

TP-08

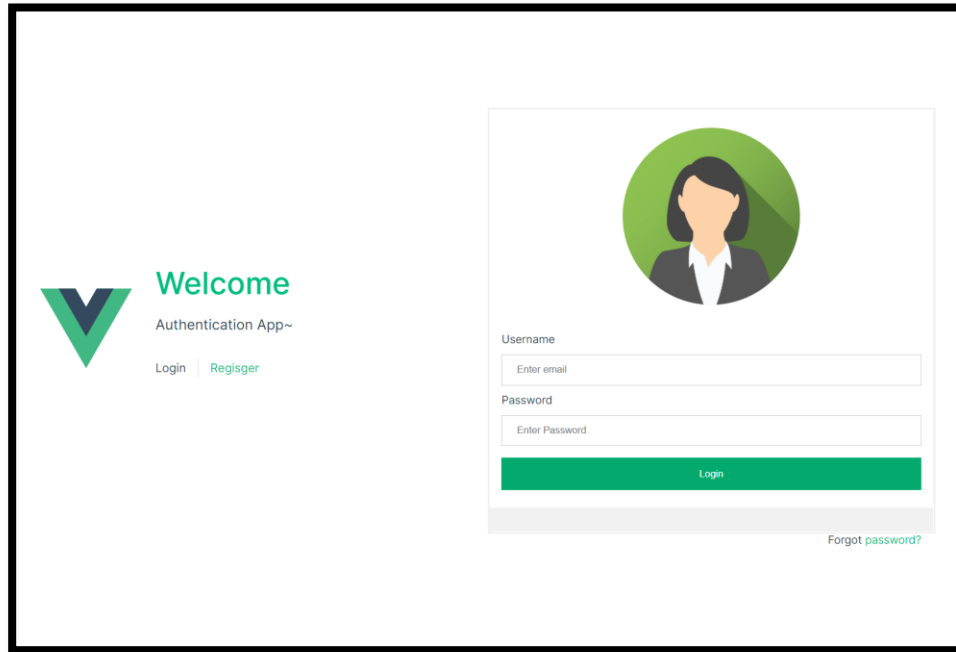
# **VueJS, NodeJS**

(Authentication frontend and APIs)

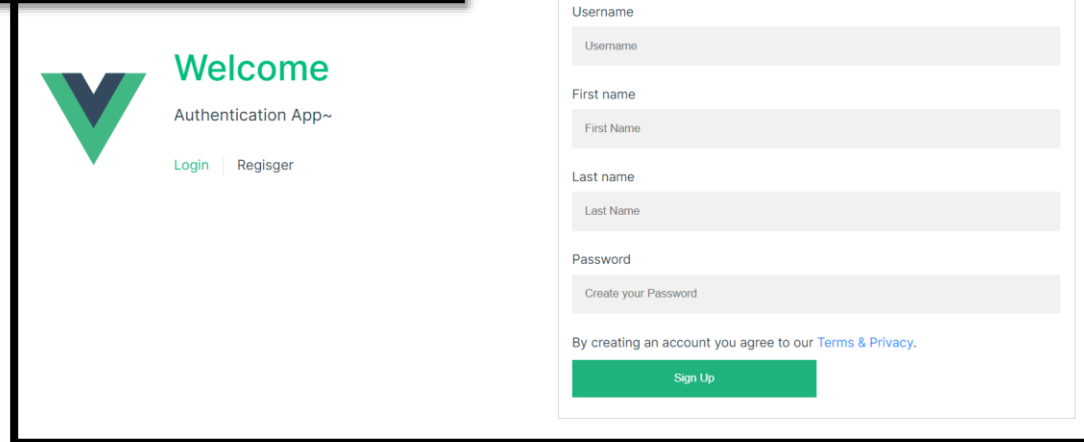
# VueJS

## EX1: Create a simple login and register form in VueJS

- By using VueJS router, we can access by
  - Login: <http://localhost:3000/>
  - Register: <http://localhost:3000/register>
  - Home: <http://localhost:3000/home>



