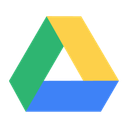
## Page:1

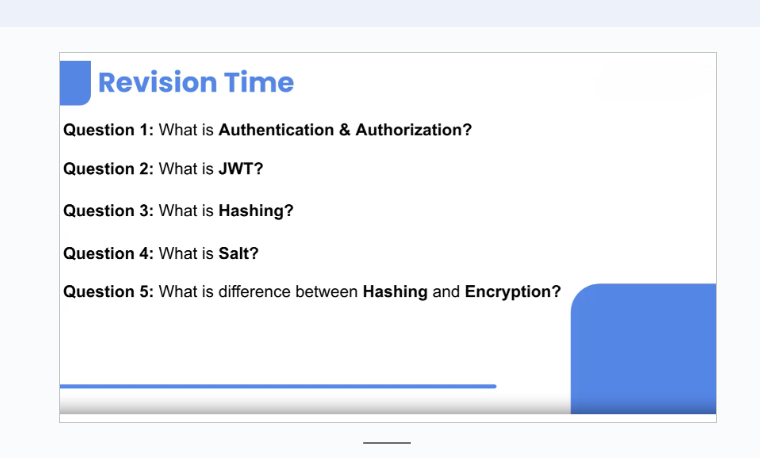
# **Fullstack CRUD-1**

[[](https://docs.google.com/presentation/d/1Q-qWAdsekPEni9yCNRV1hB-Bfjcomw9YXqf2vTIiofo/edit?usp=sharing)](https://docs.google.com/presentation/d/1Q-qWAdsekPEni9yCNRV1hB-Bfjcomw9YXqf2vTIiofo/edit?usp=sharing" \t "_blank)

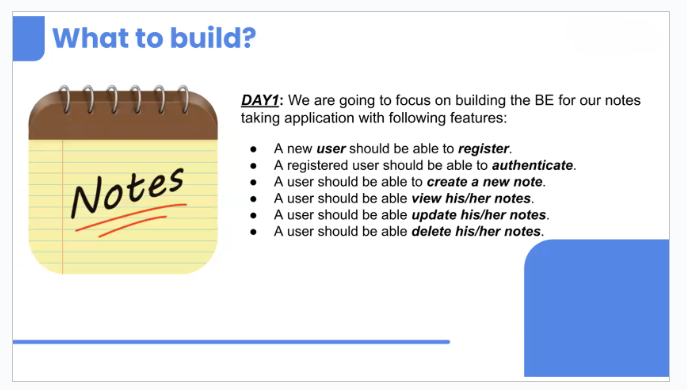
[1Q-qWAdsekPEni9yCNRV1hB-Bfjcomw9YXqf2vTIiofo](https://docs.google.com/presentation/d/1Q-qWAdsekPEni9yCNRV1hB-Bfjcomw9YXqf2vTIiofo/edit?usp=sharing" \t "_blank)

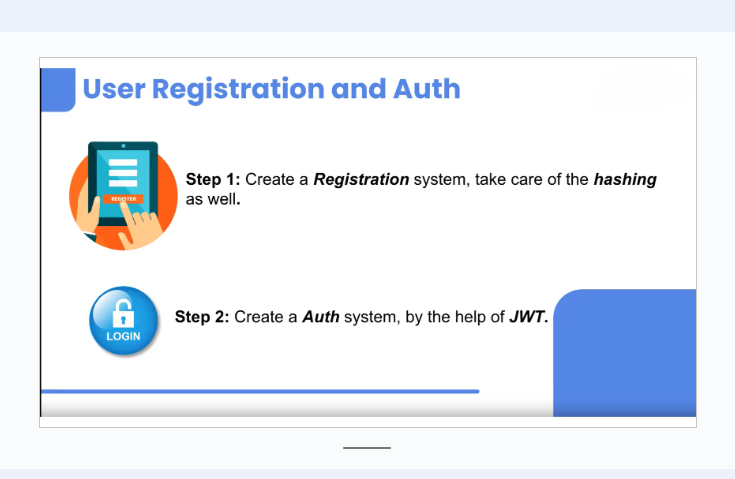
[https://docs.google.com](https://docs.google.com/presentation/d/1Q-qWAdsekPEni9yCNRV1hB-Bfjcomw9YXqf2vTIiofo/edit?usp=sharing" \t "_blank)



