complete example:

backend (Node + Express + Mongoose + bcrypt + JWT) and frontend (React functional components)

using email id: gnoteskarur1@gmail.com for mongocloud

Install (backend)

```
# in backend folder
npm init -y
npm install express mongoose bcrypt jsonwebtoken dotenv cors helmet express-rate-limit
# optionally: nodemon --save-dev
```

If `bcrypt` fails to build on some systems, you can use `bcryptjs` (pure JS) as a drop-in replacement: `npm i bcryptjs` and replace `const bcrypt = require('bcryptjs');`.

Backend folder structure:

```
backend/
  .env
  server.js
  models/User.js
  routes/auth.js
```

.env

```
MONGO_URI=mongodb+srv://gnoteskarur1:7t4CsgHcILJH2t35@gcluster.ynbe5qm.mongodb.net/mydb?appName=gcluster
JWT_SECRET=periyaragasiyam
PORT=4000
```

models/User.js

```
const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({

  email: { type: String, required: true, unique: true, lowercase: true, trim: true },

  passwordHash: { type: String, required: true },

  name: { type: String }

}, { timestamps: true });

module.exports = mongoose.model('User', userSchema);
```

routes/Auth.js

```
const express = require('express');

const router = express.Router();

const bcrypt = require('bcrypt'); // or 'bcryptjs'

const jwt = require('jsonwebtoken');

const User = require('../models/User');


const SALT_ROUNDS = 12; // 10-12 is common. Higher = more secure but slower.
```

```
router.post('/register', async (req, res) => {
  try {
    const { email, password, name } = req.body;
    if (!email || !password) return res.status(400).json({ error: 'Email and password required' });

    const existing = await User.findOne({ email });
    if (existing) return res.status(409).json({ error: 'User already exists' });

    // Hash password
    const passwordHash = await bcrypt.hash(password, SALT_ROUNDS);

    const user = new User({ email, passwordHash, name });
    await user.save();

    // Return minimal info, no password
    return res.status(201).json({ id: user._id, email: user.email, name: user.name });
  } catch (err) {
    console.error(err);
    return res.status(500).json({ error: 'Server error' });
  }
});

router.post('/login', async (req, res) => {
```

```
  try {
    const { email, password } = req.body;
    if (!email || !password) return res.status(400).json({ error: 'Email and password required' });

    const user = await User.findOne({ email });
    if (!user) return res.status(401).json({ error: 'Invalid credentials' });

    const match = await bcrypt.compare(password, user.passwordHash);
    if (!match) return res.status(401).json({ error: 'Invalid credentials' });

    // Create JWT (short expiry recommended)
    const token = jwt.sign({ sub: user._id, email: user.email }, process.env.JWT_SECRET, { expiresIn: '1h' });

    // Send token and minimal user info
    return res.json({ token, user: { id: user._id, email: user.email, name: user.name } });
  } catch (err) {
    console.error(err);
    return res.status(500).json({ error: 'Server error' });
  }
});

module.exports = router;
```

server.js

```
require('dotenv').config();
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const helmet = require('helmet');
const rateLimit = require('express-rate-limit');

const authRouter = require('./routes/auth');

const app = express();
app.use(helmet());
app.use(express.json());
app.use(cors({
  origin: 'http://localhost:3000' // adjust in prod
}));

// Basic rate limiter (adjust in production)
const limiter = rateLimit({
  windowMs: 15 * 60 * 1000, // 15 minutes
  max: 100
});
app.use(limiter);

// routes
```

```
app.use('/api/auth', authRouter);


// connect DB & start
mongoose.connect(process.env.MONGO_URI, { useNewUrlParser: true, useUnifiedTopology: true })
  .then(() => {
    console.log('mongo connected');
    const port = process.env.PORT || 4000;
    app.listen(port, () => console.log(`server running on ${port}`));
  })
  .catch(err => {
    console.error('mongo connection error', err);
  });
```

Frontend (React) — minimal example

Install:

npx create-react-app .

