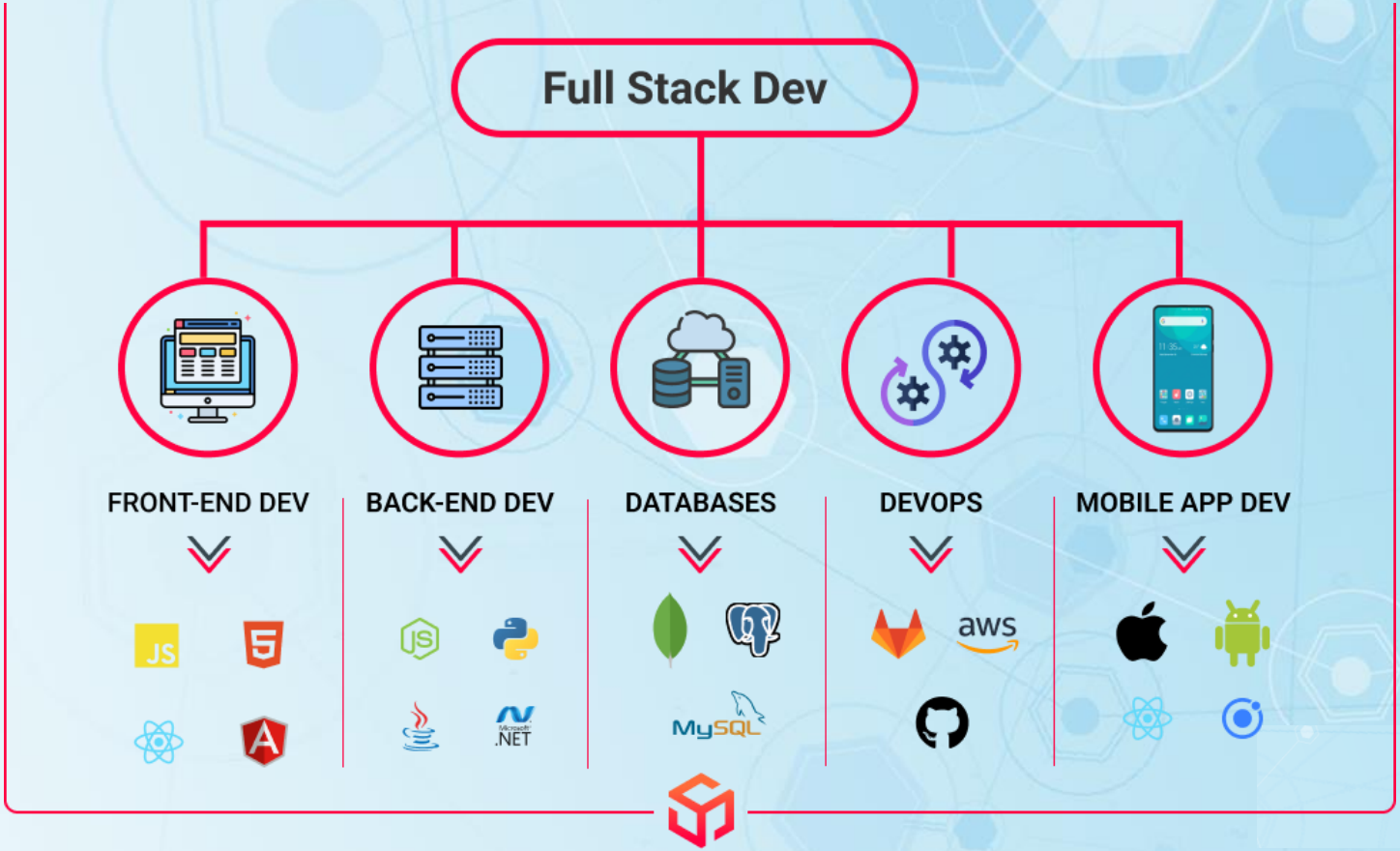
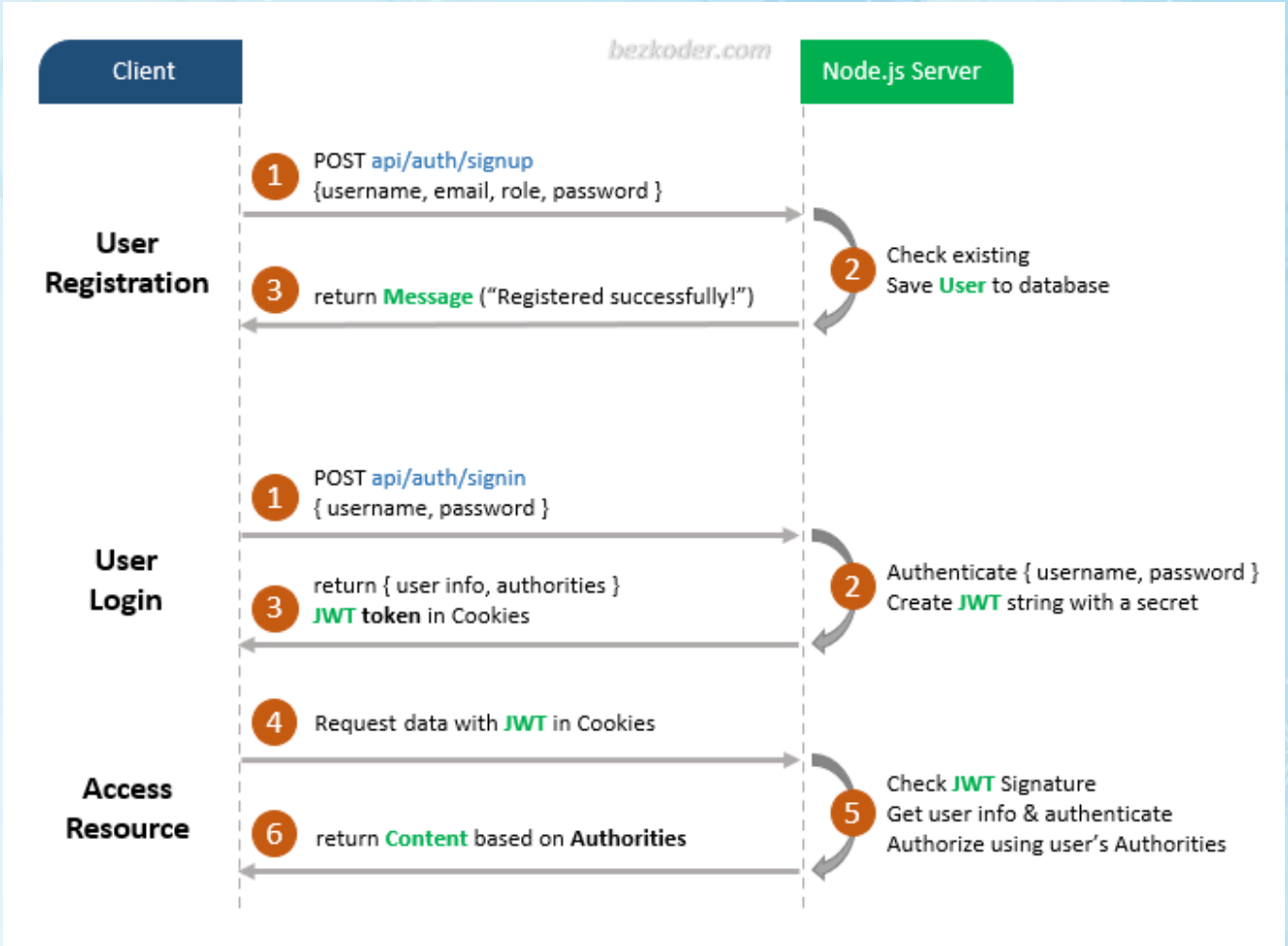
# MC4266 – Full Stack Web Development



