NimbleMeet Project Documentation

**1. Introduction**

**1.1 Purpose**

The purpose of this document is to describe the implementation of NimbleMeet. NimbleMeet is a web application used for video calling, online meetings, etc. This application was built during the Engage’ 21 as a fully functional prototype with at least **one mandatory functionality** - a minimum of two participants should be able connect with each other using your product to have a video conversation.

1.2 Scope

This document describes the implementation details of the NimbleMeet. The software will consist of a two major functions. First to Host and schedule meetings, and the second to Join the meetings.

1.3 Definitions, Acronyms and Abbreviations

1.4 Description of Problem

Build a Microsoft Teams clone. A fully functional prototype with at least one mandatory functionality - a minimum of two participants should be able connect with each other using your product to have a video conversation. Use agile development methodology and integrate a surprise feature - a chat feature in your application where meeting participants can share info without disrupting the flow of the meeting.

Technologies used

2 Technologies Used

* Node.js – An open-source, cross-platform, back-end JavaScript runtime environment used to run the applications.
* React JS - a free and open-source front-end JavaScript library for building user interfaces or UI components. Maintained by Facebook React is used to develop single page progressive web apps
* Firebase – A cloud-hosted Realtime NoSQL database used to store and sync data between users in Realtime. Also used to authenticate and manage users.
* Twilio Programmable video – A SDK Build on top of WebRTC with a set of flexible APIs that integrate video into apps. This is used to create and manage video rooms and WebRTC channels.
* Twilio Programmable chat – A SDK used to create and manage communication channels for chat.

2.1 Selection of technologies

* Backend – Node.js
  + Node.js is fast, lightweight and efficient. It uses the asynchronous mode of operation, event-driven Input/output rather than using the traditional threads or separate threads for each process.
  + It provides cross-platform applications which run easily on any web. So you basically don’t need anything extra for running up a node application. You only need for making one
  + Huge package libraries and a dedicated package manager which makes it easier for installing and using required modules.
  + Large community support.
* Frontend – React
  + Easy to create reusable components and break complex UI into smaller parts.
  + Single way data flow makes it easy to handle the flow of data.
  + React only changes the component that is updated making it fast.
  + Large community support.
  + Simple and easy to learn.
* Database – Firebase
  + Firebase NOSQL database fire store is a Realtime database that stores data in form of collections and document making it easier to create a database architecture.
  + Firebase also sync the data in Realtime with all the connected clients which helps in maintaining the server-client synchronization.
  + Firebase allows different methods to login which include login via third party authentication or using email and password. It also verifies the email. This helps in creating and managing the users.
* WebRTC server – Twilio programmable video
  + Twilio uses WebRTC for making connections which is used by almost all video calling applications.
  + It manages all the video call rooms. Keeps log of every room and delivers high quality video call
  + Easy and simple client-side SDK for development.
  + Great documentation and sample applications to get started.
  + Gives option for SFU and MSU WebRTC Architecture.
* Chats – Twilio Programmable chat
  + Easy to create and maintain multiple conversation channels.
  + Keeps logs of the channels.

WebRTC Architecture

3. What is WebRTC

WebRTC allows devices to connect directly peer-to-peer and send and receive data like Realtime video and audio.

WebRTC was designed to be a peer-to-peer communication technology. This means that most of the technology development is focused around the client device. Despite this, it is also very important to have a clear understanding of the server-side infrastructure for WebRTC. Every WebRTC application must have an infrastructure, at the very least for exchange of signaling messages. Highly advanced WebRTC applications require infrastructure support for media handling as well.

3.1 Why SFU WebRTC architecture

Disadvantages of Peer-to-peer connection:

* High bandwidth requirement for more than 4 connections.
* Non-scalable as complexity increases with the number of participants.

Disadvantages of MCU architecture:

* The main disadvantage of MCU is its cost, as It decodes and re-encodes streams to compose the final stream, hence requires significant computing power by the MCU.
* A secondary disadvantage is in delay, the recomposing the picture can be hampered by packet loss on one of the links, as it must wait for the complete frame to encode.
* No flexibility for UI. As the UI is defined at the server custom UI is not possible.
* no end-to-end encryption as the video needs to be decoded at the server.

Advantages of SFU architecture:

* Highly scalable as the video need to be sent only once to the server.
* Less cost as compared to MSUs
* The architecture allows the call participant to send multiple media streams to the SFU, where the SFU may decide which of the media streams should be forwarded to the other call participants.
* Allow participants to have different video layouts.
* End to end encryption – Because an SFU doesn’t need to manipulate your video in the way that an MCU does, that means you could conceivably implement true end to end encryption where the media server doesn’t have the keys to decrypt your video and audio streams

Conclusion – using SFU Architecture for the application as it suits the best.