 # no extra packages required for this minimal example

src/Register.js

```
import React, { useState } from 'react';


export default function Register() {
  const [form, setForm] = useState({ email: '', password: '', name: '' });
  const [msg, setMsg] = useState(null);


  const handleChange = e => setForm({ ...form, [e.target.name]: e.target.value
});


  const submit = async e => {
    e.preventDefault();
    setMsg(null);
    try {
      const res = await fetch('http://localhost:4000/api/auth/register', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ email: form.email, password: form.password, name:
form.name })
      });
```

```
    const data = await res.json();
    if (!res.ok) throw new Error(data.error || 'Registration failed');
    setMsg('Registered successfully. Please login.');
  } catch (err) {
    setMsg(err.message);
  }
 };

 return (
  <form onSubmit={submit}>
    <h2>Register</h2>
    <input name="name" placeholder="Name" value={form.name}
onChange={handleChange} />
    <input name="email" placeholder="Email" value={form.email}
onChange={handleChange} />
    <input name="password" type="password" placeholder="Password"
value={form.password} onChange={handleChange} />
    <button type="submit">Register</button>
    {msg && <p>{msg}</p>}
  </form>
 );
}
```

src/Login.js

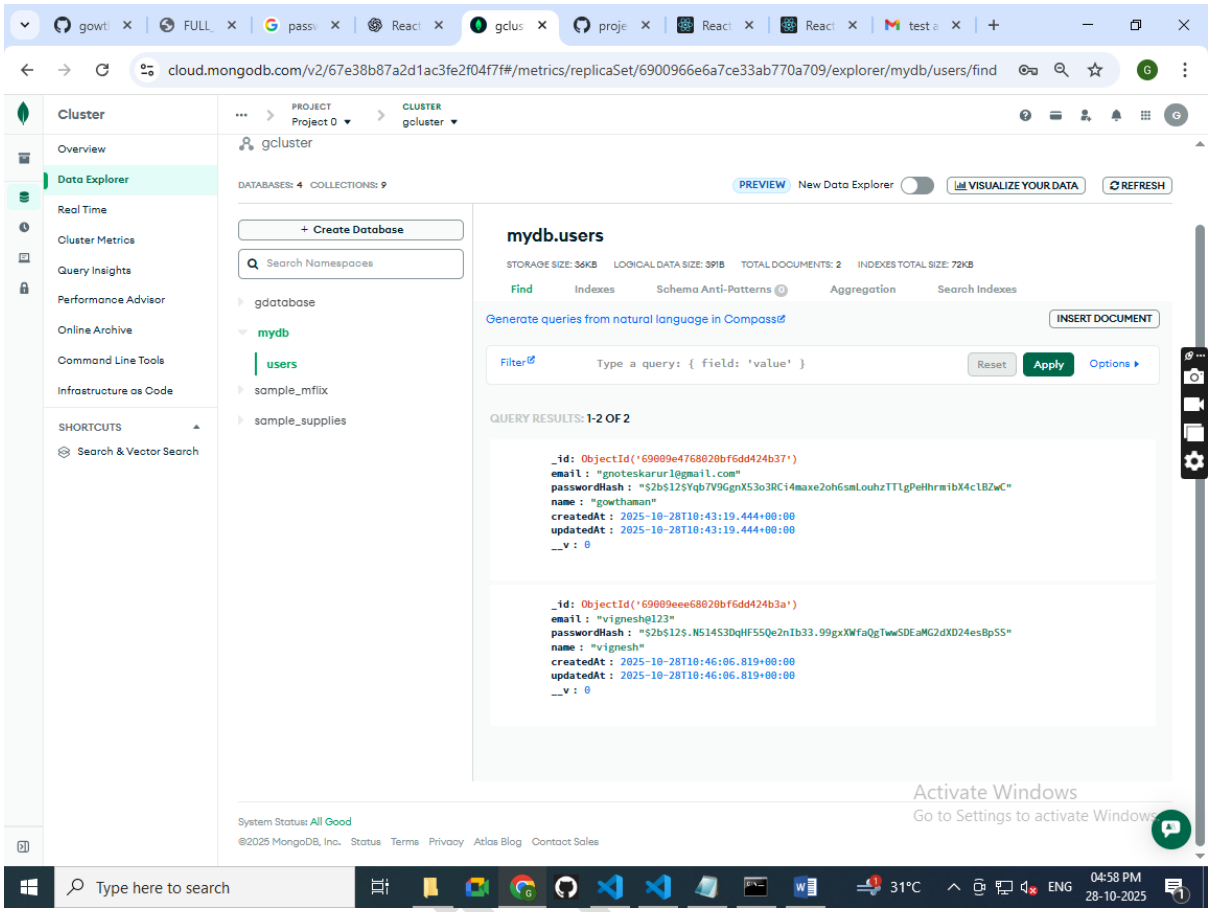
```
import React, { useState } from 'react';


export default function Login({ onLogin }) {
  const [form, setForm] = useState({ email: '', password: '' });
  const [msg, setMsg] = useState(null);


  const handleChange = e => setForm({ ...form, [e.target.name]: e.target.value
});


  const submit = async e => {
    e.preventDefault();
    setMsg(null);
    try {
      const res = await fetch('http://localhost:4000/api/auth/login', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify(form)
      });
      const data = await res.json();
      if (!res.ok) throw new Error(data.error || 'Login failed');


      // store token securely: localStorage is common but vulnerable to XSS.
      // For higher security use httpOnly cookies set by server.
      localStorage.setItem('token', data.token);
      onLogin(data.user);
```

