INFORMATION SECURITY ANALYSIS AND AUDIT

CSE-3501

LAB DIGITAL ASSIGNMENT-5

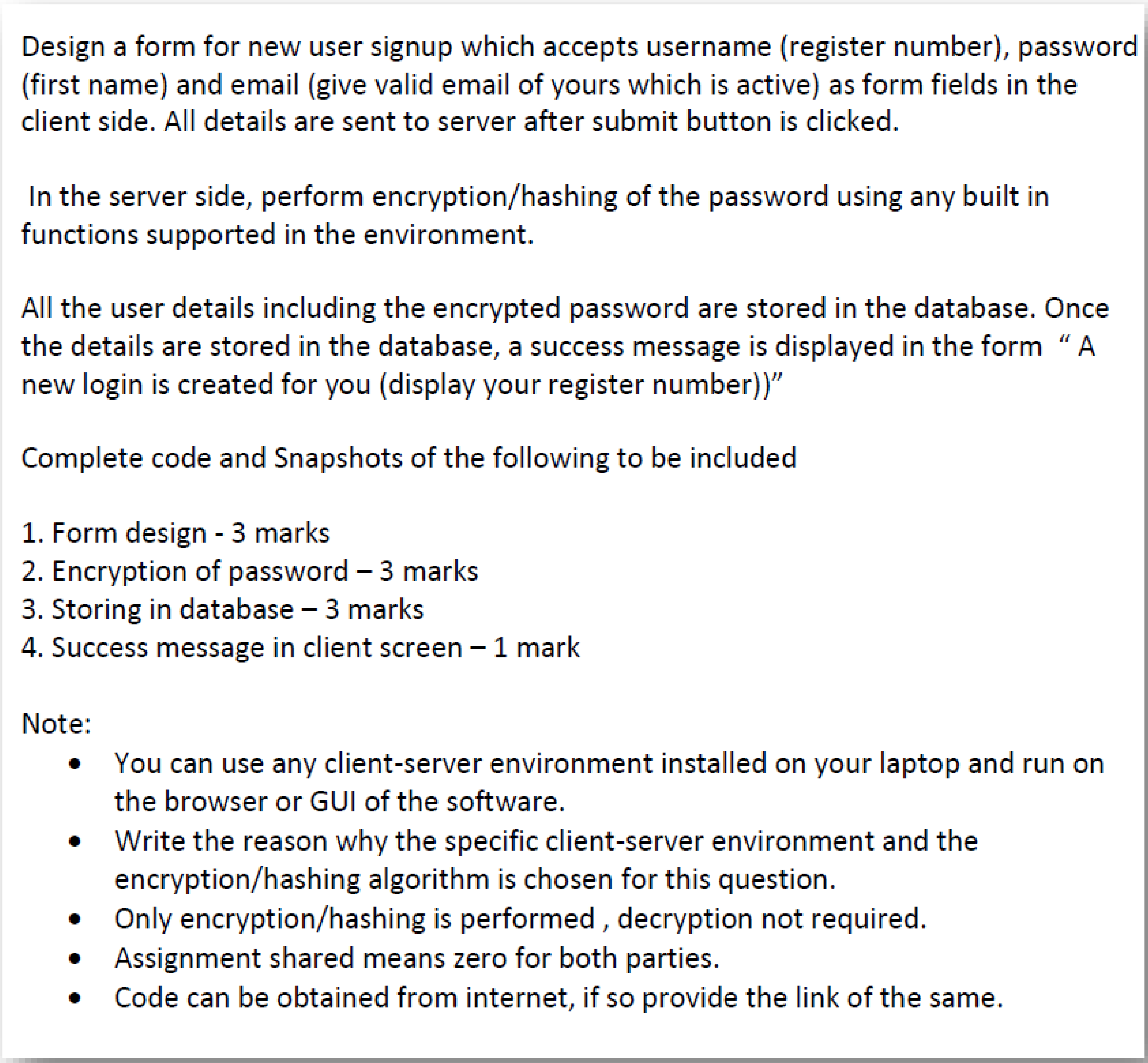
SLOT: L31+32

HRITISHA- 18BIT0226

FACULTY: Dr. Sumaiya Thaseen

SIGNUP NEW USER

Signup Form



Design a form for new user signup which accepts username (register number), password (first name) and email (give valid email of yours which is active) as form fields in the client side. All details are sent to server after submit button is clicked.

In the server side, perform encryption/hashing of the password using any built in functions supported in the environment.

All the user details including the encrypted password are stored in the database. Once the details are stored in the database, a success message is displayed in the form " A new login is created for you (display your register number))"

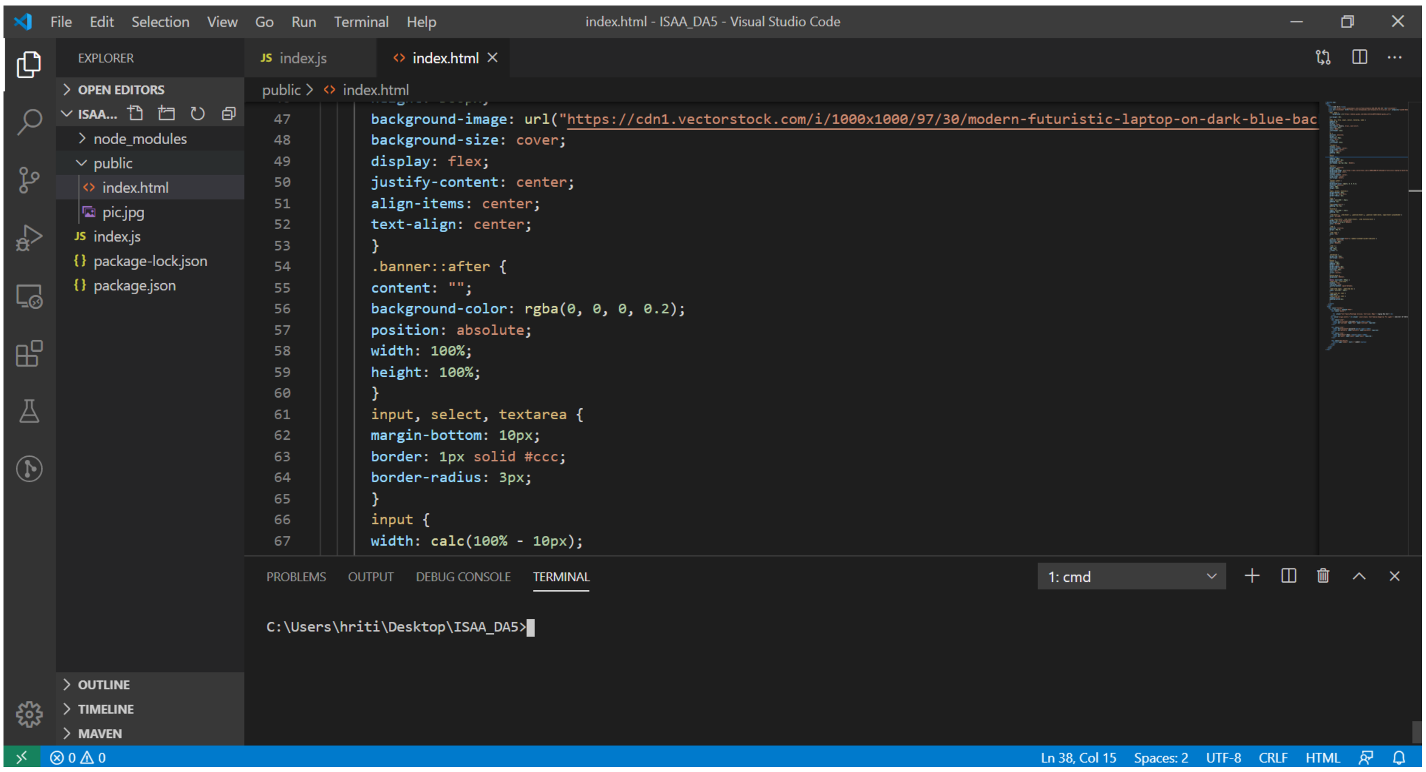
Complete code and Snapshots of the following to be included

