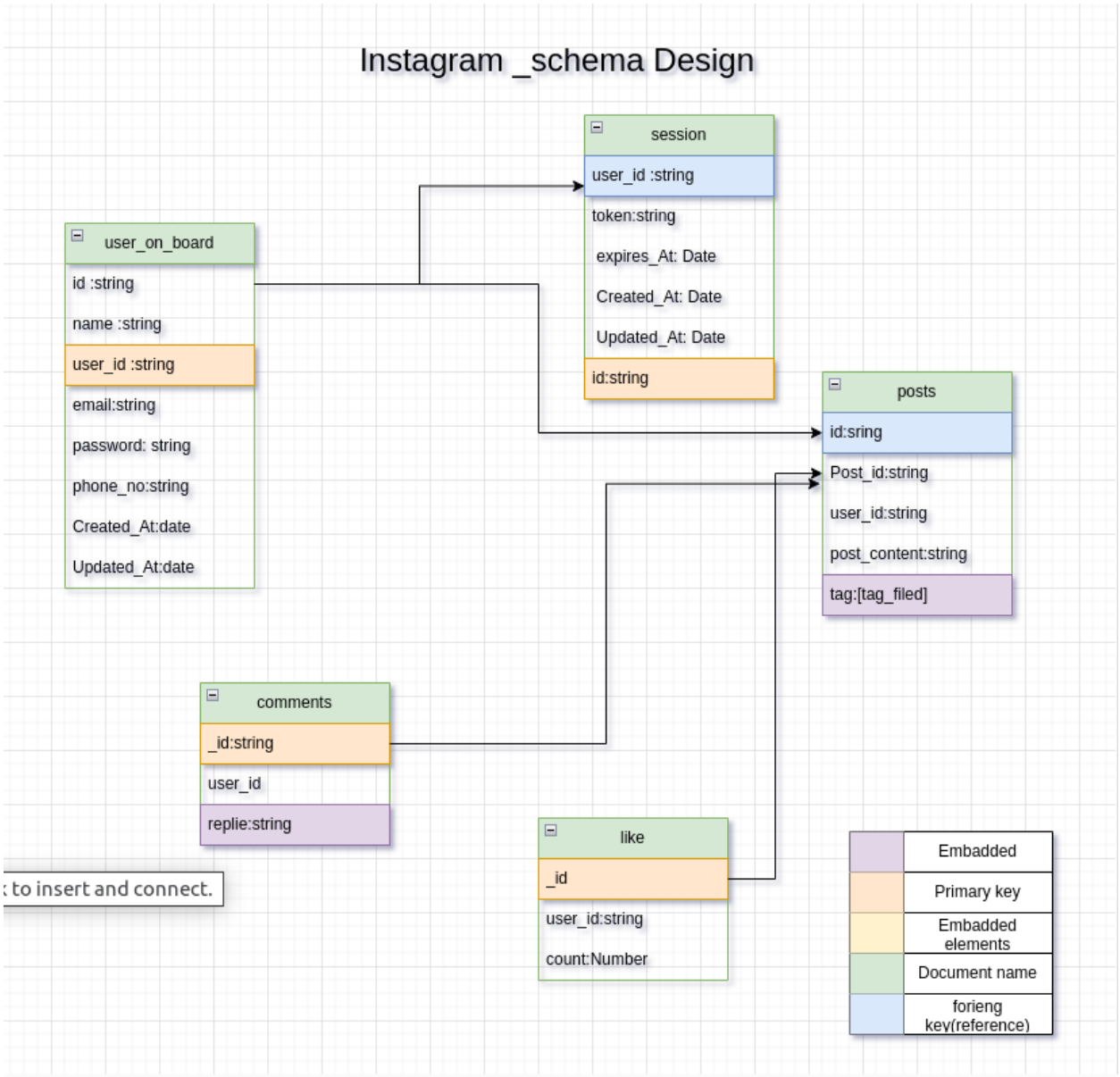
Instagram _schema Design
session
user_id :string
token:string
expires_At: Date
Created_At: Date
Updated_At: Date
id:string
user_on_board
id :string
name :string
user_id :string
email:string
password: string
phone_no:string
Created_At:date
Updated_At:date
posts
id:sring
Post_id:string
user_id:string
post_content:string
tag:[tag_filed]
comments
_id:string
user_id
replie:string
like
_id
user_id:string
count:Number
to insert and connect.
Embadded
Primary key
Embadded elements
Document name
forieng key(reference)