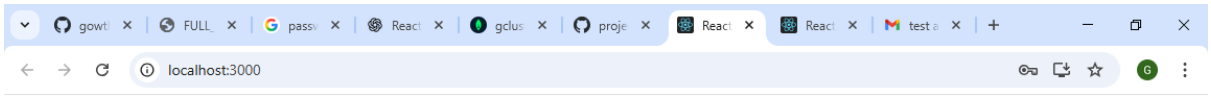
```
    } catch (err) {
      setMsg(err.message);
    }
  };

  return (
    <form onSubmit={submit}>
      <h2>Login</h2>
      <input name="email" placeholder="Email" value={form.email}
onChange={handleChange} />
      <input name="password" type="password" placeholder="Password"
value={form.password} onChange={handleChange} />
      <button type="submit">Login</button>
      {msg && <p>{msg}</p>}
    </form>
  );
}
```

src/App.js

```
import React, { useState } from 'react';
import Register from './Register';
import Login from './Login';

function App() {
  const [user, setUser] = useState(null);

  return (
    <div style={{ padding: 20 }}>
      {!user ? (
        <>
          <Register />
          <Login onLogin={setUser} />
        </>
      ) : (
        <div>
          <h3>Welcome, {user.name || user.email}</h3>
          <button onClick={() => { localStorage.removeItem('token'); setUser(null);
}}>Logout</button>
        </div>
      )}
    </div>
  );
}

export default App;
```

