

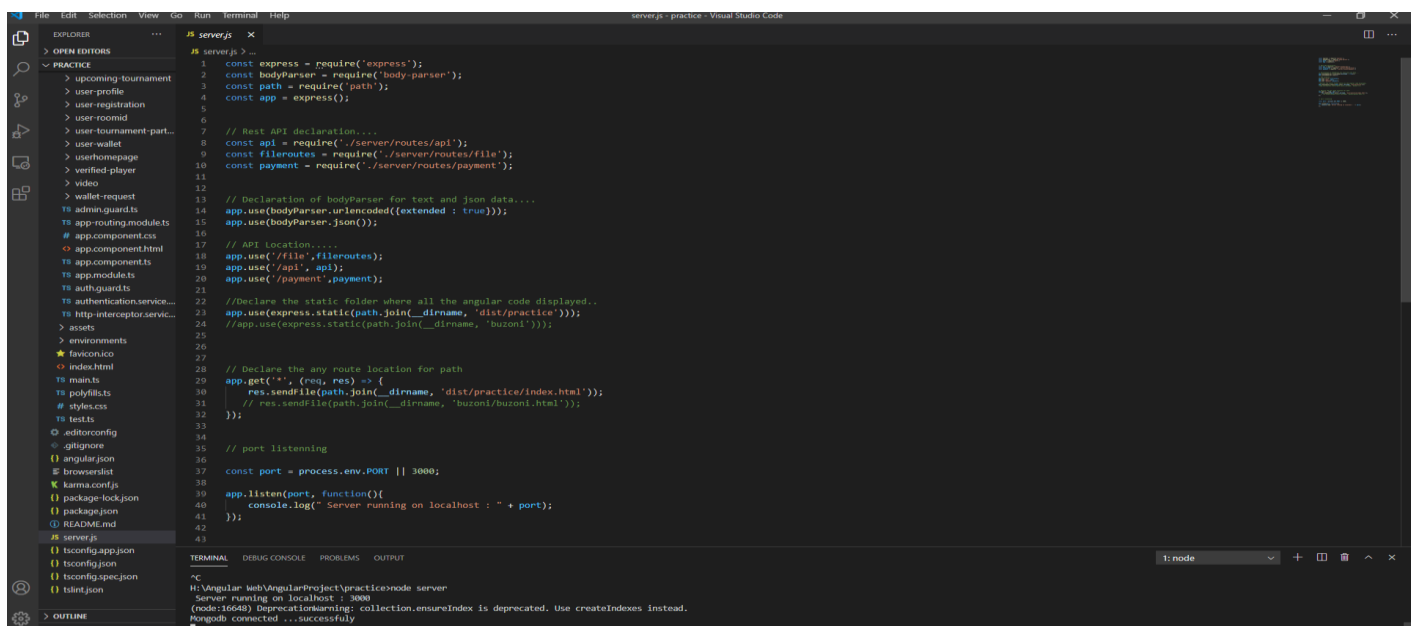
Esports hosting Platform : This project is about hosting an online esports tournament, where players can register for the upcoming tournament and get notified when the tournament starts. Players will receive the results at the end of the tournament with the ranking and the prizes they won.

This project has a payment gateway for paying the participation fees and sending a request for withdrawing winning prize money. Also, players can use wallet money as a participation fee for the next tournament.

Tools and Technologies Used:

- Visual Studio Code
- MongoDB
- Node.js
- Express.js
- Angular 10

Visual Studio Code: Visual Studio Code is a lightweight but powerful source code editor. It comes with built-in support for JavaScript, TypeScript, and Node.js, and a good thing is built-in terminal that is useful to Node.js developers.



The screenshot shows the Visual Studio Code interface with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The file explorer shows a project structure for 'server.js - practice'. The code editor displays the contents of 'server.js', which is a Node.js Express application. The terminal shows the command 'node' being executed, and the output indicates that the server is running on localhost:3000 and MongoDB is connected successfully.

```
server.js
1 const express = require('express');
2 const bodyParser = require('body-parser');
3 const path = require('path');
4 const app = express();
5
6 // Rest API declaration...
7
8 const api = require('./server/routes/api');
9 const fileroutes = require('./server/routes/file');
10 const payment = require('./server/routes/payment');
11
12 // Declaration of bodyParser for text and json data...
13 app.use(bodyParser.urlencoded({extended: true}));
14 app.use(bodyParser.json());
15
16 // API location...
17 app.use('/file', fileroutes);
18 app.use('/api', api);
19 app.use('/payment', payment);
20
21 //Declare the static folder where all the angular code displayed..
22 app.use(express.static(path.join(__dirname, 'dist/practice')));
23 //app.use(express.static(path.join(__dirname, 'buzoni')));
24
25 // Declare the any route location for path
26 app.get('/', (req, res) => {
27   res.sendFile(path.join(__dirname, 'dist/practice/index.html'));
28   // res.sendFile(path.join(__dirname, 'buzoni/buzoni.html'));
29 });
30
31 // port listening
32 const port = process.env.PORT || 3000;
33
34 app.listen(port, function(){
35   console.log(" Server running on localhost : " + port);
36 });
37
38
39
40
41
42
43
```

TERMINAL

```
node
H:\Angular Web\AngularProject\practice\node server
Server running on localhost : 3000
(node:1068) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
MongoDB connected ...successfully
```

MongoDB: MongoDB is a document-oriented NoSQL database used for high volume data storage. Instead of using tables and rows as in the traditional relational databases. MongoDB makes use of collections and documents. Documents consist of key-value pairs which are the basic unit of data in MongoDB. Collections contains sets of documents and function which is the equivalent of relational database tables.