1. Form design - 3 marks
2. Encryption of password – 3 marks
3. Storing in database – 3 marks
4. Success message in client screen – 1 mark

Note:

- You can use any client-server environment installed on your laptop and run on the browser or GUI of the software.
- Write the reason why the specific client-server environment and the encryption/hashing algorithm is chosen for this question.
- Only encryption/hashing is performed , decryption not required.
- Assignment shared means zero for both parties.
- Code can be obtained from internet, if so provide the link of the same.

EDITOR USED- VISUAL STUDIO CODE


File Edit Selection View Go Run Terminal Help
index.html - ISAA_DA5 - Visual Studio Code
EXPLORER
OPEN EDITORS
ISAA...
node_modules
public
index.html
pic.jpg
index.js
package-lock.json
package.json
OUTLINE
TIMELINE
MAVEN
index.js
index.html
public > index.html
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: cmd
Ln 38, Col 15 Spaces: 2 UTF-8 CRLF HTML

```
background-image: url("https://cdn1.vectorstock.com/i/1000x1000/97/30/modern-futuristic-laptop-on-dark-blue-bac
background-size: cover;
display: flex;
justify-content: center;
align-items: center;
text-align: center;
}
.banner::after {
content: "";
background-color: rgba(0, 0, 0, 0.2);
position: absolute;
width: 100%;
height: 100%;
}
input, select, textarea {
margin-bottom: 10px;
border: 1px solid #ccc;
border-radius: 3px;
}
input {
width: calc(100% - 10px);
```

```
C:\Users\hriti\Desktop\ISAA_DA5>
```

TECHNOLOGY STACK/ CLIENT SERVER ARCHITECTURE USED

1. CLIENT SIDE (FRONTEND)

<HTML> and {CSS} have been used to develop the interface. HTML for content structure for defining the content and CSS to style the appearance of content.

2. SERVER FRAMEWORK (BACKEND)

Node JS and Express JS are used for the rendering of user side or the frontend to the server side.

For managing the data from the user I have used **mongoDB Compass**

Reason for Choosing this Client Server Architecture

Node.js offers the users the luxury of writing the server-side applications in the JavaScript. This allows the Node.js developers to write both the front-end as well as the back-end web application in JavaScript using a runtime environment.

And they don’t need to use any other server-side programming language. It also makes the deployment of the web applications simpler because almost all the web browsers support JavaScript.

Also many languages such as PHP, .NET are no more in trend and many developers prefer to use this mode.

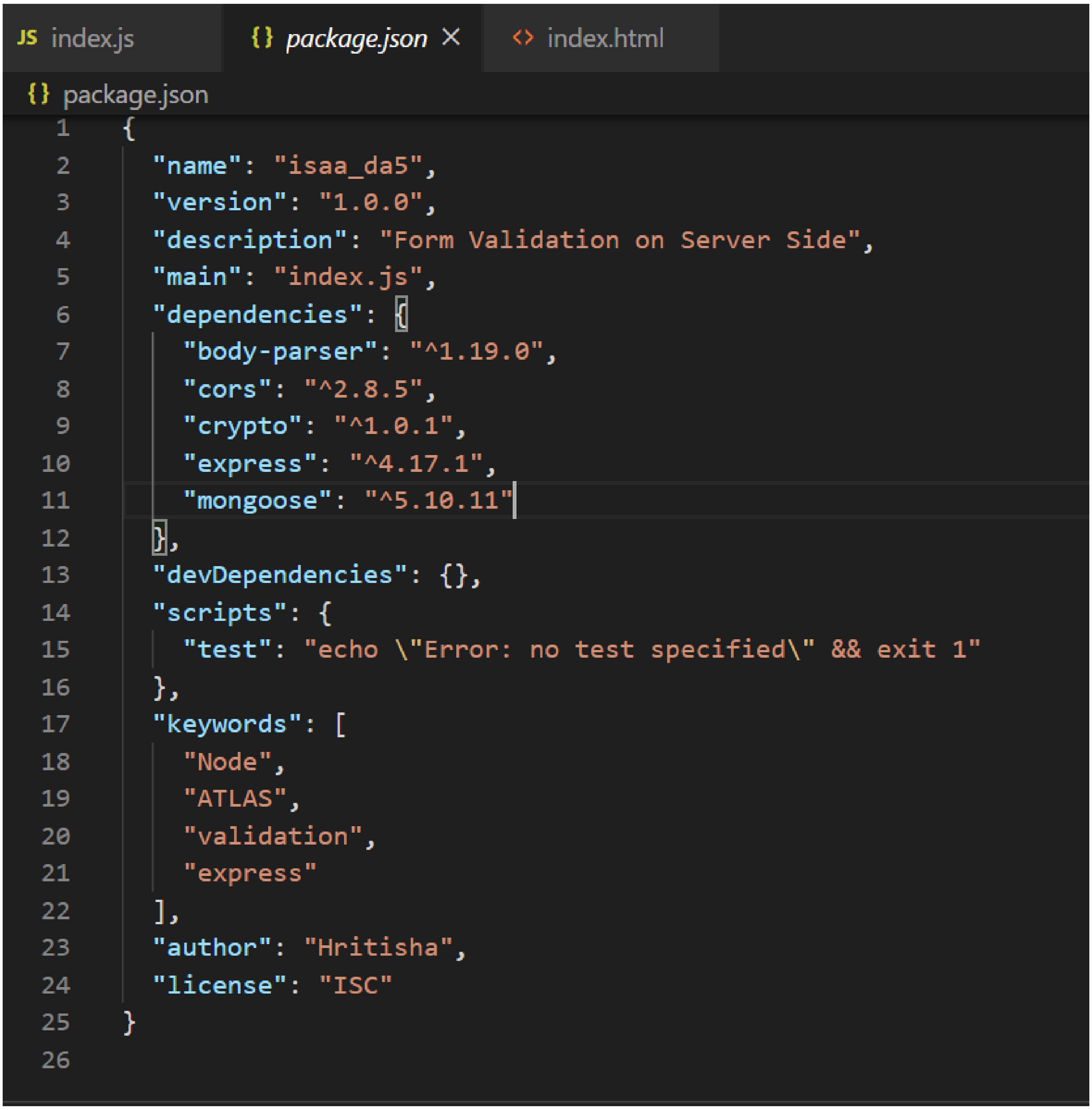
MongoDB- the document data model is a powerful way to store and retrieve data that allows developers to move fast. **MongoDB's** horizontal, scale-out architecture can support huge volumes of both data and traffic.