Output Screen shots



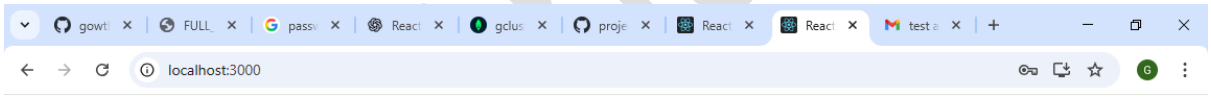


Register

Name | Email | Password | Register

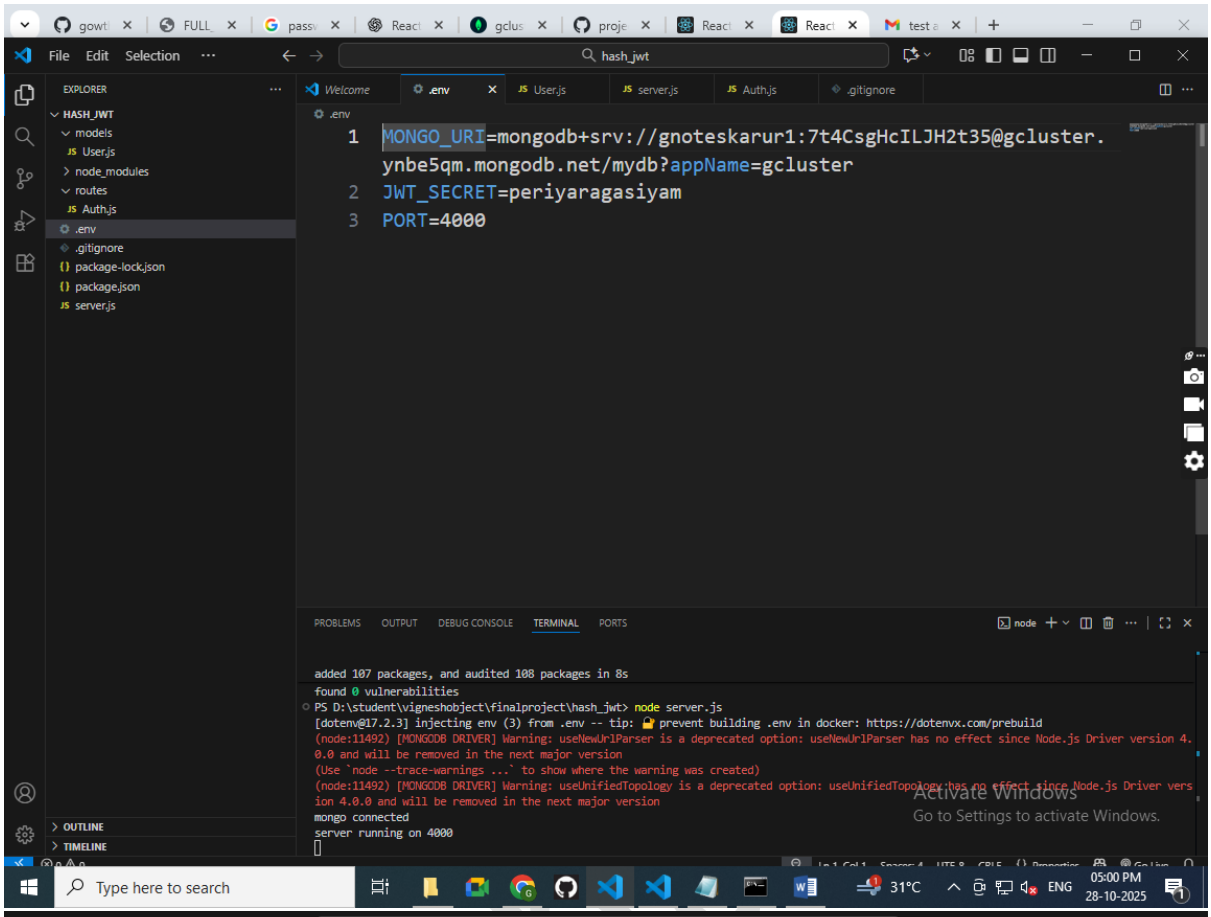
Login

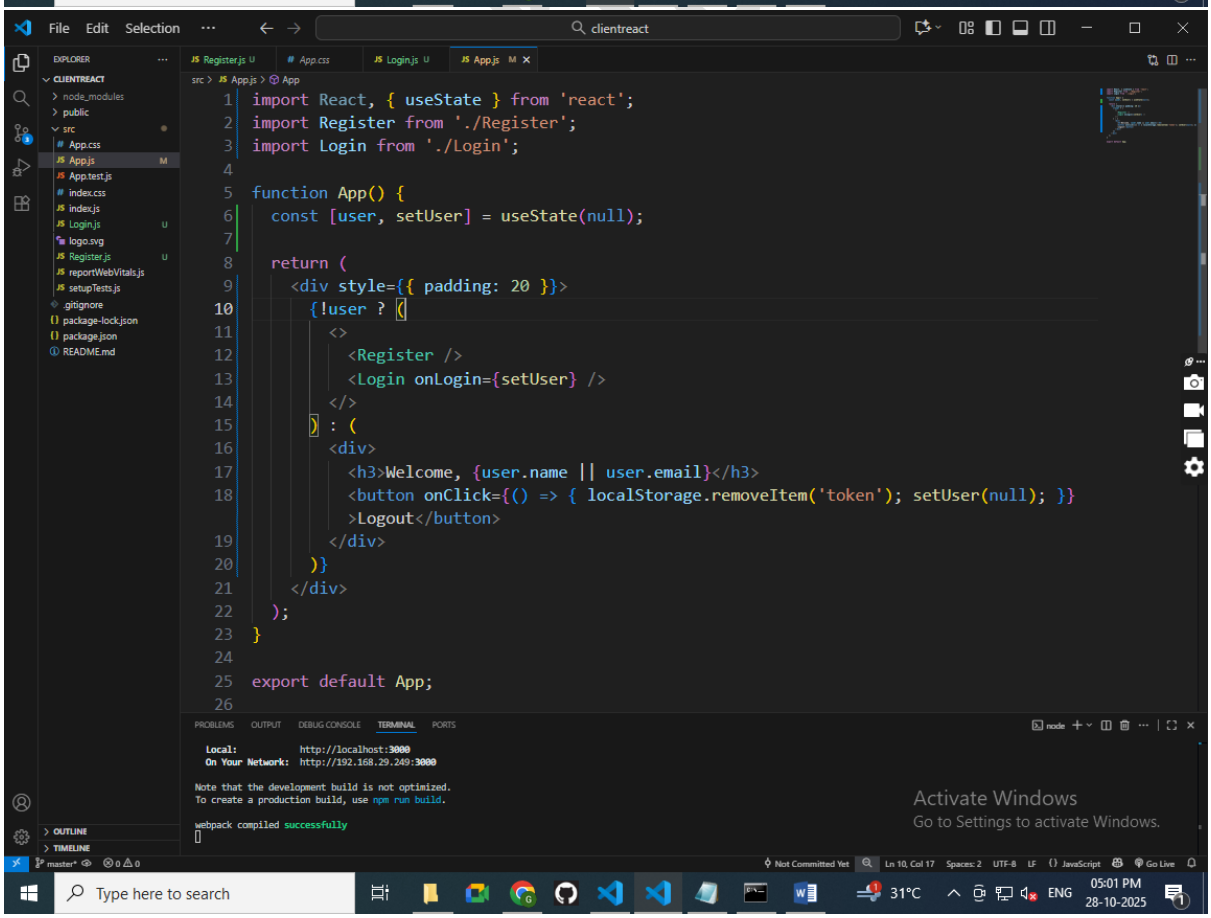
Email | Password | Login



Welcome, gowtham

Logout


File Edit Selection
hash_jwt
EXPLORER
HASH_JWT
models
User.js
node_modules
routes
Auth.js
.env
.gitignore
package-lock.json
package.json
server.js
Welcome
.env
User.js
server.js
Auth.js
.gitignore
1 MONGO_URI=mongodb+srv://gnoteskarur1:7t4CsgHcILJH2t35@gcluster.ynbe5qm.mongodb.net/mydb?appName=gcluster
2 JWT_SECRET=periyaragasiyam
3 PORT=4000
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
added 107 packages, and audited 108 packages in 8s
found 0 vulnerabilities
PS D:\student\vigneshobject\finalproject\hash_jwt> node server.js
[dotenv@17.2.3] injecting env (3) from .env -- tip: prevent building .env in docker: https://dotenvx.com/prebuild
(node:11492) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
(Use `node --trace-warnings ...` to show where the warning was created)
(node:11492) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
mongo connected
server running on 4000
Activate Windows
Go to Settings to activate Windows.
OUTLINE