Basically I used MongoDB for store players data and perform some query like as delete data, find data, update data etc.

First start MongoDB In local Machine Using mongod command

```
Microsoft Windows [Version 10.0.19041.746]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\rj\>mongod
2021-01-27T13:12:48.665+0530 I CONTROL [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'
2021-01-27T13:12:41.872+0530 W ASIO [main] No TransportLayer configured during NetworkInterface startup
2021-01-27T13:12:41.912+0530 I CONTROL [initandlisten] MongoDB starting : pid=2152 port=27017 dbpath=C:\data\db\ 64-bit host=DESKTOP-C01D22A
2021-01-27T13:12:41.912+0530 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2021-01-27T13:12:41.934+0530 I CONTROL [initandlisten] db version v4.2.5
2021-01-27T13:12:41.934+0530 I CONTROL [initandlisten] git version: 2201270b51e13df0aa708ff278f0679c59dc32
2021-01-27T13:12:41.934+0530 I CONTROL [initandlisten] allocator: tcmalloc
2021-01-27T13:12:41.934+0530 I CONTROL [initandlisten] modules: none
2021-01-27T13:12:41.934+0530 I CONTROL [initandlisten] build environment:
2021-01-27T13:12:41.935+0530 I CONTROL [initandlisten] distmod: 2012plus
2021-01-27T13:12:41.935+0530 I CONTROL [initandlisten] distarch: x86_64
2021-01-27T13:12:41.935+0530 I CONTROL [initandlisten] target_arch: x86_64
2021-01-27T13:12:41.958+0530 I STORAGE [initandlisten] Detected data files in C:\data\db\ created by the 'wiredTiger' storage engine, so setting the active storage engine to 'wiredTiger'.
2021-01-27T13:12:42.184+0530 I STORAGE [initandlisten] _configure: create_cache_size=3520M,cache_overflow=(file_max=0M),session_max=33000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enable=true,archive=true,path=journal,compression=snappy),file_managers=(close_idle_files=10000,close_scan_interval=10,close_handle_minimum=250),statistics_log=(wait=0),verbose=[recovery_progress,checkpoint_progress],
2021-01-27T13:12:42.222+0530 I STORAGE [initandlisten] WiredTiger message [161733362:18367][2152:140737064293712]: txn-recover: Recovering log 81 through 82
2021-01-27T13:12:42.313+0530 I STORAGE [initandlisten] WiredTiger message [161733362:358695][2152:140737064293712]: txn-recover: Main recovery loop: starting at 81/14080 to 82/256
2021-01-27T13:12:42.313+0530 I STORAGE [initandlisten] WiredTiger message [161733362:83244][2152:140737064293712]: txn-recover: Recovering log 81 through 82
2021-01-27T13:12:42.917+0530 I STORAGE [initandlisten] WiredTiger message [161733362:917428][2152:140737064293712]: txn-recover: Recovering log 82 through 82
2021-01-27T13:12:42.904+0530 I STORAGE [initandlisten] WiredTiger message [161733362:993435][2152:140737064293712]: txn-recover: Set global recovery timestamp: (0, 0)
2021-01-27T13:12:43.265+0530 I RECOVERY [initandlisten] WiredTiger recoveryTimestamp: Ts: Timestamp(0, 0)
2021-01-27T13:12:43.409+0530 I STORAGE [initandlisten] Timestamp monitor starting
2021-01-27T13:12:43.500+0530 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2021-01-27T13:12:43.501+0530 I CONTROL [initandlisten] ** WARNING: This server is bound to localhost.
2021-01-27T13:12:43.502+0530 I CONTROL [initandlisten] ** Remote systems will be unable to connect to this server.
2021-01-27T13:12:43.504+0530 I CONTROL [initandlisten] ** Start the server with --bind_ip address(es) to specify which IP
2021-01-27T13:12:43.504+0530 I CONTROL [initandlisten] ** addresses it should serve responses from, or with --bind_ip_all to
2021-01-27T13:12:43.506+0530 I CONTROL [initandlisten] ** bind to all interfaces. If this behavior is desired, start the
2021-01-27T13:12:43.506+0530 I CONTROL [initandlisten] ** server with --bind_ip 127.0.0.1 to disable this warning.
2021-01-27T13:12:43.512+0530 I CONTROL [initandlisten]
2021-01-27T13:12:43.512+0530 I SHARDING [initandlisten] Marking collection local.system.replset as collection version: <unsharded>
2021-01-27T13:12:43.512+0530 I STORAGE [initandlisten] Flow Control is enabled on this deployment.
2021-01-27T13:12:43.513+0530 I SHARDING [initandlisten] Marking collection admin.system.roles as collection version: <unsharded>
2021-01-27T13:12:43.513+0530 I SHARDING [initandlisten] Marking collection admin.system.version as collection version: <unsharded>
2021-01-27T13:12:43.514+0530 I SHARDING [initandlisten] Marking collection local.startup_log as collection version: <unsharded>
2021-01-27T13:12:43.516+0530 I SHARDING [initandlisten]
2021-01-27T13:12:43.581+0530 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory 'C:\data\db\diagnostic.data'
2021-01-27T13:12:43.582+0530 I SHARDING [LogicalSessionCacheRefresh] Marking collection config.system.sessions as collection version: <unsharded>
2021-01-27T13:12:43.582+0530 I SHARDING [LogicalSessionCacheRefresh] Marking collection config.transactions as collection version: <unsharded>
2021-01-27T13:12:43.582+0530 I NETWORK [listener] listening on 127.0.0.1
2021-01-27T13:12:43.582+0530 I NETWORK [listener] listening on port 27017
2021-01-27T13:12:46.081+0530 I SHARDING [ftdc] Marking collection local.oplog.rs as collection version: <unsharded>
```