System Architecture

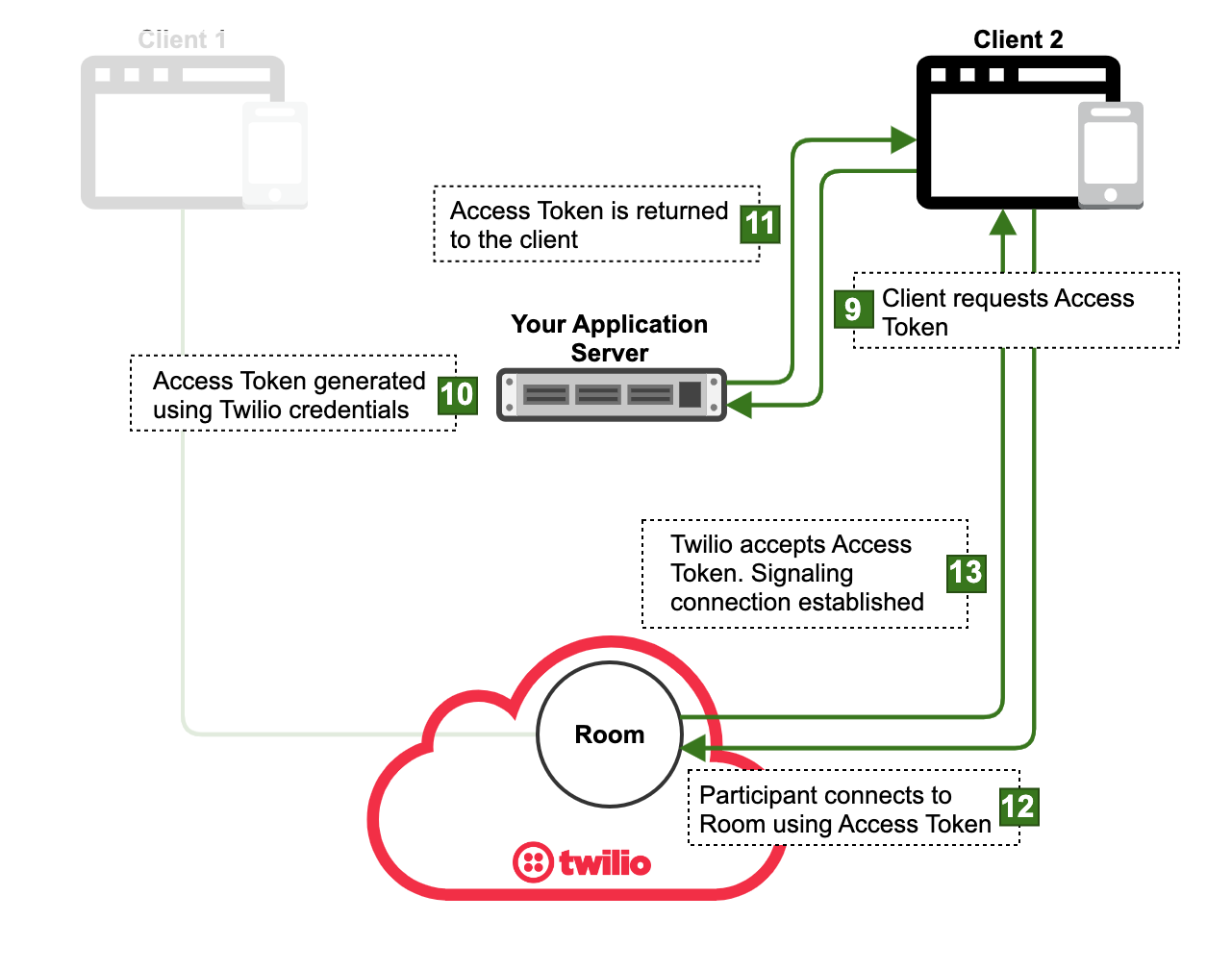


Figure Twilio architecture

1. Figure 1 depicts the high-level design for video calls.

Firebase Database Structure

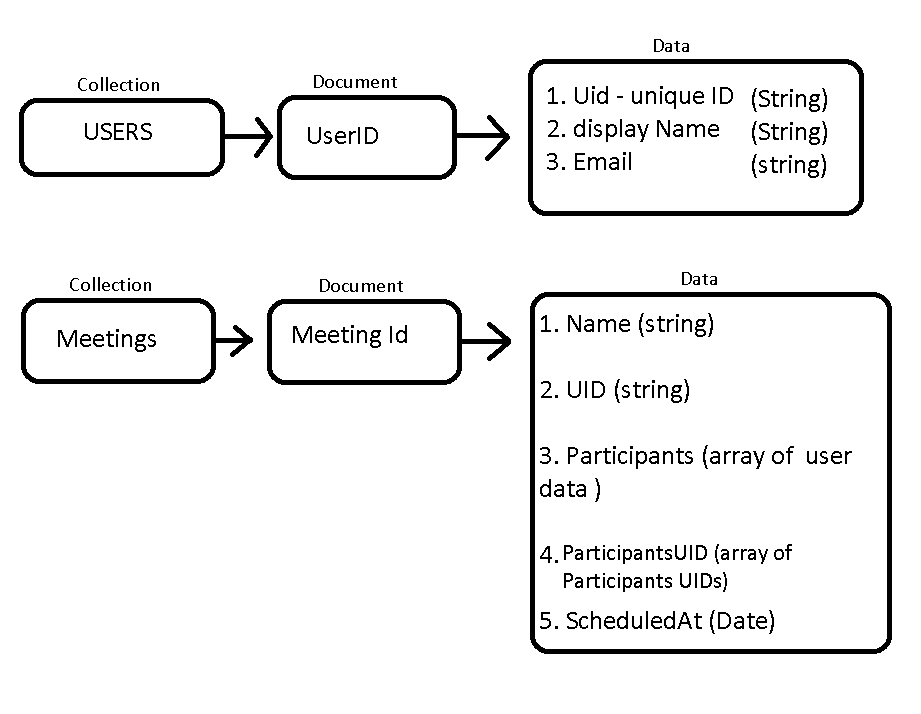
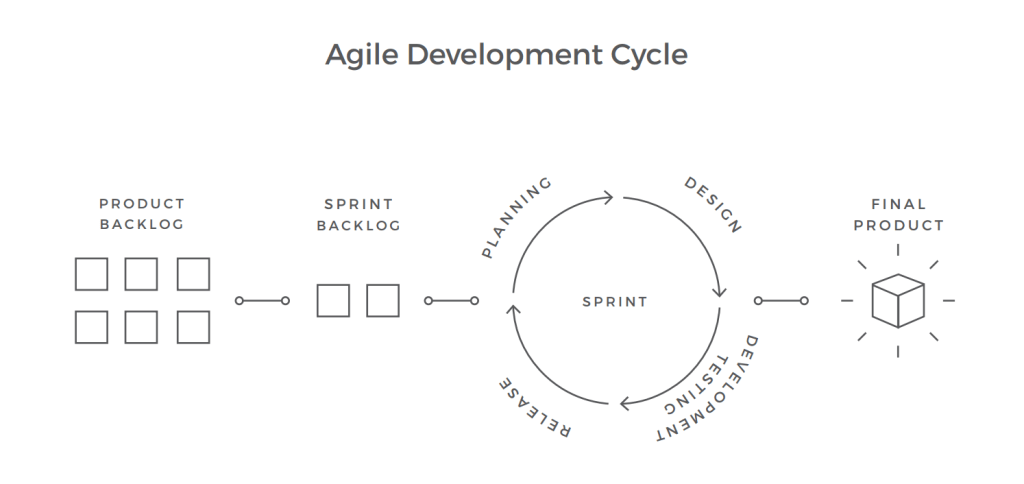


Figure Firebase Database

1. Users -> collection of users, this database consists of a collection of user data.
   1. UID – Unique ID that refers to a user.
   2. Display name – The name that will be displayed.
   3. Email – The Registered email of the user.
2. Meetings -> Collection of meetings, this database consists collection of the meetings data
   1. Name – the title of meeting.
   2. UID- the unique ID of meeting.
   3. Participants – this is an array of user data (name, UID, Email).
   4. participantsUID – array of UIDs of participants that are part of the meeting.

Agile Process documentation



The agile approach is based on teamwork, close collaboration with customers and stakeholders, flexibility, and ability to quickly respond to changes. The basic building blocks of agile development are iterations; each one of them includes planning, analysis, design, development, and testing. The agile method doesn’t require comprehensive documentation at the beginning.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **Marketing Project** |  | WEEK 1 | | | WEEK 2 | | | WEEK 3 | | | WEEK 4 | | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **Task** |  | Compare and select technologies | | | Implement video call | | | add participant support up to 20 people | | | Add user authentication | | |  |
|  |  | Read documentation of the technologies | | | implement the call UI | | | refactor and comment the code | | | add the meetings log and chat functionality | | |  |
|  |  | design UI and database | | | breakdown application into smaller components | | | solve the UI issue for more than 4 participants | | | add ability to schedule call | | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **Outcome** |  | Selected tech - react, NodeJS, firebase and Twilio | | | Implemented two people video call | | | added support for up to 20 participants | | | added user authentication | | |  |
|  |  | selected SFU architecture | | | created the call UI | | | refactored the code | | | added meeting logs and chat functionality | | |  |
|  |  | designed UI and database setup | | | broke down the application into components | | | made UI responsive | | | added schedule meeting functionality | | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **Backlogs** |  | NIL | | | Refactoring of code | | | enhance the UI | | | code commenting | | |  |
|  |  | commenting the code | | | refactor new code | | | documentation | | |  |
|  |  | Make UI responsive | | |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **Testing** |  | NIL | | | tested two people video call, result - irresponsive UI for more than 4 participants | | | tested video call with 12 people, result - smooth call | | | tested all functionality, result - solved all bugs | | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Agile RoadMap