database > TS dbconnection.ts > ...

```ts
import mongoose from 'mongoose';
const url = "mongodb://localhost:27017/instagramdb";



export const connection = async()=>{
    try{
        await mongoose.connect(url)
        console.log("Succesfully connected to the db");

    }catch(e){
        console.log(e,"ERRRRRR");
    }

}
```

Database > models > TS user.ts > default

```ts
import mongoose, { Document, Schema } from 'mongoose';

interface User extends Document {
  _id: string;
  name: string;
  email: string;
  user_id:string;
  password: string;
  phone_no:string;
  created_At:Date;
  updated_At:Date;

}

const userSchema: Schema<User> = new Schema<User>({
  _id: { type: String, required: true },
  name: { type: String, required: true },
  email: { type: String, required: true },
  user_id: { type: String, required: true },
  password: { type: String, required: true },
  phone_no: { type: String, required: true },
  created_At: { type: Date, default: Date.now },
  updated_At: { type: Date, default: Date.now },

});

const UserModel = mongoose.model<User>('User', userSchema);

export default UserModel;
```

database > models > posts.ts > PostSchema > user_id

```ts
import mongoose, { Schema } from "mongoose";
import User from "./user";

interface Post {
  _id: string;
  user_id: mongoose.Types.ObjectId;
  post_content: string;
  post_tag: object;
  created_At: Date;
  updated_At: Date;
}


const PostSchema: Schema = new Schema({
  _id: {
    type: String,
    required: true
  },
  user_id: {
    type:mongoose.Schema.Types.ObjectId,
    ref: User,
    required: true,
  },
  post_content: { type: String, required: true },
  post_tag: { type: Object, required: true },
  created_At: { type: Date, default: Date.now },
  updated_At: { type: Date, default: Date.now },
});

const Post = mongoose.model("Post", PostSchema);
export default Post;

```

```ts
import mongoose, { Document, Schema } from 'mongoose';
import User from './posts'
interface Session extends Document {
  _id: mongoose.Types.ObjectId;
  token:String;
  user_id:string;
  expire_At:Date;
  created_At:Date;
  updated_At:Date;

}

const SessionSchema: Schema<Session> = new Schema<Session>({
  _id: {
    type:mongoose.Schema.Types.ObjectId,
    ref: User,
    required: true,
  },
  token: { type: String, required: true },
  user_id: { type: String, required: true },
  expire_At: { type: Date, required: true },
  created_At: { type: Date, default: Date.now },
  updated_At: { type: Date, default: Date.now },

});

const SessionModel = mongoose.model<Session>('Session', SessionSchema);

export default SessionModel;
```

```ts
import mongoose, { Document, Schema } from 'mongoose';
import Post from './posts';

interface Comment extends Document {
  _id:mongoose.Types.ObjectId;
  user_id:string;
  replie:object
  created_At:Date;
  updated_At:Date;

}

const Comment: Schema<Comment> = new Schema<Comment>({
  _id: {
    type:mongoose.Schema.Types.ObjectId,
    ref: Post,
    required: true },
    user_id: { type: String, required: true },
  replie: { type: Object, required: true },
  created_At: { type: Date, default: Date.now },
  updated_At: { type: Date, default: Date.now },

});

const CommentModel = mongoose.model<Comment>('Comment', Comment);

export default CommentModel;
```

```ts
import mongoose, { Document, Schema } from 'mongoose';
import Post from './posts';

interface Like extends Document {
  _id: mongoose.Types.ObjectId;
  user_id:string;
  count:Number;
  created_At:Date;
  updated_At:Date;

}

const Like: Schema<Like> = new Schema<Like>({
  _id: {
    type:mongoose.Schema.Types.ObjectId,
    ref: Post,
    required: true
    },
  user_id: { type: String, required: true },
  count: { type: Number, required: true },


  created_At: { type: Date, default: Date.now },
  updated_At: { type: Date, default: Date.now },

});

const LikeModel = mongoose.model<Like>('Like', Like);

export default LikeModel;
```

```
db.posts.aggregate([{
    //this $ match the value of array greater than the 1 ($exists specify the array value exist or not )

    $match: {'post_tag.1':{$exists: true}}
}])
```