TIMELINE
Type here to search
31°C
ENG
05:00 PM
28-10-2025


File Edit Selection
clientreact
EXPLORER
CLIENTREACT
node_modules
public
src
App.css
App.js
App.test.js
index.css
index.js
Login.js
logo.svg
Register.js
reportWebVitals.js
setupTests.js
.gitignore
package-lock.json
package.json
README.md
Register.js
App.css
Login.js
App.js
src > App.js > App
1 import React, { useState } from 'react';
2 import Register from './Register';
3 import Login from './Login';
4
5 function App() {
6 const [user, setUser] = useState(null);
7
8 return (
9 <div style={{ padding: 20 }}>
10 {!user ? (
11 <>
12 <Register />
13 <Login onLogin={setUser} />
14 </>
15) : (
16
17 <h3>Welcome, {user.name || user.email}</h3>
18 <button onClick={() => { localStorage.removeItem('token'); setUser(null); }}>Logout</button>
19
20)}
21
22);
23 }
24
25 export default App;
26
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Local: http://localhost:3000
On Your Network: http://192.168.29.249:3000
Note that the development build is not optimized.
To create a production build, use npm run build.
webpack compiled successfully
Activate Windows
Go to Settings to activate Windows.
OUTLINE
TIMELINE
Type here to search
31°C
ENG
05:01 PM
28-10-2025