# Design Document & Testing guide

13-07-2021

1.)Apurva Sharma
VIT
Vellore
BTech CSE
apurva.sharma866@gmail.com

2.)Kapil Meghwal
IIT Roorkee B.Tech ECE
kapil.731@iitr.ac.in

Video conferencing solutions with Unite, using peer-to-peer connections.

## How to test UNITE - Video conferencing app?

### Testing web app (points to note)

1. Allow permissions for camera and mic when joining the video chat
2. In case any **user is not broadcasted** it is probably due to server overload, **REFRESH** the window to solve this.
3. Make sure the URL is starting with https

4. While **scheduling a meet** make sure the start and end date follow a logical sequence or else it'll show an **error**.
5. While testing the **Posture bot,** allow permissions for the camera and allow **notifications,** and **REFRESH** the page for changes to take effect.
6. Wait for the model to analyze, and check for notifications

## Testing locally:

1. **`git clone https://github.com/Apurva-tech/unite.git`**
2. **`cd ./unite`**
3. Install node dependencies

   - **`npm install`**

4. Replace firebase API keys with your configurations from your firebase console
5. Create a `.env` file

   - Add relevant credentials

   - **`cp .env.example .env`** which looks something like this:

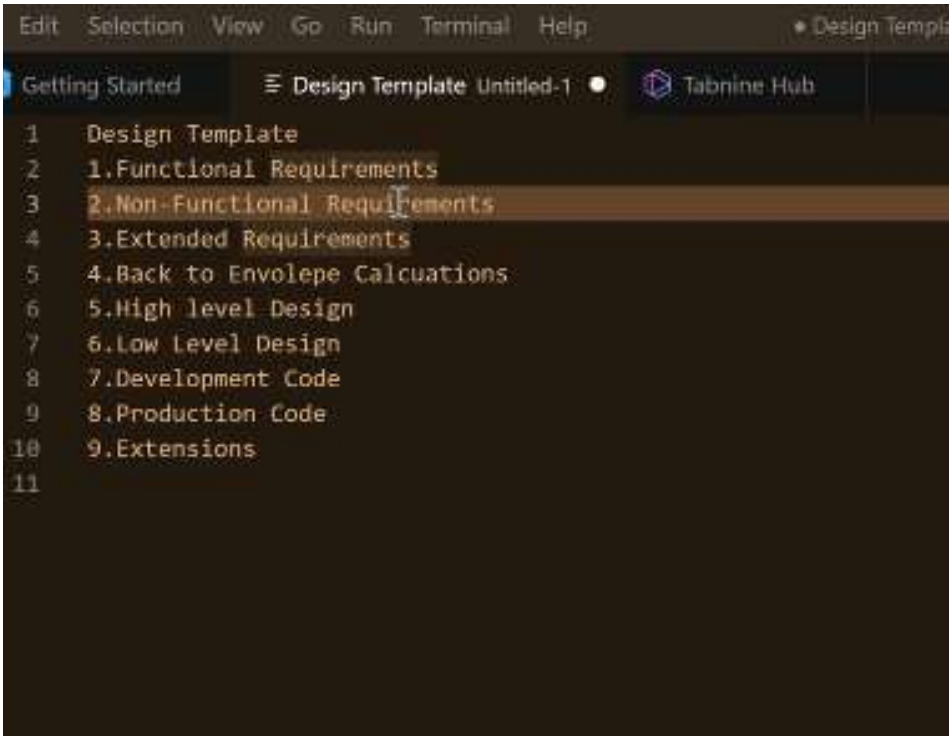
```
.env.example

MONGODB_URI="<mongoURI>"
GAPI_CLIENT_ID="<google-api-client-id>"
GAPI_CLIENT_SECRET="<google-api-secret>"
GAPI_REFRESH_TOKEN="<google-api-refresh-token>"
EMAIL="<nodemailer-email>"        You, seconds ago • U
PASS="<nodemailer-pass>"
```

6. **`npm run dev`**
7. The app is now running at [http://localhost:3030/landing](http://localhost:3030/landing)

# Design Document

## Sprint 1 - Design (June 14 - June 21)

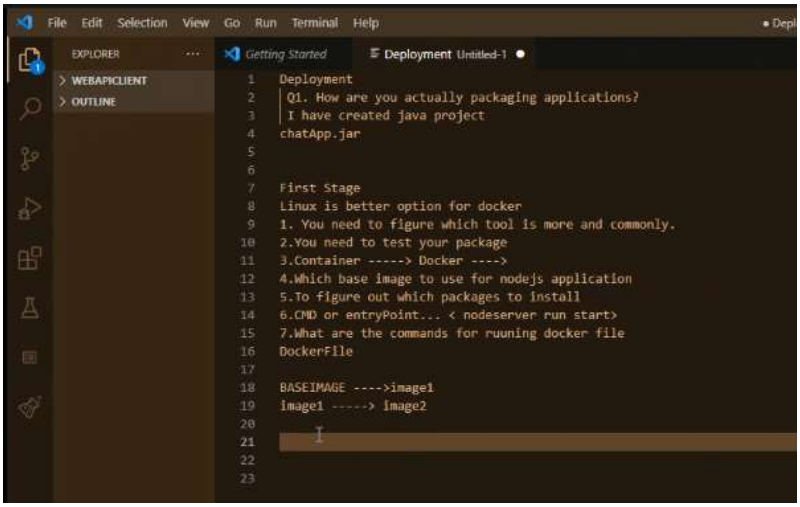
1. Researched about WebRTC, and PeerJS and why is NodeJS the best suit for me!

2. The first prototype was a simple application with a connection between two people and had a text chat feature.
3. Learned how to use peerJS and creating a unique ID to connect two people.
4. In the 1st mentor session, I learned about various design requirements and how to make the code modular. And also, how to integrate features in an application in an industrial environment.
5. Learned how to create a design of an application following **Agile** methodology.
6. 

7. After thorough research I fixated on the following tech stack:

## Sprint 2 - Build (June 21 - July 05)

1. My build sprint started with **Requirement Specification**
2. These were the **Functional requirements** for the application:
    - Video chat b/w multiple users
    - Text chat during the meet
    - Google-based auth using firebase
    - Google Calendar API to schedule meeting
    - User-friendly interface
    - Create an interface for user feedbacks
    - Posture bot for users to get notified when sitting in a bad posture
3. These were the **Non-Functional requirements** for the application:
    - The application be able to accommodate at least 1000 requests per minute
    - The response time of the app should not exceed 30ms
    - The firebase auth should be secure
    - The application should schedule a meet and send a response within 20ms
4. The second mentor session was about Architecture and interface design and how does deployment take place!

    - 


## Sprint 3 - Adopt (June 05 - June 09)

1. Built the video call feature on top of a chat feature

2. Using UUID to create rooms and add video call and chat options from a common room
3. Persist chat with MongoDB and render recent meetings to the home page.

## Sprint 3 - Adopt (June 05 - June 09)

1. Deployed the application on Heroku
2. Manual testing
3. Stress testing the application
4. The last sprint was dedicated to performing a few UI fixes and fixing a few bugs.