User Requirements Document (URD)

1. Introduction

1.1 Purpose

The purpose of this document is to outline the user requirements for the **Collaboration Station** project. This platform is designed to support high schoolers with neurodivergence, especially Autism, in learning software engineering concepts through collaborative game development in Scratch. The document serves as a guide for developers and stakeholders to ensure that the final product meets the needs of the users.

1.2 Scope

The Collaboration Station project is a real-time, web-based platform that allows users to collaboratively create games in Scratch. It includes features like real-time editing, audio/video chat integration, breakout rooms, and secure user authentication. The platform is designed for use by children with Autism and their educators or facilitators.

2. Project Overview

2.1 Project Description

The Collaboration Station is an educational tool designed to introduce children with disabilities, particularly those on the autism spectrum, to the basics of software engineering through game development. The platform is based on Scratch, a block-based programming language, and is enhanced with features that allow for real-time collaboration, communication, and project management.

2.2 Project Objectives

- Enable collaborative game development in real-time using Scratch.
- Provide integrated communication tools (audio, video, and chat) to facilitate collaboration.
- Support educators in managing and facilitating collaborative learning sessions.
- Ensure accessibility and ease of use for children with Autism.

3. User Characteristics

3.1 Primary Users

- **Children with Autism (ages 13-18)**: The main users who will be creating and collaborating on Scratch projects. They have varying levels of comfort with technology and need an intuitive, supportive environment.
- **Educators and Facilitators**: Teachers and TAs who will guide and support the children. They require tools for managing sessions, monitoring progress, and providing assistance.

3.2 Secondary Users

- **Researchers and Developers**: Individuals involved in the development and research of the platform, requiring access to data for analysis and platform improvement.
- **Parents**: May occasionally interact with the platform to monitor their child's progress.

4. Functional Requirements

4.1 Collaboration Features

- **Real-Time Editing**: Multiple users should be able to edit the same Scratch project simultaneously, with real-time updates visible to all collaborators.
- **Live Cursors and Block Synchronization**: Display user cursors in real-time and synchronize code blocks as they are added or removed.
- **Breakout Rooms**: Allow TAs to create smaller groups within a larger session for focused collaboration.

4.2 Communication Tools

- **Integrated Audio/Video/Chat**: Seamless communication between users for video, audio, and chat within the collaboration space.
- **Mute/Unmute and Camera Control**: Users have the ability to mute/unmute and turn their cameras on/off.

4.3 User Authentication and Management

- **User Registration and Login**: Secure registration and login functionalities.
- **Role-Based Access**: Different access levels for students, TAs, and administrators.
- **Password Management**: Features for password reset, including email verification and token-based security.

4.4 Project Management

- **Project Creation and Selection**: Users can create new projects or select existing ones from a list.
- **Project Assignment**: TAs can assign or remove students from projects.
- **Save and Load Projects**: Ensure that projects can be saved and loaded efficiently with real-time updates.

5. Non-Functional Requirements

5.1 Usability

- **Accessibility**: The platform should be designed with accessibility in mind, ensuring it is easy to navigate for users with Autism.
- **User Interface**: Intuitive and straightforward, minimizing the need for extensive guidance.

5.2 Performance

- **Scalability**: The system should support a large number of concurrent users without significant degradation in performance.
- **Response Time**: Real-time updates should occur within milliseconds to ensure a seamless experience.

5.3 Security

- **Data Security**: All user data, including personal information and project data, must be securely stored and transmitted.
- **Session Management**: Secure session handling to prevent unauthorized access.

5.4 Reliability

- **Uptime**: The platform should maintain a high uptime, with minimal interruptions to service.
- **Backup and Recovery**: Regular backups should be performed to prevent data loss in case of system failures.

6. Assumptions and Dependencies

For the first iteration of Collaboration Station, we have selected AWS services for backend, Ably for cursor tracking and synchronization, Agora for communication, and Scratch GUI for frontend. The assumptions and dependencies are based on this decision, and are subject to change in the future.

6.1 Technical Assumptions

- Users have access to a modern web browser (Chrome, Firefox, Edge) and a stable internet connection.
- The platform will integrate seamlessly with existing tools like Scratch and AWS services.

6.2 Dependencies

- **Scratch**: The platform depends on the Scratch GUI for its primary user interface and coding environment.
- **AWS Services**: Reliance on AWS DynamoDB, Lambda, and SES for backend services.
- **Ably**: Dependency on Ably for real-time cursor tracking and synchronization.
- **Agora API**: Dependency on Agora for video, audio, and chat functionalities.

7. Acceptance Criteria

- The platform must allow at least 30 students to collaborate on Scratch projects simultaneously without performance issues.
- Users must be able to communicate via integrated audio and video during collaboration sessions.
- The system should handle user authentication securely and efficiently.
- All projects must be saved and retrievable by users without data loss.

8. Appendix

8.1 Glossary

- **Scratch**: A block-based visual programming language designed especially for children.
- **AWS DynamoDB**: A key-value and document database used to store user and project data.
- **AWS Lambda**: A serverless compute service that runs code in response to events and automatically manages the underlying compute resources.
- **Ably**: A platform for cursor tracking and block synchronization in collaborative environments.
- **Agora**: A platform for integrating real-time video, audio, and messaging into applications.

8.2 References

- Collaboration Station GitHub Repository: [Link]
- Scratch GUI: [Link]
- AWS DynamoDB Documentation: [Link]
- AWS Lambda Documentation: [Link]
- Ably Documentation: [Link]
- Agora Documentation: [Link]