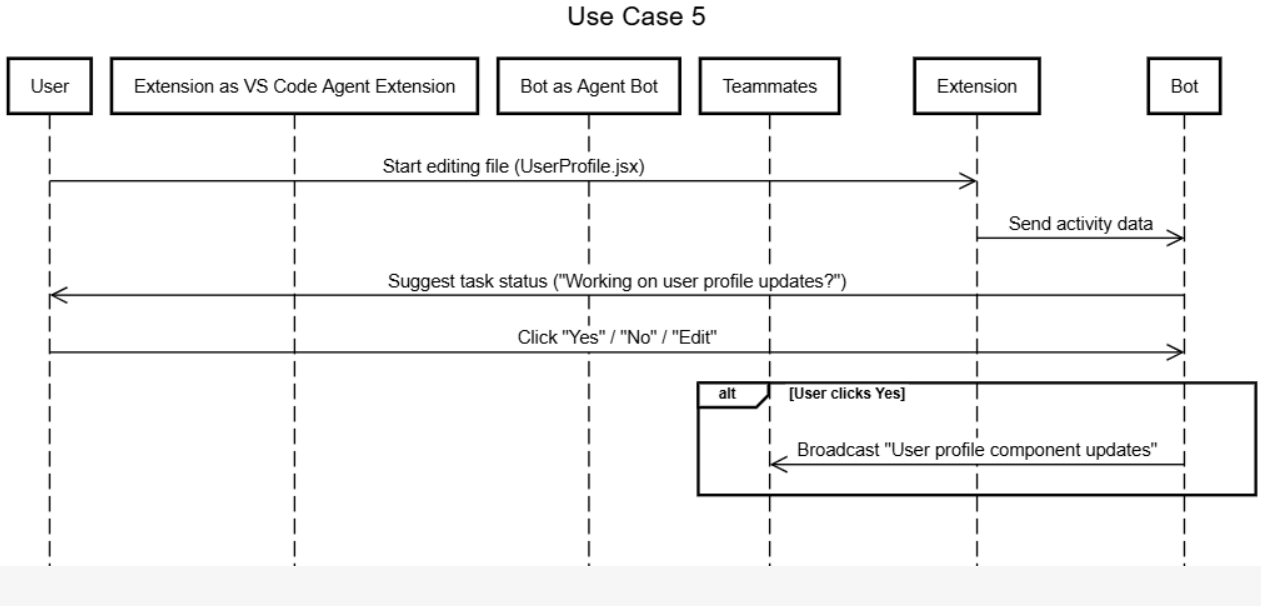
## **Use Case 3 - Agent Bot Activity Monitoring**



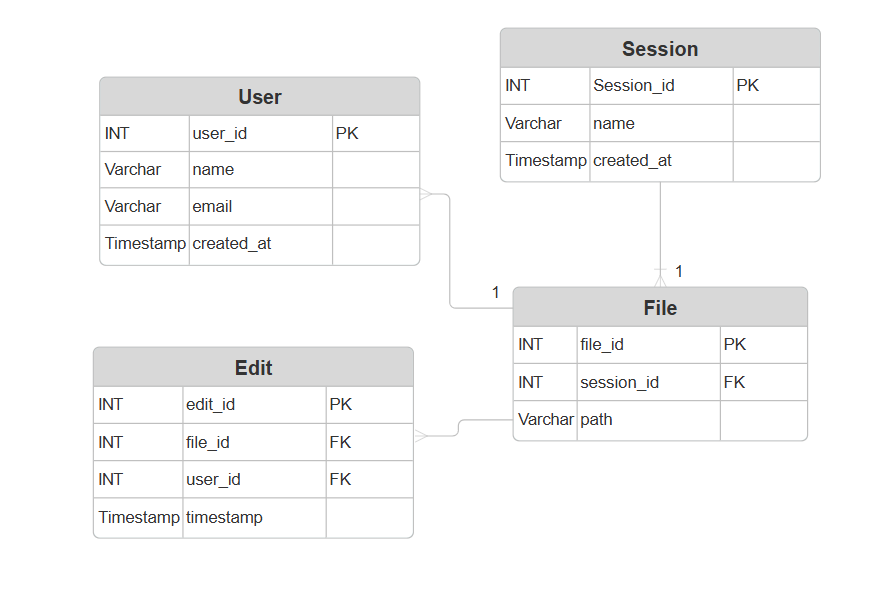
## **Use Case 4 - Manual Task Status Updates**



## **Use Case 5 - Agent Bot Smart Task Detection**



**Entity-Relationship Diagram**

****

**Table Design**

1. **User Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| user\_id | INT | PK, AUTO\_INCREMENT |
| name | VARCHAR(100) | NOT NULL |
| email | VARCHAR(150) | UNIQUE, NOT NULL |
| created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP |

1. **Session Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| session\_id | INT | **PK**, AUTO\_INCREMENT |
| name | VARCHAR(100) | NOT NULL |
| created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP |

1. **File Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| file\_id | INT | **PK**, AUTO\_INCREMENT |
| session\_id | INT | **FK** → Session(session\_id) |
| path | VARCHAR(255) | NOT NULL |

1. **Edit Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| edit\_id | INT | **PK**, AUTO\_INCREMENT |
| file\_id | INT | **FK** → File(file\_id) |
| user\_id | INT | **FK** → User(user\_id) |
| timestamp | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP |

**Development Environment**

Required Hardware

* Minimum of 8GB of RAM

Tools

* IDE
  + Visual Studio Code for extension
  + Any other text editor
* Package Managers
  + NPM for Visual Code extension
* Languages
  + TypeScripts
    - For extension
  + Python
    - For creating an API

**Version Control**

This project is managed over **Git** and **GitHub,** which will use the following components

* Extention
* Docusarious Documentation

**Branching Strategy**

**Main Branch**

* Main will hold the most up-to-date and stable version of the project.
* Changes will be merged via pull requests

**Sprint Branch**

* At the start of each spring there will be new branches created to ensure that what gets updated won’t immediately be pushed to main to safeguard any unforeseen errors
* Will accumulate any new features added during the sprint

**End of Spring Merging**

* At the end of each sprint will review the sprint and push the approved features
* Ensures anything added or updated is stable before merge