Database Management System Project

MEDIAVERSE

- Documentation

Dhev Sabarish S 2020103518

Logeshwaran S 2020103538

Velmurugan J 2020103585

“A non-addictive social media app”

ABSTRACT

Most of the social media apps we use today are overwhelming and addictive which affects students' mental health. Nowadays free social media apps are not actually free, they use our data as their prime revenue to generate personalized ads which many people won't like as they feel insecure. And all the features are not available in a single social media. We have to use multiple social media for multiple purposes.

And here we are with our app Mediaverse which solves the above-mentioned problems. In this app, users can post images, share their thoughts, send or receive messages and many more just like any social media. But the difference is we are planning to provide a disappearing post/thoughts feature that helps users to avoid doom scrolling. We are also planning to include some of the unique features of most of the famous apps. This app doesn't use user data to generate personalized ads.

One of the most useful features for students and developers is discussion platforms like StackOverFlow, StackExchange, Quora, Brainly, etc. are not much focused on the famous social media apps that we are using today. We are here to solve this problem with our Discussion Forum where users can post their doubts and other users can help them with the answers.

The most liked feature in Snapchat is SnapMap which is not available in most of the famous apps. We are planning to build a modified version of that feature that helps users to track their followers and enables them to know the location of the users around them. These are some of the features of our app but we also planning to include more features. From this project, we hope to build a better, non-addictive, secured, and all-in-one social media app.

INTRODUCTION

Features offered

- Disappearing posts
- Trending posts based on Trending Hashtags
- Mapverse – Users Map
- Discussion forum
- Trending question based on Trending Topics
- Private chat and Global Chat

Why use our app ?

A Social Media App that is comparable to the ones we use every day, but with extra functionality.Users don't have to switch between social networking apps to get the best of both worlds. In our app, we integrate the most popular features that the majority of people today choose. As a result, it's as if you have everything in one app.

Advantages:

➔ **Disappearing Posts:**

Snapchat is notable for disappearing messages and snaps, which are not focused on by other social media giants. We've included a feature called "disappearing posts," which was inspired by Snapchat. We feel that this functionality will benefit users by deleting spam postings and retaining posts that become really popular and well-liked.

➔ **MapVerse:**

Live location tracking is one of the most popular features in social media apps nowadays. Snapchat popularised this feature, and many people enjoy using it. We implemented a modified version of that feature, which tracks users' location every time they open the app and displays their current location as well as other users nearby. This allows the user to be aware of others in his or her immediate vicinity.

➔ **Discussion Forum:**

People are always on the lookout for solutions. Many individuals appreciate websites like stackOverFlow, Quora, and others because of this. On these websites, users can ask and answer questions. This is extremely beneficial to students, programmers, and others. While working on a project, a developer, for example, may discover several defects and flaws. He

may debug for hours or even days. Rather than wasting time attempting to troubleshoot the problem on our own, we may go to stackoverflow and learn the specific procedures to resolve the issue. Stackoverflow is well-known among developers because of this.

We offer a "discussion forum" modelled after stackoverflow, which allows users to ask and answer questions. We'll customise the most popular debate topics for users' convenience.Similar queries are grouped together and made easily accessible to users. The user can also conduct a topic search and get right in.

➔ **Global Chat:**

NOT EVERYONE appreciates chatting with complete strangers from all over the world. Conversely, extroverts like it! This removes all geographical barriers, allowing people to converse regardless of where they are. We included 'Global Chat,' which is mostly utilised in games and isn't targeted by social media behemoths. This does not rule out the possibility of private chat! To provide users the best of both worlds, we have both.

➔ **Trending posts:**

Twitter became popular because it is always up to date with the most recent news and posts. This feature is one of the reasons why people use Twitter. We've taken inspiration from Twitter and created trending posts, which will show you what's trending and what's being liked by a lot of people.