We can make use of mongoose or mongoclient for creating the data base and inserting the data. I have used Mongoose for the same.

COMMON MODULES USED

- EXPRESS
- BODY-PARSER
- MONGOOSE
- CRYPTO: Through this module I have used the encryption algorithm AES-128-CCM for the password

SOME USEFUL INFORMATION



```json
{
  "name": "isaa_da5",
  "version": "1.0.0",
  "description": "Form Validation on Server Side",
  "main": "index.js",
  "dependencies": {
    "body-parser": "^1.19.0",
    "cors": "^2.8.5",
    "crypto": "^1.0.1",
    "express": "^4.17.1",
    "mongoose": "^5.10.11"
  },
  "devDependencies": {},
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "Node",
    "ATLAS",
    "validation",
    "express"
  ],
  "author": "Hritisha",
  "license": "ISC"
}
```

CLIENT-SIDE FRAMEWORK

index.html

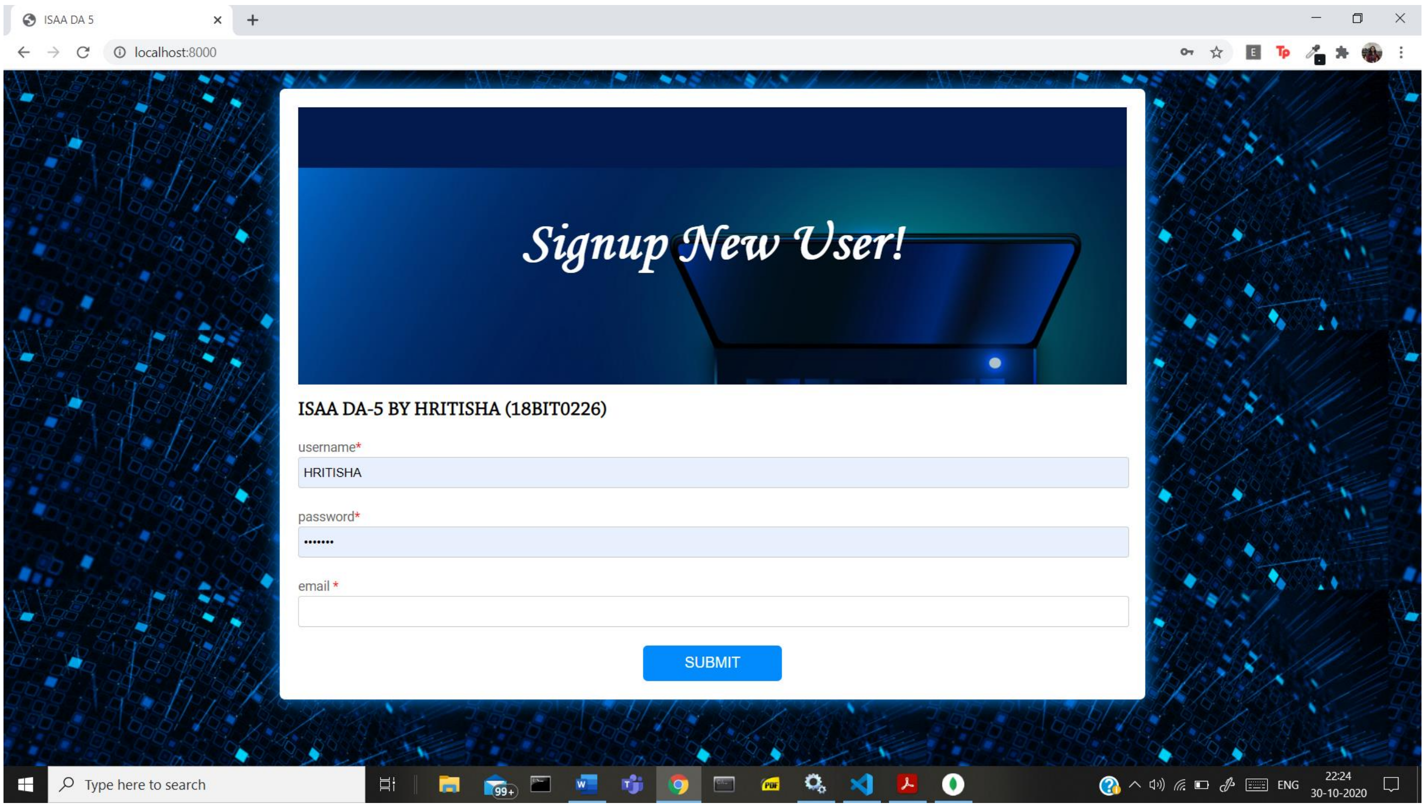
```
<!DOCTYPE html>
<html>
  <head>
    <title> 18BIT0226 ISAA DA 5</title>
    <link href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700" rel="stylesh
eet">
    <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.5.0/css/all.css" in
tegrity="sha384-
B4dIYHKNBt8Bc12p+WXckhzcICo0wtJAoU8YZTY5qE0Id1GSseTk6S+L3BlXeVIU" crossorigin="anonymous">
    <style>
      html, body {
          background: url("https://media1.giphy.com/media/3oFzmrqRPhYnFg9oGs/giphy.gif");

      min-height: 50%;
      }
      body, div, form, input, select, textarea, label {
      padding: 0;
      margin: 0;
      outline: none;
      font-family: Roboto, Arial, sans-serif;
      font-size: 14px;
      color: #666;
      line-height: 22px;
      }
      h1 {
      position: absolute;
      margin: 0;
      font-size: 40px;
      color: #fff;
      z-index: 2;
      line-height: 83px;
      }
      .testbox {
      display: flex;
      justify-content: center;
      align-items: center;
      height: inherit;
      padding: 20px;
      }
      form {
      width: 60%;
      padding: 20px;
      border-radius: 6px;
      background: #fff;
      box-shadow: 0px 0px 20px  #028bfc;
      }
      .banner {
      position: relative;
      height: 300px;
      background-image: url("https://cdn1.vectorstock.com/i/1000x1000/97/30/modern-
futuristic-laptop-on-dark-blue-background-vector-29119730.jpg");
      background-size: cover;
      display: flex;
      justify-content: center;
      align-items: center;
      text-align: center;
      }
      .banner::after {
      content: "";
      background-color: rgba(0, 0, 0, 0.2);
      position: absolute;
      width: 100%;
      height: 100%;
      }
      input, select, textarea {
      margin-bottom: 10px;
      border: 1px solid #ccc;
      border-radius: 3px;
      }
      input {
      width: calc(100% - 10px);
      padding: 5px;
      }
      input[type="date"] {
      padding: 4px 5px;