After that create connection using node.js and mongoose

```
//mongodb connection...
const url = "mongodb://localhost:27017/test";
//const url = "mongodb+srv://rajesh:Rajesh@2020@cluster0.sfg3m.mongodb.net/vasiesport?retryWrites=true&w=majority";

mongoose.connect(url, {useUnifiedTopology: true,useNewUrlParser: true }, function(err){
  if(err){
    console.error(err);
  }else{
    console.log("Mongodb connected ...successfully");
  }
});

mongoose.Promise = global.Promise;
```

After that created data schema models that require to my project using mongoose

```

const mongoose = require('mongoose');

//create Schema for tournaments
const Schema = mongoose.Schema;

const tournamentResultSchema = new Schema({
  tournamentId : {
    type : String,
    required : true
  },
  name : {
    type : String ,
    required : true
  },
  email:{
    type : String,
    required : true
  },
  prize : {
    type : Number,
    default : 0
  },
  kill : {
    type : Number,
    required : true
  },
  rank : {
    type : Number,
    required : true
  },
  status : {
    type : Boolean,
    default : false
  }
});

// model contain three parameter.... first one is Model name
// Second one is Schema name
// collection name...
module.exports = mongoose.model('TournamentResult',tournamentResultSchema,'TournamentResults');

```

After that create **API** using **node.js**, **MongoDB**, **express.js**

```

//all registered Users.....
router.get('/users', async function(req, res){
  try{
    const result= await User.find({ });
    res.json(result);
  }catch(err){
    res.status(422).send(['Something Went Wrong Please Try Again Later']);
  }
});

//all registered Users.....
router.get('/userRank', async function(req, res){
  try{
    const result= await User.find().sort({"score":-1});
    res.json(result);
  }catch(err){
    res.status(422).send(['Something Went Wrong Please Try Again Later']);
  }
});

/// authenticated user data api.....
router.get('/user/:id' ,async function(req,res){
  try {
    const data = await User.findOne({_id : req.params.id});
    res.json(data);
  } catch (err) {
    res.status(422).send(['Something Went Wrong Please Try Again Later']);
  }
});

```

Node.js : Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine.

I used node.js for developing Rest-API for my project.

It is very good environment to develop Rest-API

```
const express = require('express');
const bodyParser = require('body-parser');
const path = require('path');
const app = express();

// Rest API declaration....
const api = require('./server/routes/api');
const fileroutes = require('./server/routes/file');
const payment = require('./server/routes/payment');

// Declaration of bodyParser for text and json data....
app.use(bodyParser.urlencoded({extended : true}));
app.use(bodyParser.json());

// API Location.....
app.use('/file',fileroutes);
app.use('/api', api);
app.use('/payment',payment);

//Declare the static folder where all the angular code displayed..
app.use(express.static(path.join(__dirname, 'dist/practice')));
//app.use(express.static(path.join(__dirname, 'buzoni')));

// Declare the any route location for path
app.get('*', (req, res) => {
  res.sendFile(path.join(__dirname, 'dist/practice/index.html'));
  // res.sendFile(path.join(__dirname, 'buzoni/buzoni.html'));
});

// port listenning

const port = process.env.PORT || 3000;

app.listen(port, function(){
  console.log(" Server running on localhost : " + port);
});
```

Express.js: Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.

Angular 10: Angular is an application design framework and development platform for creating efficient and sophisticated single-page apps.

I used angular 10 for design user-interface and angular services for integrate REST-API with frontend