Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

AY: 2024-25

Class:	TE	**Semester:**	
Course Code:		**Course Name:**	WC

Name of Student:	Sainath Khot
Roll No. :	20
Assignment No.:	5
Title of Assignment:	Express js
Date of Submission:	
Date of Correction:	

Evaluation

Performance Indicator	Max. Marks	Marks Obtained
Completeness	5	4
Demonstrated Knowledge	3	2
Legibility	2	2
Total	10	8.

Performance Indicator	Exceed Expectations (EE)	Meet Expectations (ME)	Below Expectations (BE)
Completeness	5	3-4	1-2
Demonstrated Knowledge Legibility	3	2	1
Legibility	2	1	0

Checked by

Name of Faculty : Kshitija Gharat

Signature : Gharat

Date : 18/10/24

WC - 5

Q1 Create an Express.js applicatio with 2 route, one that responds with "This is the home page" when the root URL is accessed and another that responds with "About US" when "/about is accessed.

```
-)  const express = require ('express'),
    const app = express ();
    app get ("/', (req . res) =)res. sed ('This is the
        hompage '));
    app. get ('/about ', (req. res)=) res . send ('About US'));
    app . listen (3000, ) =) console . log ('Server running on
        port 3000 ));
```

Q2 Built an Express application that uses a session managment middleware (like Express . session) to track user session. Create a route that displays the users session data

```
=)  const express = require ('express');
    const session = require (' express_session');
     const app= express ();
    app.use session ( {
              secreter : 'secret key';
              resave : false
              save uninitiazed : true,
              cookie : { secure : false
             }));
      app. get ('/', (req . res) => {
            if (! req . session . view) {
                req . session . view) {
                else {
```

```
    req . session view Id ; }
app . listen ( 3000 , () => console . log ('Server running
    on port 3000'));
```

Q3 Develop an Express application that connects to a MongoDB database. Create routes to handle create Read, Update and Delete (CRUD) operation for a user collection.

=>

```
const express = require ('require');
const mongoose = require ('mongoose');
const app = express ();
mongoose . connect ('mongo : // local host : 27107 /user
DB' , { use New Url Parser : true, use Unified
    topo logy : true });
const user Schema = new mongoose . Schema ({ name :
  string , age : Numbers });
const user = mongoose . model ('User', user Schema);
app . use (express . json ());

// CREATE
app . post ('/user' , (req , res) => {
  const new User = new User (req . body );
  new User . save (). then (() => res . send ('User created')),
  catch (err => res . status (400) . send (err))});

// READ
app . get ('/user' (req , res ) => {
User find () . then (users => res json (user)) . catch
(err => res . status . (400) . send (err)); });
```

```
// UPDATE
app put ('/ usr / id ' (req res) =>{
usr find By ID And Update (req .params id,
   req body)
then (c) => res . send ('usr updated ')) . catch
   (err => res . status (400) send (err));
});

//Delete
app . delete ('/usr /:id' (req .res) => {
usr . find By Id And Delete (req . params .id ).
then (c) =>
 res.send ('usr delete')). catch (err => res . status
  (400)
  send (err));
 });

app . listen (3000 , c) => console . log (' Serve
running on port 3000'));
```