      }
      textarea {
      width: calc(100% - 12px);
      padding: 5px;
      }
      .item:hover p, .item:hover i, .question:hover p, .question label:hover, input:hover::p
laceholder {
      color: #cc7a00;
      }
      .item input:hover, .item select:hover, .item textarea:hover {
      border: 1px solid transparent;
      box-shadow: 0 0 3px 0 #028bfc;
      color: #cc7a00;
      }
      .item {
      position: relative;
      margin: 10px 0;
      }
      .item span {
      color: red;
```

```
    }

    .item i, input[type="date"]::-webkit-calendar-picker-indicator {
    position: absolute;
    font-size: 20px;
    color: #028bfc;
    }
    .item i {
    right: 1%;
    top: 30px;
    z-index: 1;
    }

    .btn-block {
    margin-top: 10px;
    text-align: center;
    }
    button {
    width: 150px;
    padding: 10px;
    border: none;
    border-radius: 5px;
    background: #028bfc;
    font-size: 16px;
    color: #fff;
    cursor: pointer;
    }
    button:hover {
    background: #001927;
    }
    @media (min-width: 568px) {
    .name-item, .city-item {
    display: flex;
    flex-wrap: wrap;
    justify-content: space-between;
    }
    .name-item input, .name-item div {
    width: calc(50% - 20px);
    }
    .name-item div input {
    width:97%;}
    .name-item div label {
    display:block;
    padding-bottom:5px;
    }
    }
  </style>
</head>
<body>
  <div class="testbox">
```

```
<form action="/" method="POST">
        <div class="banner">

          <h1  style="font-family:Monotype Corsiva; font-size: 60px;"> Signup New User!</h1>
        </div>
       <div style="align:center;"><h2 style=" color:black; font-
family:Chaparral Pro Light;"> ISAA DA-5 BY HRITISHA (18BIT0226)</h2>
       </div>
        <div class="item">
          <label for="username">username<span>*</span></label>
          <input id="username" type="text" name="username" required/>
        </div>

        <div class="item">
          <label for="password">password<span>*</span></label>
          <input id="password" type="password" name="password" required/>
        </div>
        <div class="item">
          <label for="email">email <span>*</span></label>
          <input id="email" type="email" name="email" required/>
        </div>

        <div class="btn-block">
          <button type="submit" href="/">SUBMIT</button>
        </div>
      </form>
    </div>
  </body>
</html>
```

FORM

The form takes USERNAME, PASSWORD and EMAIL as inputs. And through the server-side language express and node and the POST request this html is rendered to the server side.



ISAA DA-5 BY HRITISHA (18BIT0226)

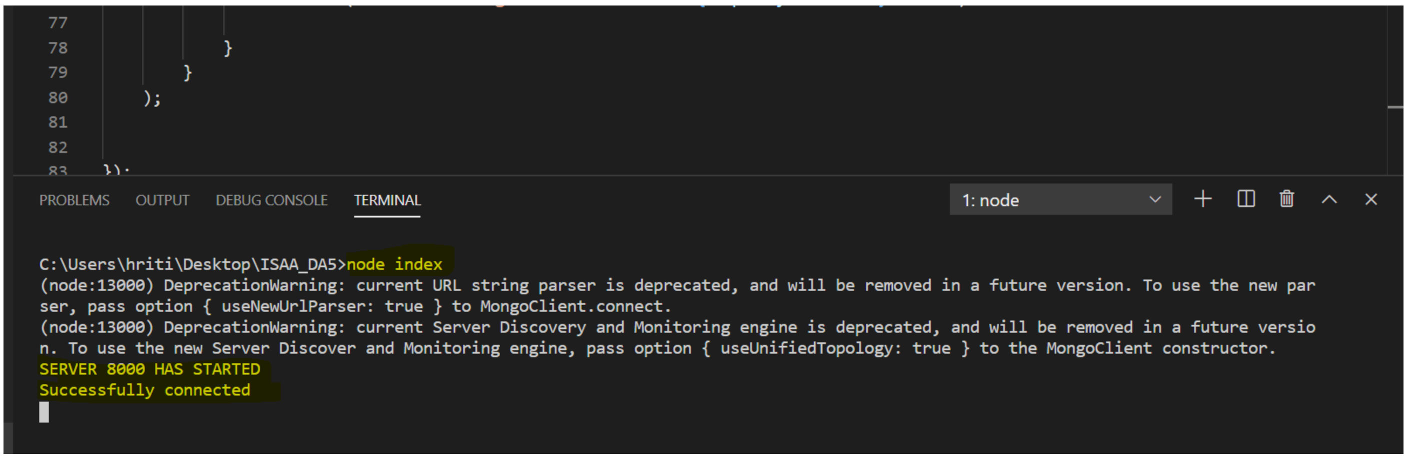
username*: HRITISHA

password*: •••••••

email *:

This has been achieved by running on the localhost through node.