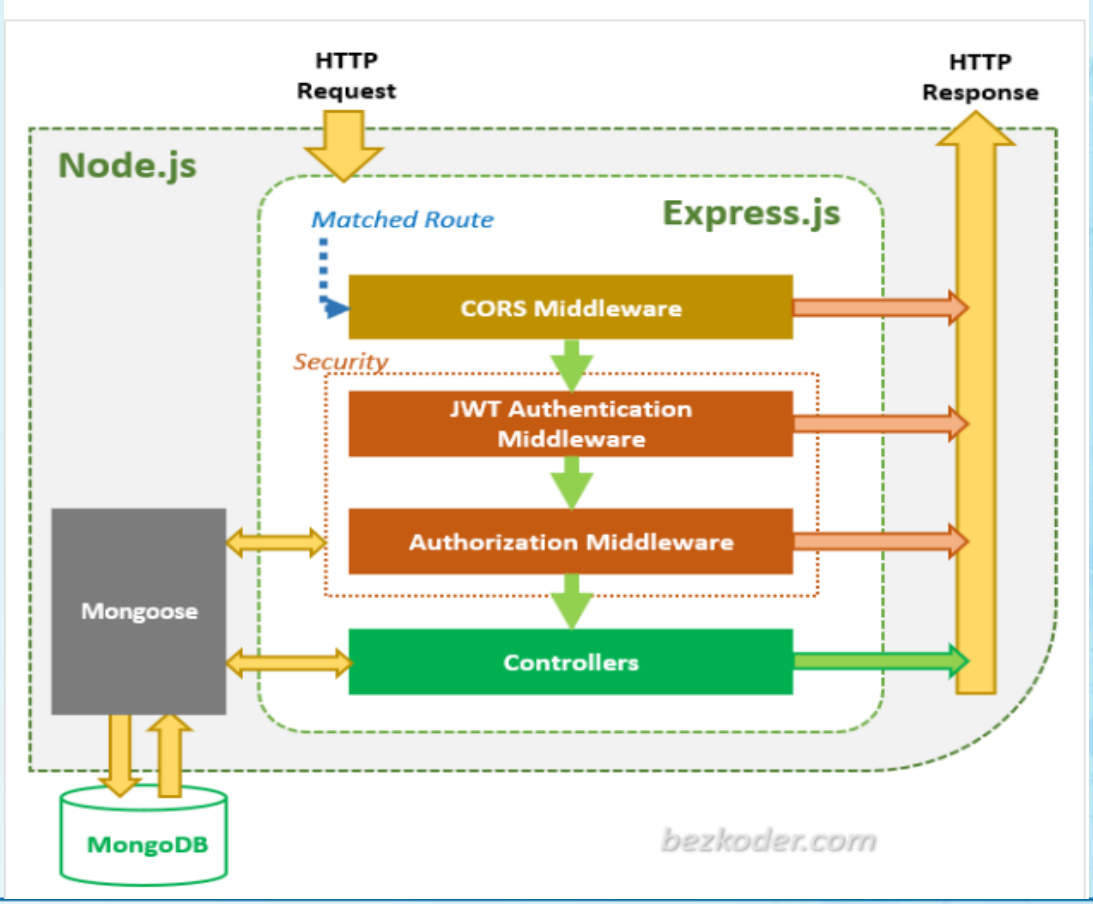
# Ex : 9

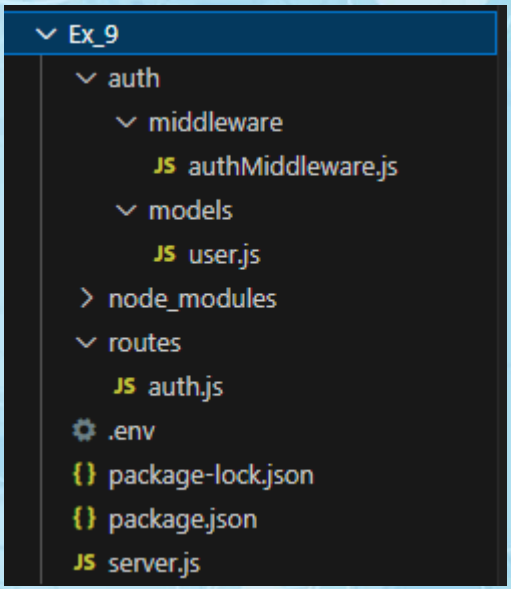
**Ex 9:** Create a simple Sign up and Login mechanism and authenticate the user using cookies. The user information can be stored in either MongoDB or MySQL and the server should be built using NodeJS and Express Framework.