➔ **No ad policy:**

Advertisements ! Advertisements ! Advertisements are despised by the people! If we are offered a free application, we are their products. Our data is a source of income for them. This makes a lot of individuals feel insecure. we are here to provide a solution. We have a "no ad policy" in place. We keep your information safe. There are no advertising in our app. We respect everyone's right to privacy. and that is a promise! We're planning to maintain it indefinitely.

TECH STACK USED

“MERN Stack”

➔ Frontend

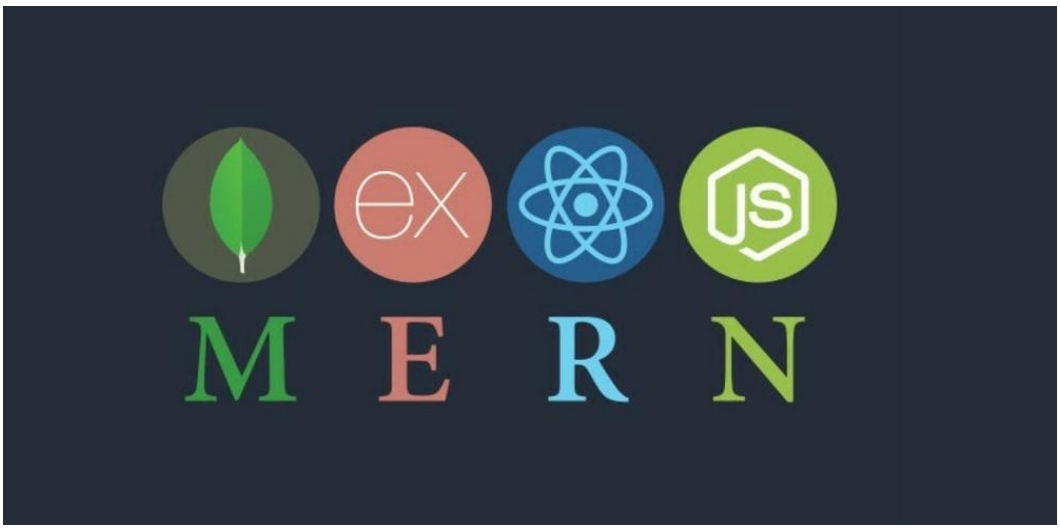
- React Native – Frontend Native JS library
- Expo - Framework to build Native React apps for Android
- Axios - making API calls.
- App will be published in Expo GO

➔ Backend

- Node JS – Runtime environment for JavaScript
- Express JS – Creating Resource based REST API’s
- Babel JS for transpiling JavaScript
- API will be deployed in HEROKU

Database used in Backend :