Index.js file will help to render the html page to the server side.

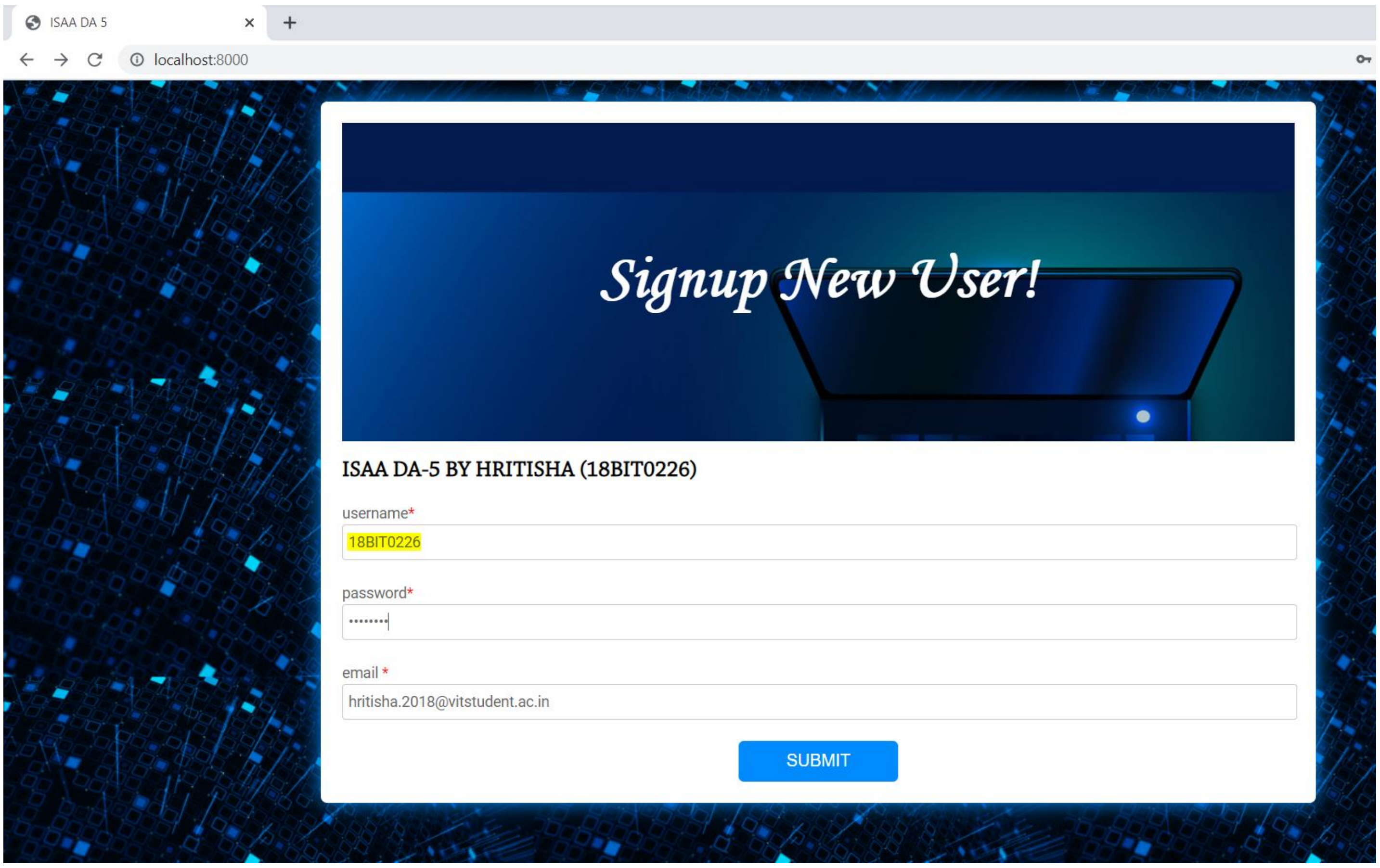


```
            }
        }
    );


});
```

```
C:\Users\hriti\Desktop\ISAA_DA5>node index
(node:13000) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the new par
ser, pass option { useNewUrlParser: true } to MongoClient.connect.
(node:13000) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future versio
n. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
SERVER 8000 HAS STARTED
Successfully connected
```

LOCALHOST PORT- 8000

The database is also successfully connected.


ISAA DA 5

localhost:8000

Signup New User!

ISAA DA-5 BY HRITISHA (18BIT0226)