# Ex : 9


Client
bezkoder.com
Node.js Server
User Registration
1 POST api/auth/signup
{username, email, role, password }
2 Check existing
Save User to database
3 return Message ("Registered successfully!")
User Login
1 POST api/auth/signin
{ username, password }
2 Authenticate { username, password }
Create JWT string with a secret
3 return { user info, authorities }
JWT token in Cookies
Access Resource
4 Request data with JWT in Cookies
5 Check JWT Signature
Get user info & authenticate
Authorize using user's Authorities
6 return Content based on Authorities

# Ex : 9


HTTP
Request
HTTP
Response
Node.js
Express.js
Matched Route
CORS Middleware
Security
JWT Authentication
Middleware
Authorization Middleware
Controllers
Mongoose
MongoDB
bezkoder.com


Ex_9
auth
middleware
authMiddleware.js
models
user.js
node_modules
routes
auth.js
.env
package-lock.json
package.json
server.js

# Ex:9

## 1. Create a NodeJS - Directory

D:\FSWD_Lab>mkdir Ex_9

D:\FSWD_Lab>cd Ex_9

## 2. Install NaodeJS

D:\FSWD_Lab> npm init -y

D:\FSWD_Lab\Ex_9>npm install express mongoose cookie-parser bcrypt dotenv

## 3. Create ex_9_server.js in the root directory

Ex_9/ex_9_server.js

# Ex:9

## 4. Create .env

PORT=5000

MONGO_URI=mongodb://localhost:27017/Ex9db

JWTAUTHSECRET=your_secret_key

## 5. Create models/user.js

## 6. Create routes/user.js

Set path/url

# Ex:9

7. Create server.js

8. Create controller/user.js

For SignUp / Login / Logout / getAllUsers

9. SignUp using postman http://localhost:2222/auth/signUp

```
{
  "name": "Mani",
  "mobile":"7896541230",
  "email": "mani@gmail.com",
  "password": "12345"
}
```

# Ex:9

**10.Login using Postman : http://localhost:2222/auth/login**

```
{
  "email": “mani@gmail.com",
  "password": "12345"
}
```

**11.Getallusers using Postman : http://localhost:2222/auth/getAllUsers**

**Add  Cookie Token in Header Value**

# Ex:9 Server.js - Explanation

- import express, cookie-session and cors modules:
- Express is for building the Rest apis
- cookie-session helps to stores the session data on the client within a cookie without requiring any database/resources on the server side
- cors provides Express middleware to enable CORS
- create an Express app, then add request parsing, cookie-based session middleware and cors middlewares using app.use() method. Notice that we set origin: http://localhost:8081.
- define a GET route which is simple for test.
- listen on port 8080 for incoming requests.

# Ex:9 Server.js - Explanation

- keys: sign & verify cookie values. Set cookies are always signed with keys[0], while the other keys are valid for verification, allowing for key rotation.
- httpOnly: indicate that the cookie is only to be sent over HTTP(S), and not made available to client JavaScript.

# Ex:9 - CheckList

**1. Check Server Running**

node server.js

**2. Check URL & Method in Postman or Browser**

**http://localhost:5000/api/signup**

**3. Check routing in server**

```
const authRoutes = require('./routes/api');
app.use('/api', authRoutes);
```

# Ex:9 - CheckList

**4. Check Body parsing Enabled**

```
app.use(express.json());
```

**5. Verify route path** routes/auth.js

```
router.post('/signup', ...)  // full path is /api/signup

router.post('/login', ...)   // full path is /api/login

router.get('/getAllUsers', ...) // full path is /api/getAllUsers
```

# Ex:9 – To Do Activity

## Create a Front End using

## React JS

# Any Queries?

*manirajk.mca@srmvalliammai.ac.in*