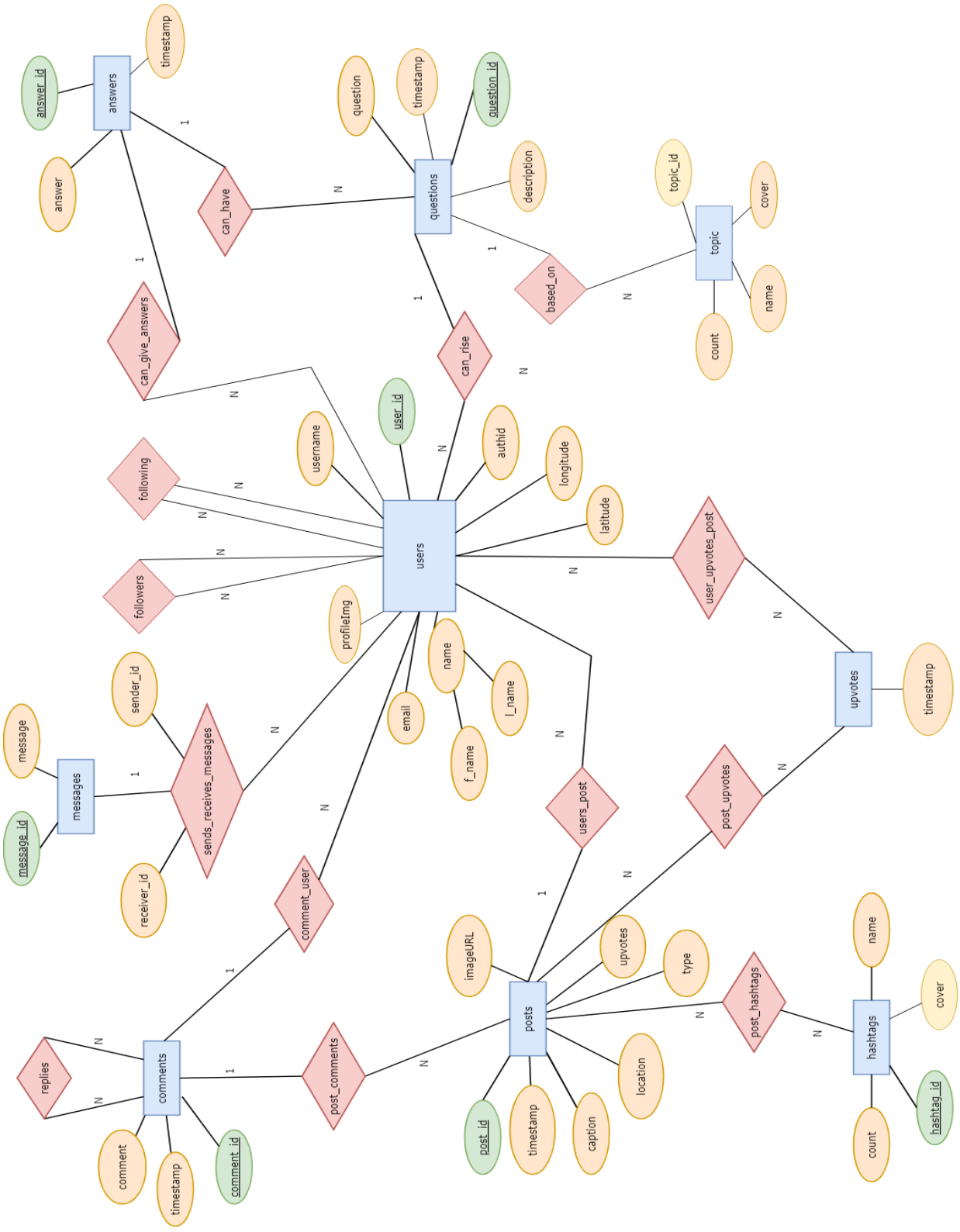
- MongoDB – NoSQL database (cloud)
- Mongoose – provides abstraction over MongoDB
- MongoDB cloud is used .
- Firebase Authentication
- Firestore database for chatting



IMPLEMENTATION:

BACKEND OF THE APP

Entity-Relationship diagram:



Database description:

Since MongoDB is a NOSQL database we don't table to store the data. It's a document based database and it is capable of processing structured, semi-structured and unstructured data. So we need to define specific schemas for each collections.

In MongoDB,

- Set of Entities are called as **Collections**
- Each Entities are called as **Documents**.
- Documents can have attributes as key value pairs.

1) User schema:

```
const userSchema = mongoose.Schema(
   {
      authid: { type: String, required: true }, // comes from
firebase
      fname: String,
      lname: String,
      username: { type: String, required: true },
      email: String,
      latitude: Number,
      longitude: Number,
      profileImg: String,
      upvotedPosts: [{ type: mongoose.Types.ObjectId, ref:
'Posts' }],
      following: [{ type: mongoose.Types.ObjectId, ref: 'Users'
}],
      followers: [{ type: mongoose.Types.ObjectId, ref: 'Users'
}],
   },
   {
      timestamps: true,
   }
);
```

2) Post schema:

```
const postSchema = mongoose.Schema(
   {
      userid: { type: mongoose.Types.ObjectId, ref: 'Users',
required: true },
      type: { type: String, enum: ['text', 'image'], default:
'text' },
      upvotes: { type: Number, default: 0 },
```

```
      comments: [{ type: mongoose.Types.ObjectId, ref: 'Comments'
}],
      caption: String,
      hashtags: [{ type: mongoose.Types.ObjectId, ref: 'Hashtags'
}],
      imageURL: String,
      location: String,
   },
   {
      timestamps: true,
   }
);
```