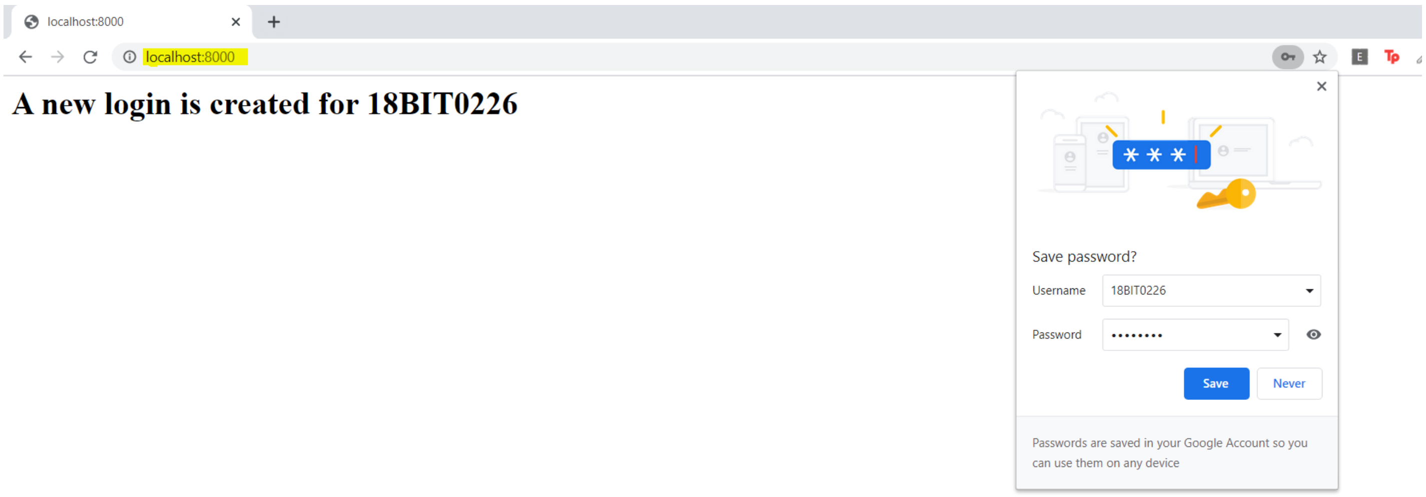
username*
18BIT0226

password*
••••••••

email *
hritisha.2018@vitstudent.ac.in

SUBMIT

After filling the desired details of the user we are able to see that the page takes us to the new page which says that “ *A new login is created for 18BIT0226”*


localhost:8000

localhost:8000

A new login is created for 18BIT0226

Save password?

Username 18BIT0226

Password ••••••••

Save Never

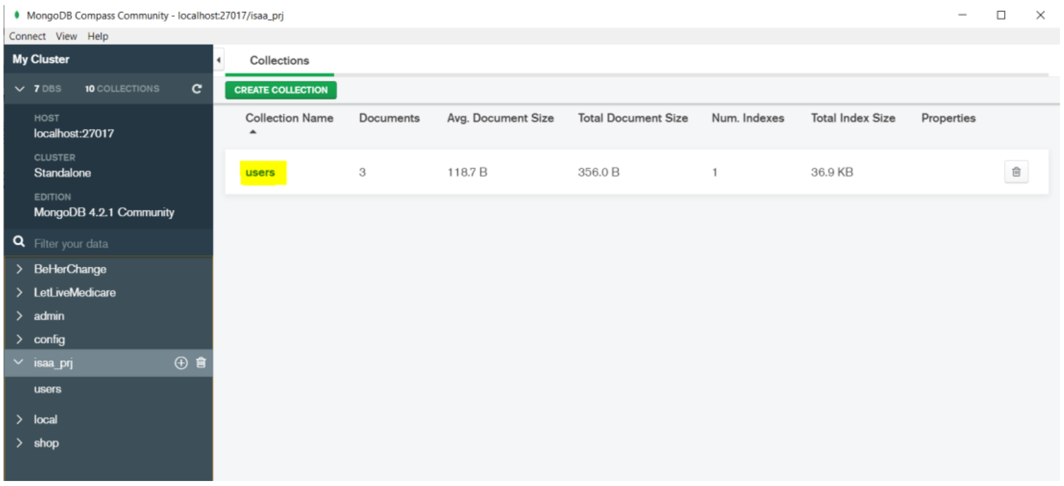
Passwords are saved in your Google Account so you can use them on any device

SUCCESS MESSAGE FOR A NEW USER

SERVER-SIDE FRAMEWORK

```
var express = require("express");
var bodyParser = require("body-parser");

const crypto = require('crypto');
var app = express();
var mongoose=require('mongoose');

app.use(bodyParser.urlencoded({ extended: true }));
app.use(express.static("public"));
mongoose.connect('mongodb://localhost:27017/isaa_prj', function (err) {
   if (err) throw err;
   console.log('Successfully connected');
});


var userSchema = new mongoose.Schema({
    username: String,
    password: String,
    email: String,
});


var user = mongoose.model("user", userSchema);


app.get("/", function (req, res) {
    res.render("index");

})
```

This helps to connect the frontend to the backend. With the help of Mongoose I have connected to LOCALHOST: 27017/ISAA PROJECT .



HOST localhost:27017
CLUSTER Standalone
EDITION MongoDB 4.2.1 Community

BeHerChange
LetLiveMedicare
admin
config
isaa_prj
users
local
shop

Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
users	3	118.7 B	356.0 B	1	36.9 KB	

ALL the data will be stored here in user collection under the database isaa_prj.

ENCRYPTION ALGORITHM USED

Aes-128-ccm

This is a base of Encryption though AES 128 bytes.

CCM is a generic authenticated encryption block cipher mode. CCM is defined for use with any 128-bit block cipher. I have used it w with AES block cipher.

AES-CCM has four inputs: an AES key, a nonce, a plaintext, and

optional additional authenticated data (AAD).