Find × | Aggregate × | Aggregate (1) × | **Aggregate (2) ×**

0.049 s | 8 Docs

```
/* 7 createdAt:18/07/2023, 21:54:04*/
{
    "_id" : ObjectId("64b6bca46ec679a9ecae5df3"),
    "user_id" : ObjectId("64b6ba266ec679a9ecae5de9"),
    "post_content" : "hm sath sath oh no ??",
    "post_tag" : [ "hilo ", "hsklo" ]
},

/* 8 createdAt:18/07/2023, 21:54:29*/
{
    "_id" : ObjectId("64b6bcbd6ec679a9ecae5df4"),
    "user_id" : ObjectId("64b6ba266ec679a9ecae5dea"),
    "post_content" : "hm sath sath oh no ??",
    "post_tag" : [ "hilo ", "hsklo" ]
```

```
db.posts.aggregate([{
    //sort -1 dec order / 1 incressing order
      $sort: {post_content:1}
}])
```

posts | 0.051 s | 8 Docs

	Value	T
(1) 64b6bb4d6ec679a9ecae5ded	{ user_id : ObjectId("64b6ba266ec679a9ecae5de3"), post_content : "abcd", post_tag : ["hilo ", "hsklo"] } (4 fields)	D
_id	64b6bb4d6ec679a9ecae5ded	O
user_id	64b6ba266ec679a9ecae5de3	O
post_content	abcd	S
post_tag	Array[2]	A
0	hilo	S
1	hsklo	S
(2) 64b6bbf46ec679a9ecae5def	{ post_content : "hm na sath sath h ??" } (4 fields)	D
(3) 64b6bb9c6ec679a9ecae5dee	{ user_id : ObjectId("64b6ba266ec679a9ecae5de4"), post_content : "hm sath sath", post_tag : ["hilo ", "hsklo"] } (4 fields)	D
(4) 64b6bc286ec679a9ecae5df0	{ post_content : "hm sath sath h / yes" } (4 fields)	D
(5) 64b6bc516ec679a9ecae5df1	{ post_content : "hm sath sath oh no" } (4 fields)	D

```
db.posts.aggregate([{
    //sort -1 dec order / 1 incressing order
      $sort: {post_content:-1}
}])
```

Aggregate (1) × | Aggregate (2) × | Error × | Aggregate (3) × | **Aggregate (4) ×**

posts | 0.053 s | 8 Docs

	Value	Ty
(1) 64b6bc7f6ec679a9ecae5df2	{ post_content : "hm sath sath oh no ??" } (4 fields)	D
_id	64b6bc7f6ec679a9ecae5df2	O
user_id	64b6ba266ec679a9ecae5de8	O
post_content	hm sath sath oh no ??	S
post_tag	Array[2]	A
0	hilo	S
1	hsklo	S
(2) 64b6bca46ec679a9ecae5df3	{ post_content : "hm sath sath oh no ??" } (4 fields)	D
(3) 64b6bcbd6ec679a9ecae5df4	{ post_content : "hm sath sath oh no ??" } (4 fields)	D

```
db.posts.find().explain()
```