3) Hashtag schema:

```
const HashtagSchema = mongoose.Schema({
   name: String,
   count: Number,
   cover: String,
});
```

4) Comment schema:

```
const commentSchema = mongoose.Schema(
   {
      userid: { type: mongoose.Types.ObjectId, ref: 'Users' },
      comment: String,
      replies: [{ type: mongoose.Types.ObjectId, ref: 'Comments'
}],
   },
   {
      timestamps: true,
   }
);
```

5) Upvote schema:

```
const UpvoteSchema = mongoose.Schema(
   {
      userid: { type: mongoose.Types.ObjectId, ref: 'Users',
required: true },
      postid: { type: mongoose.Types.ObjectId, ref: 'Posts',
required: true },
```

```
    },
    {
        timestamps: true,
    }
);
```

6) Question Schema:

```
const questionSchema = mongoose.Schema({
    userid: { type: mongoose.Types.ObjectId, ref: 'Users' },
    topic: { type: mongoose.Types.ObjectId, ref: 'Topics' },
    question: String,
    description: String,
    answers: [{ type: mongoose.Types.ObjectId, ref: 'Answers' }],
}, {
    timestamps: true,
});
```

7) Answer Schema:

```
const answerSchema = mongoose.Schema(
    {
        userid: { type: mongoose.Types.ObjectId, ref: 'Users' },
        answer: String,
    },
    {
        timestamps: true,
    }
);
```

8) Topic Schema:

```
const topicSchema = mongoose.Schema({
    name: String,
    count: { type: Number, default: 0 },
    cover: String,
});
```

Rest API routes:

➔ Users related routes

Get user details with authid	GET /users/signin
Get user detais with userid	GET /users/user/:userid
Create new user	POST /users/signup
Edit users details	PATCH /users/
Delete user	DELETE /users
Search for a user (with username)	GET /users/search?searchString='something'
Get all users	GET /users
Follow / Unfollow an user	GET /users/follow/:followingid
Remove follower	DELETE users/follower/:followerUserId
Get all followers of an user	GET /users/followers/
Get all followings of an user	GET /users/following/

➔ Posts related routes

Get all posts of an user	GET /posts/
Get all posts of an user	GET /posts/user/:userid
Get users feed	GET /posts/feed
Get post by Postid	GET /posts/post/:postid
Get trending posts (based on trending Hashtags)	GET /posts/trending
Create new post	POST /posts/
Update post by Postid	PATCH /posts/post/:postid
Get all followings of an user	GET /users/following/
Upvote post	PATCH /posts/upvote/:postid
Delete post	DELETE /posts/post/:postid

➔ Comment related routes:

Get comments of a post	GET /comments/:postid
Create comment on post	POST /comments/post/:postid
Create a reply on comment	POST /comments/:commentid
Delete a comment	DELETE /comments/:postid/:commentid