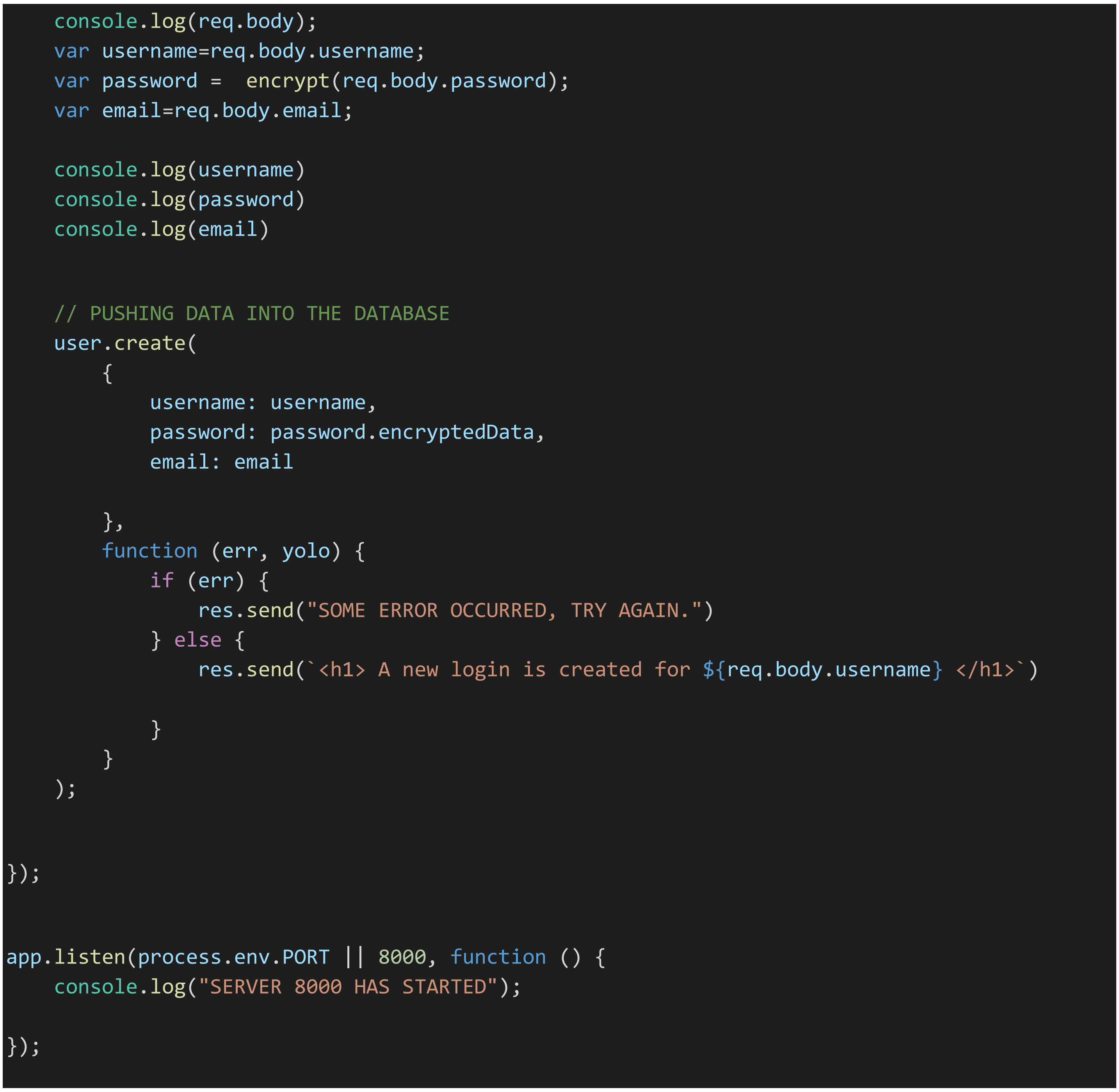
AES-CCM generates two outputs: a ciphertext and a message authentication code (also called

an authentication tag). I have generated only the cipher text for the PASSWORD

CCM is a mode of operation for cryptographic block ciphers. It is an authenticated encryption algorithm designed to provide both authentication and confidentiality. CCM mode is only defined for block ciphers with a block length of 128 bits

This algorithm is used under the encrypt function:

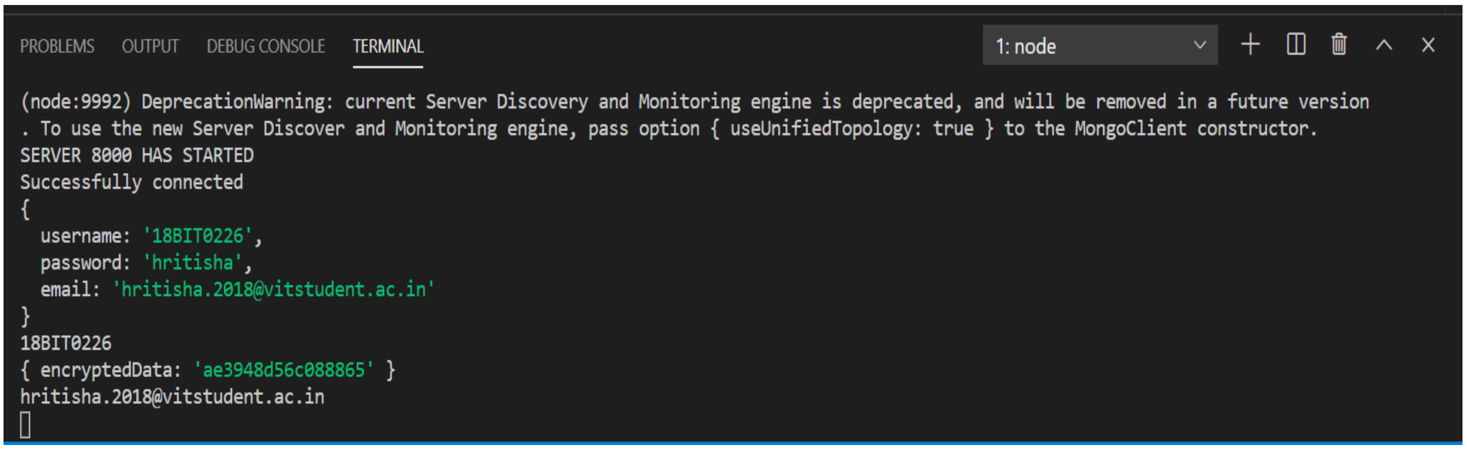
```
function encrypt(text) {

const key = 'keykeykeykeykeykeykeykey';
const nonce = crypto.randomBytes(12);

const aad = Buffer.from('0123456789', 'hex');
const cipher = crypto.createCipheriv('aes-192-ccm', key, nonce, {
  authTagLength: 16
});
cipher.setAAD(aad, {
  plaintextLength: Buffer.byteLength(text)
});
const ciphertext = cipher.update(text, 'utf8');

encrypted = Buffer.concat([ciphertext, cipher.final()]);
return { encryptedData: encrypted.toString('hex') };

}

app.post("/", function (req, res) {
```



```
    console.log(req.body);
    var username=req.body.username;
    var password =  encrypt(req.body.password);
    var email=req.body.email;

    console.log(username)
    console.log(password)
    console.log(email)


    // PUSHING DATA INTO THE DATABASE
    user.create(
        {
            username: username,
            password: password.encryptedData,
            email: email

        },
        function (err, yolo) {
            if (err) {
                res.send("SOME ERROR OCCURRED, TRY AGAIN.")
            } else {
                res.send(`<h1> A new login is created for ${req.body.username} </h1>`)

            }
        }
    );


});


app.listen(process.env.PORT || 8000, function () {
    console.log("SERVER 8000 HAS STARTED");

});
```