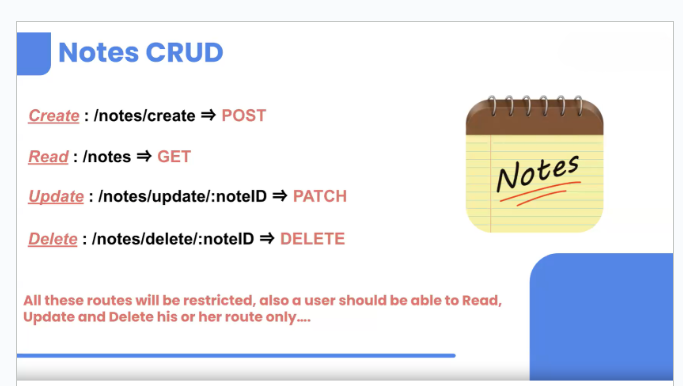


## Page:2





## Page:3



**📚 Notes 1**

**🔒 Hashing**

* **Purpose**: Hashing is done to protect passwords.
* Before storing the password in the database during signup, it should be hashed.
* **Why hash passwords?**
  + In a database, passwords can be visible and potentially misused in case of a data breach.
  + The best practice is to hash the password before storing it and then compare it back to the plain text password during login.
* **Package used**: bcrypt.

**🔑 Bcrypt**

* Bcrypt is a secure password hashing function designed to resist brute-force and rainbow table attacks.
* **How it works**:
  + Bcrypt is a one-way hash function. It takes a plain text password and produces a fixed-length hashed output, which cannot be reversed to retrieve the original password.

## Page:4

* + This makes it difficult for attackers to extract real passwords even if they gain access to the database.
  + **Why Bcrypt?** It’s widely considered a strong password hashing method used by developers to secure users’ passwords.
* For a deeper understanding, check out [bcrypt documentation](https://www.npmjs.com/package/bcrypt).

# 🚀 Implementation Example

**While Registering:**

**js**

**Copy code**

**app.post("/register", async (req, res) => {**

**const { name, email, pass, age } = req.body;**

**try {**

**bcrypt.hash(pass, 8, async (err, hash) => {**

**const user = new UserModel({ name, email, pass: hash, age });**

**await user.save();**

**res.send("Registered");**

**});**

**} catch (err) {**

**res.send("Error in registering the user");**

**console.log(err);**

**}**

**});**

* myPlainTextPassword: The password provided by the user during registration.
* saltRounds: Bcrypt is designed to be slow to increase security. Salt rounds determine how many times the password is hashed before the final output is generated. Higher rounds increase security but also computational cost.
* hash: This is the final hashed password.

## Page:5

**While Logging In:**

js

Copy code

app.post("/login", async (req, res) => {

const { email, pass } = req.body;

try {

const user = await UserModel.find({ email });

if (user.length > 0) {

bcrypt.compare(pass, user[0].pass, function (err, result) {

if (result) {

const token = jwt.sign({ course: 'backend' }, 'zxcvbbmm');

res.send({ "msg": "Login Successful", "token": token });

} else {

res.send("Wrong Credentials");

}

});

} else {

res.send("Wrong Credentials");

}

} catch (err) {

res.send("Something went wrong");

console.log(err);

}

});

* You can also consider using node-argon2, which is similar to bcrypt.

**📂 File Organization**

1. Create two folders: backend & frontend.
2. Move all the backend files into the backend folder.

**📝 Backend: Notes App**

**Routes**

**Notes.route.js**

js

Copy code

const express = require("express");

const { NoteModel } = require("../models/notes.model");

## Page:6

const notesRouter = express.Router();

// Get all notes (authentication required)

notesRouter.get("/", (req, res) => {

// logic to get the notes

});

// Create a note

notesRouter.post("/create", async (req, res) => {

const payload = req.body;

const new\_note = new NoteModel(payload);

await new\_note.save();

res.send({ "msg": "Note Created" });

});

// Update a note

notesRouter.patch("/update/:noteID", (req, res) => {

// logic to update the note

});

// Delete a note

notesRouter.delete("/delete/:noteID", (req, res) => {

// logic to delete the note

});

module.exports = { notesRouter };

**🔐 Middleware for Authentication**

* **middlewares/authenticate.middleware.js**

js

Copy code

const jwt = require("jsonwebtoken");

const authenticate = (req, res, next) => {

const token = req.headers?.authorization?.split(" ")[1];

if (token) {

const decoded = jwt.verify(token, "zxccvb");

if (decoded) {

## Page:7

next();

} else {

res.send("Please Login");

}

} else {

res.send("Please Login");

}

};

module.exports = { authenticate };