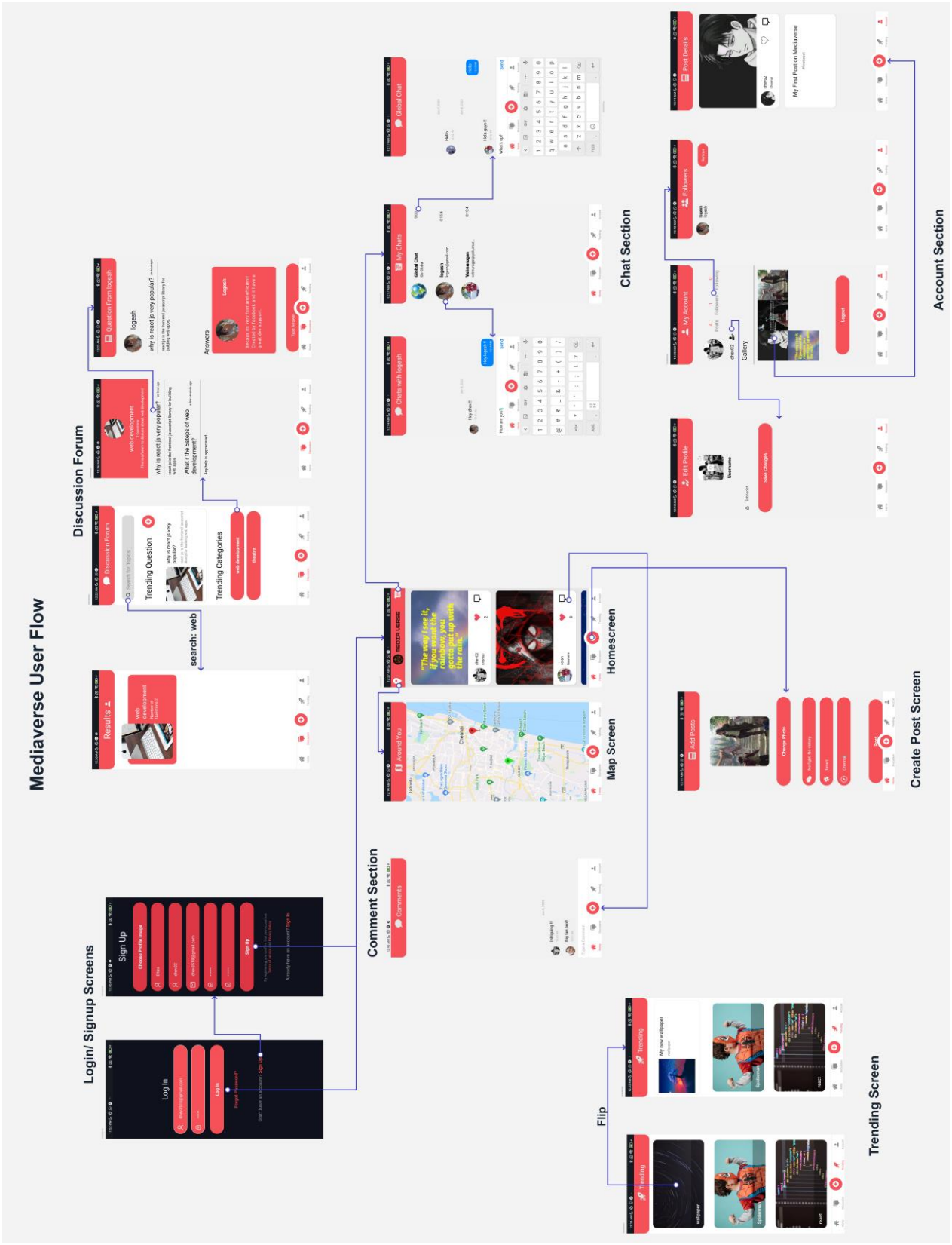
➔ Discussion related routes:

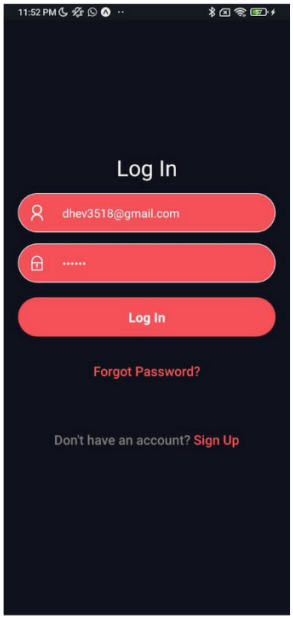
Get all the Questions based on a Topic	GET /discussions/questions/topic/:topicid
Get question by QuestionId	GET /discussions/questions/:questionid
Get trending questions	GET /discussions/questions/trending
Get trending topics	GET /discussions/topic/trending
Search for a topic	GET /discussions/topic/search?keyword='something'
Post a new question	POST /discussions/questions
Update post by Postid	PATCH /posts/post/:postid
Post a new answer	POST /discussions/questions/answer/:questionid