Find × | Aggregate × | Aggregate (1) × | Aggregate (2) × | **Result (1) ×**

79 s


```
            "stage" : "COLLSCAN",
            "direction" : "forward"
        },
        "rejectedPlans" : [ ]
    },
    "command" : {
        "find" : "posts",
        "filter" : {

        },
        "batchSize" : 1000,
        "projection" : {

        },
        "$readPreference" : {
            "mode" : "primary"
        },
        "$db" : "instagramdb"
    },
    "serverInfo" : {
        "host" : "user-HP-EliteBook-840-G6",
        "port" : 27017,
        "version" : "6.0.8",
        "gitVersion" : "3d84c0dd4e5d99be0d69003652313e7eaf4cdd74"
    },
    "serverParameters" : {
        "internalQueryFacetBufferSizeBytes" : 104857600,
        "internalQueryFacetMaxOutputDocSizeBytes" : 104857600,
        "internalLookupStageIntermediateDocumentMaxSizeBytes" : 104857600,
        "internalDocumentSourceGroupMaxMemoryBytes" : 104857600,
        "internalQueryMaxBlockingSortMemoryUsageBytes" : 104857600,
        "internalQueryProhibitBlockingMergeOnMongoS" : 0,
        "internalQueryMaxAddToSetBytes" : 104857600,
```

```
    "serverInfo" : {
        "host" : "user-HP-EliteBook-840-G6",
        "port" : 27017,
        "version" : "6.0.8",
        "gitVersion" : "3d84c0dd4e5d99be0d69003652313e7eaf4cdd74"
    },
    "serverParameters" : {
        "internalQueryFacetBufferSizeBytes" : 104857600,
        "internalQueryFacetMaxOutputDocSizeBytes" : 104857600,
        "internalLookupStageIntermediateDocumentMaxSizeBytes" : 104857600,
        "internalDocumentSourceGroupMaxMemoryBytes" : 104857600,
        "internalQueryMaxBlockingSortMemoryUsageBytes" : 104857600,
        "internalQueryProhibitBlockingMergeOnMongoS" : 0,
        "internalQueryMaxAddToSetBytes" : 104857600,
        "internalDocumentSourceSetWindowFieldsMaxMemoryBytes" : 104857600
    },
    "ok" : 1
}
```

```
db.posts.aggregate([{
    //this $ match the value of array greater than the 1 ($exists specify the array value exist or not )

    $match: {'post_tag.1':{$exists: true}}
}])
```

Find × | Aggregate × | Aggregate (1) × | **Aggregate (2) ×**

0.049 s | 8 Docs

```
/* 7 createdAt:18/07/2023, 21:54:04*/
{
    "_id" : ObjectId("64b6bca46ec679a9ecae5df3"),
    "user_id" : ObjectId("64b6ba266ec679a9ecae5de9"),
    "post_content" : "hm sath sath oh no ??",
    "post_tag" : [ "hilo ", "hsklo" ]
},

/* 8 createdAt:18/07/2023, 21:54:29*/
{
    "_id" : ObjectId("64b6bcbd6ec679a9ecae5df4"),
    "user_id" : ObjectId("64b6ba266ec679a9ecae5dea"),
    "post_content" : "hm sath sath oh no ??",
    "post_tag" : [ "hilo ", "hsklo" ]
}
```

```
db.posts.aggregate({
    $lookup: {
          from: "users",
          localField: "user_id",
          foreignField: "_id",
          as: "userdata"
         }

})
// db.posts.deleteMany({})
```

gate × | Aggregate (1) × | Result × | Find × | Result (1) × | Find (1) × | **Aggregate (2) ×**

sts | 0.128 s | 2 Docs

```
    "userdata" : [ ]
},

/* 2 createdAt:18/07/2023, 17:04:42*/
{
    "_id" : ObjectId("64b678d26a4f1b2a92d4db76"),
    "user_id" : ObjectId("64b6746a6a4f1b2a92d4db6f"),
    "post_content" : "abcd",
    "post_tag" : [ "hilo ", "hsklo" ],
    "userdata" : [
        {
            "_id" : ObjectId("64b6746a6a4f1b2a92d4db6f"),
            "name" : "amn",
            "email" : "asdf@gh",
            "user_id" : "mna@1223",
            "password" : "1234",
            "phone_no" : "9876543210"
        }
    ]
}
```