OUTPUT ON THE SERVER SIDE AND INSERTION IN THE DATABASE

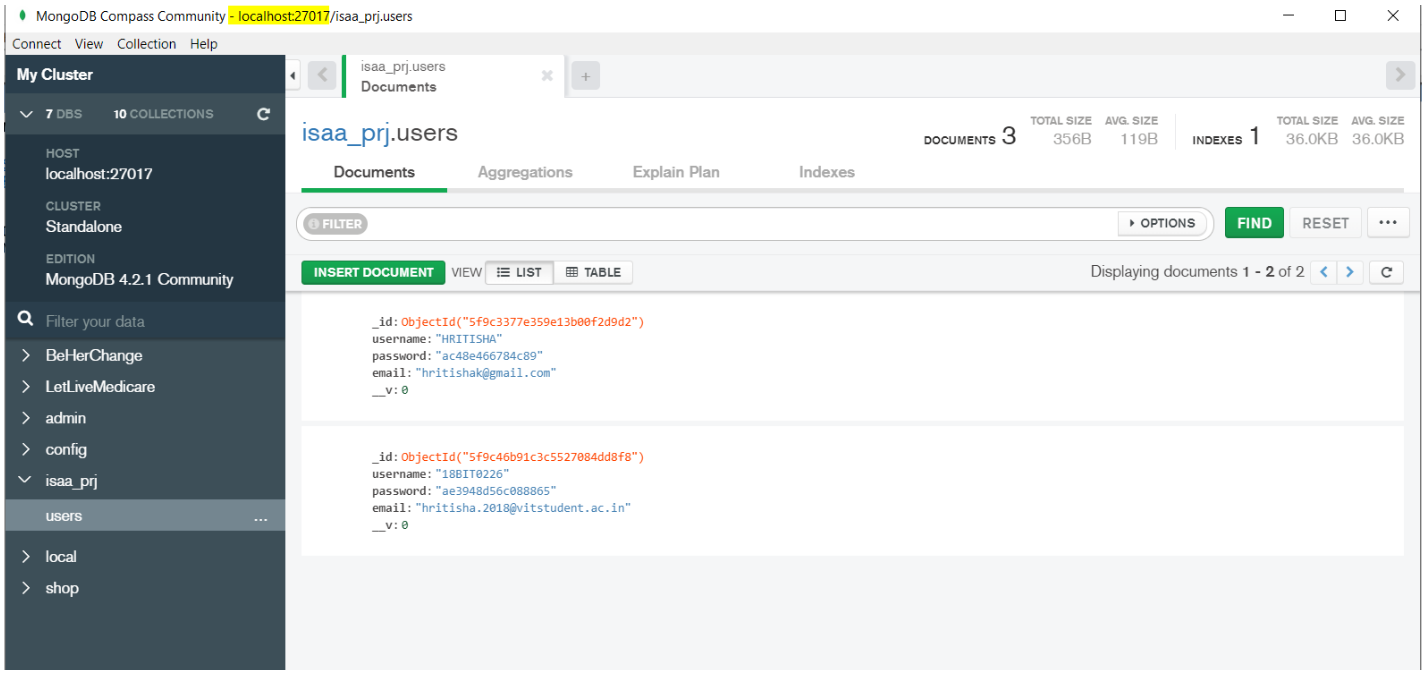
We can see that the PASSWORD is encrypted



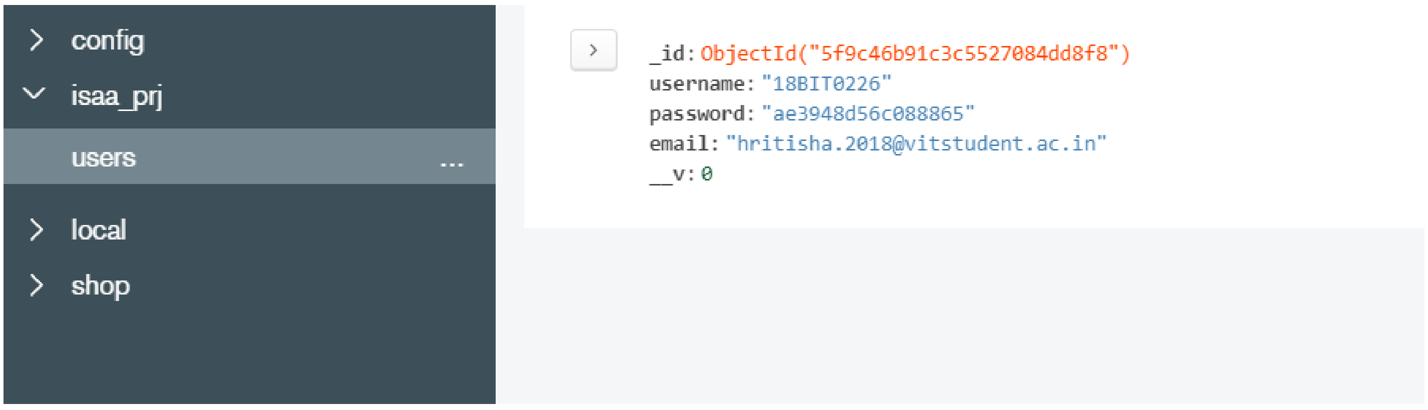
```
(node:9992) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version
. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
SERVER 8000 HAS STARTED
Successfully connected
{
  username: '18BIT0226',
  password: 'hritisha',
  email: 'hritisha.2018@vitstudent.ac.in'
}
18BIT0226
{ encryptedData: 'ae3948d56c088865' }
hritisha.2018@vitstudent.ac.in
```

INSERTION IN THE mongoDB DATABASE

As mentioned earlier the database name is isaa_prj and data will be visible inside the users collection



```
_id: ObjectId("5f9c3377e359e13b00f2d9d2")
username: "HRITISHA"
password: "ac48e466784c89"
email: "hritishak@gmail.com"
__v: 0

_id: ObjectId("5f9c46b91c3c5527084dd8f8")
username: "18BIT0226"
password: "ae3948d56c088865"
email: "hritisha.2018@vitstudent.ac.in"
__v: 0
```



```
_id: ObjectId("5f9c46b91c3c5527084dd8f8")
username: "18BIT0226"
password: "ae3948d56c088865"
email: "hritisha.2018@vitstudent.ac.in"
__v: 0
```

REFERENCES

- https://www.w3docs.com/tools/editor/5961 -FRONTEND
- https://nodejs.org/api/crypto.html - ENCRPTION
- https://expressjs.com/en/guide/database-integration.html#mongodb- Mongoose Express

Hritisha

Digitally signed by Hritisha
Date: 2020.10.30 23:00:04 +05'30'