API is deployed live on HEROKU , you can access it using below link

https://mediaverse-backend.herokuapp.com

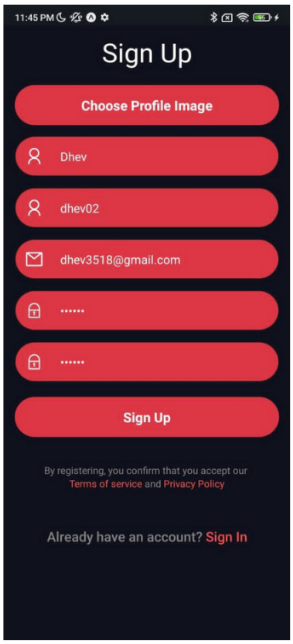
FRONTEND OF THE APP





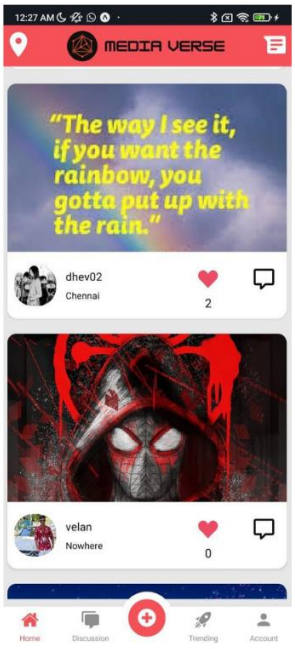
Login Screen

The login screen asks for an email address and a password, then checks to see if the user already exists in the database (Powered by Firebase). The user can now navigate to the home page if he/she exists. It stays on the same page if it doesn't exist.



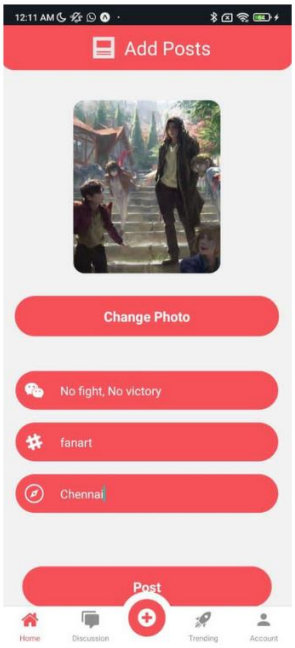
Sign Up Screen

The Sign up screen allows the user to provide basic information in order to create a new account, which links to the home page once completed. If the input does not satisfy any database constraints, it stays on the same page .



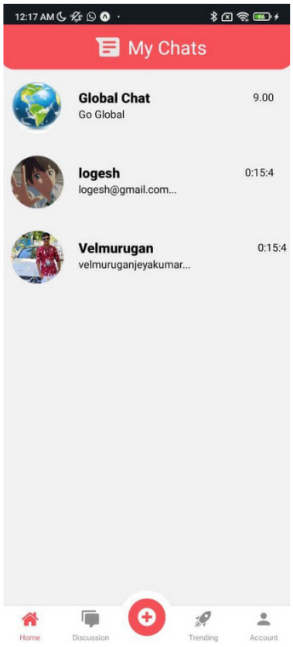
Home Screen

On the home screen, the posts of the mediaverse users are displayed. The post can be liked and a comment can be left on it. He can also see all of the comments and details that have been added to the post.



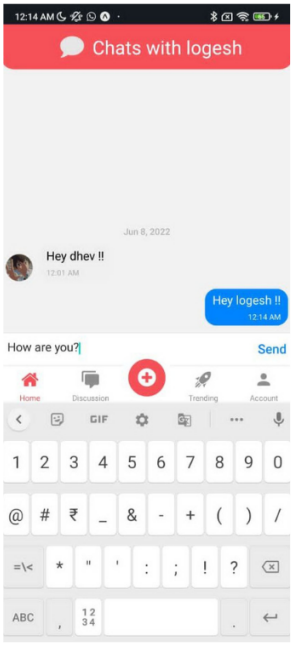
Add Post Screen

To create a post, the add post screen looks for an image component, a caption, and a hashtag. If successful, the post is published and can be accessed by anybody who has access to the mediaverse. If he is unsuccessful, he will remain on the same page with corresponding errors displayed.



