1. Helmet — Security Headers

What is it?

`helmet` is a security middleware for Express that adds **protective HTTP headers** to your API responses.

Why do we need it?

Without Helmet, your API is exposed to common web attacks such as:

- **Clickjacking**
- **XSS (Cross-site scripting)**
- **MIME sniffing**
- **Injection attacks**
- **Man-in-the-Middle attacks**

Helmet protects your API by adding security headers automatically.

Example

```
const helmet = require("helmet");
app.use(helmet());
```

What Helmet does internally

Adds headers like:

Header	What it prevents
`X-Frame-Options`	Clickjacking
`X-XSS-Protection`	Cross-site scripting
`X-Content-Type-Options`	MIME sniffing
`Strict-Transport-Security`	Forces HTTPS

2. jsonwebtoken — Token-Based Authentication

What is it?

`jsonwebtoken` (JWT) is a library to:

- Generate tokens when a user logs in
- Verify these tokens on every request

Why do we need it?

Every protected route (leave apply, leave approval, employee view) must check:

- Who is calling the API?
- Is the caller authenticated?
- Does the user have permission?

JWT allows your API to work **statelessly** (no sessions needed).

Example — Sign Token

```
const token = jwt.sign({ id: user.id },
process.env.JWT_SECRET, { expiresIn: "1d" });
```

Example — Verify Token

```
const decoded = jwt.verify(token, process.env.JWT_SECRET);
```

Where it is used?

In your `auth.js` middleware:

```
function auth(req, res, next) {
  const token = req.headers["x-auth-token"];
  if (!token) return res.status(401).json({ message: "No
token provided" });

  try {
    const user = jwt.verify(token, process.env.JWT_SECRET);
    req.user = user;
    next();
  } catch (err) {
    return res.status(400).json({ message: "Invalid
token" });
  }
}
```

3. bcryptjs — Password Hashing

What is it?

`bcryptjs` is used to **hash passwords** before saving them in the database.

Why do we need it?

Storing plain passwords in the database is dangerous.

Hackers could steal them easily.

Bcrypt hashes passwords so even if your DB is leaked, passwords remain safe.

What bcrypt does

- Converts password `"secret123"` → `$2a$10$E9Ty1vF...`
- Cannot be reversed
- Uses salt to prevent rainbow table attacks

Example — Hashing

```
const hash = await bcrypt.hash(password, 10);
```

Example — Comparing

```
const isMatch = await bcrypt.compare(inputPassword,
storedHash);
```

4. winston — Logger for Middleware

What is it?

`winston` is a **professional logging library**.

Why do we need it?

A production-level API must log:

- API calls
- Errors
- Failed login attempts

- Database failures
- Leave approval events

Winston lets you log to:

- Console
- Files
- Database
- Cloud logging systems

Example Logger (logger.js)

```
const { createLogger, transports, format } =
require("winston");

const logger = createLogger({
  level: "info",
  format: format.combine(
    format.timestamp(),
    format.json()
  ),
  transports: [
    new transports.Console(),
    new transports.File({ filename: "logs/app.log" })
  ]
});

module.exports = logger;
```

Using logger in middleware

```
const logger = require("../middleware/logger");

app.use((req, res, next) => {
  logger.info(`${req.method} ${req.url} called`);
  next();
});
```

Summary Table — Easy to Remember

Package	Purpose	Used For	Example

helmet	Security headers	Protect API from attacks	`app.use(helmet())`
jsonwebtoken	JWT authentication	Login, verify user	`jwt.sign()` / `jwt.verify()`
bcryptjs	Password hashing	Secure passwords	`bcrypt.hash()`
winston	Logging	Store logs to file/ console	`logger.info()`