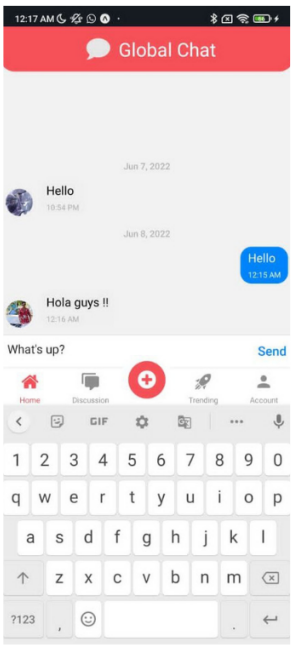
All Chat Screen

The chat screen displays a list of mediaverse users who are friends with the current user. The present user can either go to global chat and communicate with everyone in the mediaverse, or he can go to private chat and chat with only his friends. The firebase firestore database stores all chat messages.



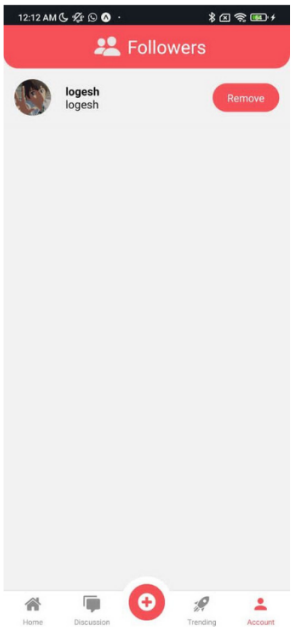
Private Chat Screen

The user can speak privately with his friends using the private chat interface. All chats are saved in a firestore with timestamps and retrieved when the user enters the private chat.



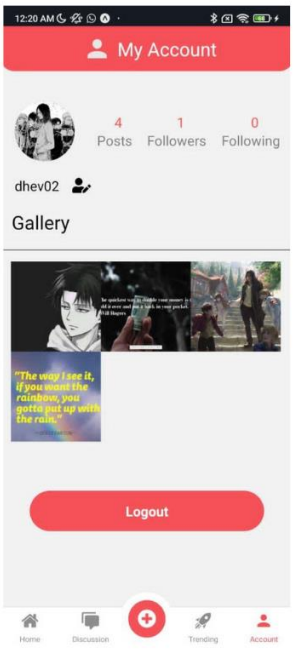
Global Chat Screen

global chat screen allows users to communicate with all mediaverse users.



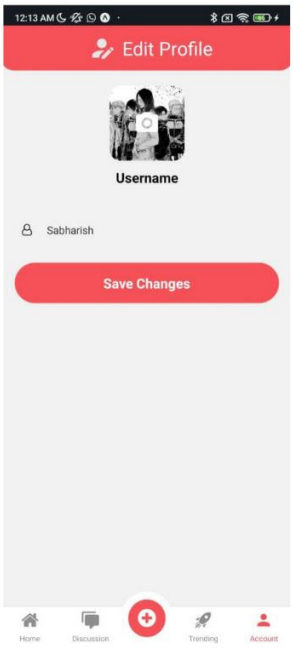
Followers or Following Screen

Displays all followers or Following of the current User.



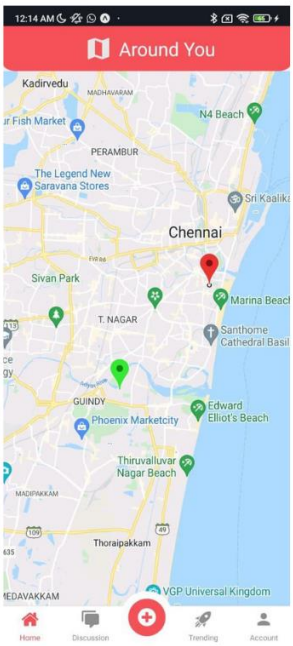
Account Screen

The account screen displays all relevant information about the current user, such as the number of followers, followers, posts, and so on. the user can view all post he posted in mediaverse.



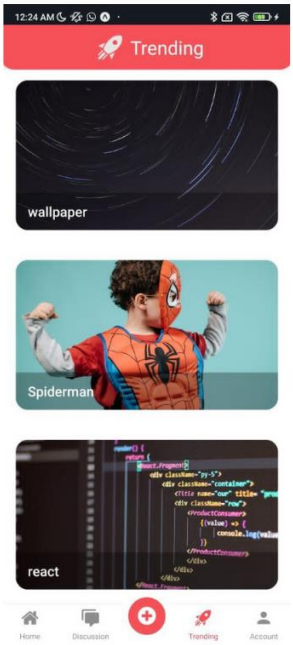
Edit Profile Screen

Users can change their profile picture, username on the edit profile screen.



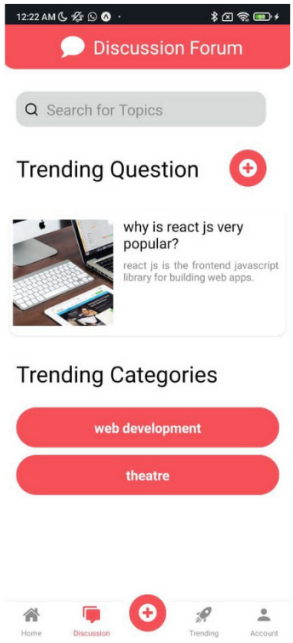
Map Screen

The map screen takes the user's location as input and displays his current position as well as all of his friends' locations. When the user's location changes, the location is dynamic and varies over time. This allows the present user to be aware of all of his friends' whereabouts.



Trending Screen

The trending screen shows posts that have been popular in the last several days, as well as posts that have been trending with hashtags.

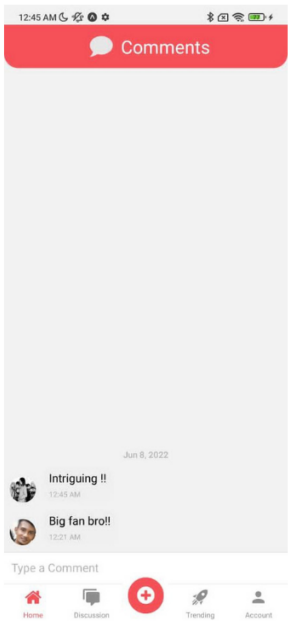


Discussion Forum Screen

The discussion forum screen is similar to quora and stackoverflow sites in that it allows users to pose questions that can be addressed by anybody in the mediaverse. For the convenience of the users, the hot topics and questions based on categories are grouped together and displayed on the discussion forum screen.

Forum Screen

Group of similar questions are categorized together into a single topic for users easy access.



Comments Screen

Users can leave comments on the post, which are available to everyone.

FUTURE SCOPE:

According to the most recent figures, internet usage by the global population has expanded dramatically, with 40% of the world's population currently utilising the internet. This equates to roughly 3.42 billion users. Even though our software does not generate revenue from advertisements, we will generate revenue from users through premium features. We intend to incorporate VR (Virtual Reality) posts, which will allow users to share VR content. AR and Face Filters, Artificial Intelligence (AI) and Chatbots, Multi-Language and SupportVirtual Reality (VR) Shift are among our upcoming features.

TEAM CONTRIBUTION:

• VELMURUGAN J	Backend of the app
• LOGESHWARAN • DHEV SABARISH	Frontend of the app

REFERENCES:

React Native: https://reactnative.dev/docs/getting-started/

Expo: https://docs.expo.dev/

NodeJS: https://nodejs.org/en/docs/

ExpressJS: https://expressjs.com/en/4x/api.html

MongoDB: https://www.mongodb.com/docs/

Mongoose: https://mongoosejs.com/docs/