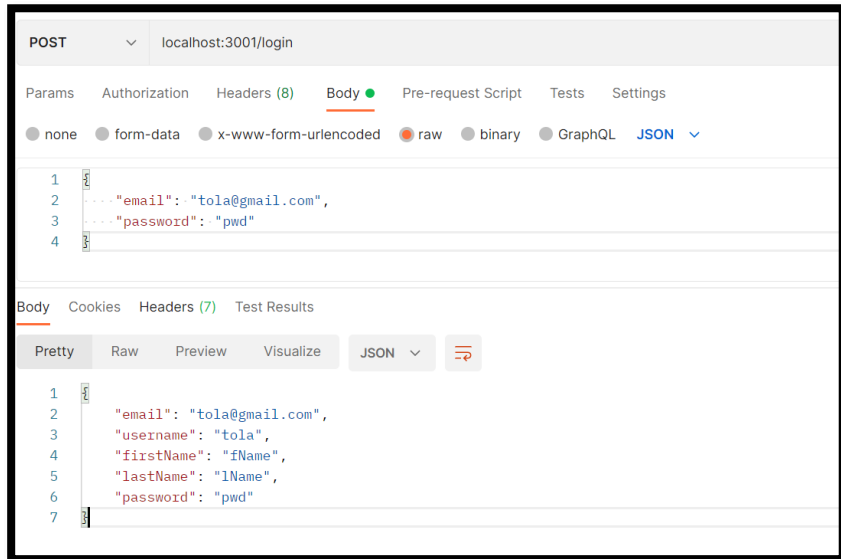
A login form mockup with a white background. On the left, there is a green and blue 'V' logo, the text 'Welcome' in green, 'Authentication App~' in gray, and links for 'Login' and 'Register' in green. On the right, there is a circular profile picture of a woman. Below it are input fields for 'Username' (placeholder 'Enter email') and 'Password' (placeholder 'Enter Password'). A green 'Login' button is at the bottom. A link 'Forgot password?' is at the bottom right.



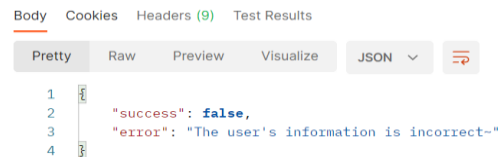
A sign up form mockup with a white background. On the left, there is a green and blue 'V' logo, the text 'Welcome' in green, 'Authentication App~' in gray, and links for 'Login' and 'Register' in green. On the right, there is a 'Sign Up' heading, followed by the text 'Please fill in this form to create an account.' Below this are input fields for 'Email' (placeholder 'Enter Email'), 'Username' (placeholder 'Username'), 'First name' (placeholder 'First Name'), 'Last name' (placeholder 'Last Name'), and 'Password' (placeholder 'Create your Password'). At the bottom, there is a line of text 'By creating an account you agree to our [Terms & Privacy](#).' and a green 'Sign Up' button.

# NodeJS with ExpressJS

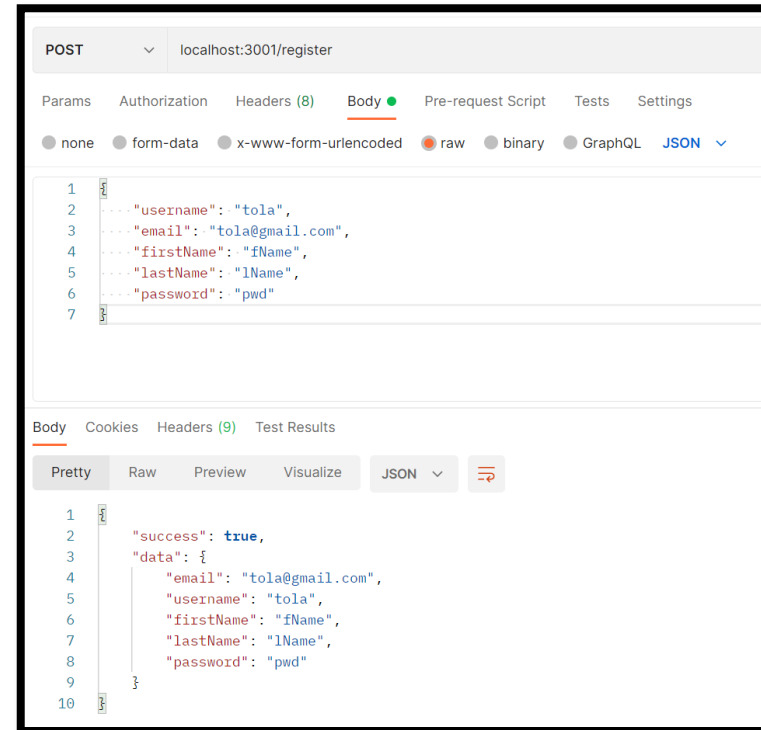
EX2: Create a nodejs APIs for authentication with ExpressJS. Use Json file as a database to store user's information.



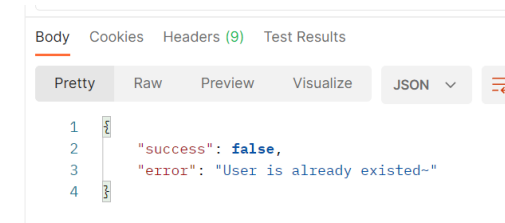
```
POST localhost:3001/login
Body
[{"email": "tola@gmail.com", "password": "pwd"}]
```



```
Body
[{"success": false, "error": "The user's information is incorrect-"}]
```



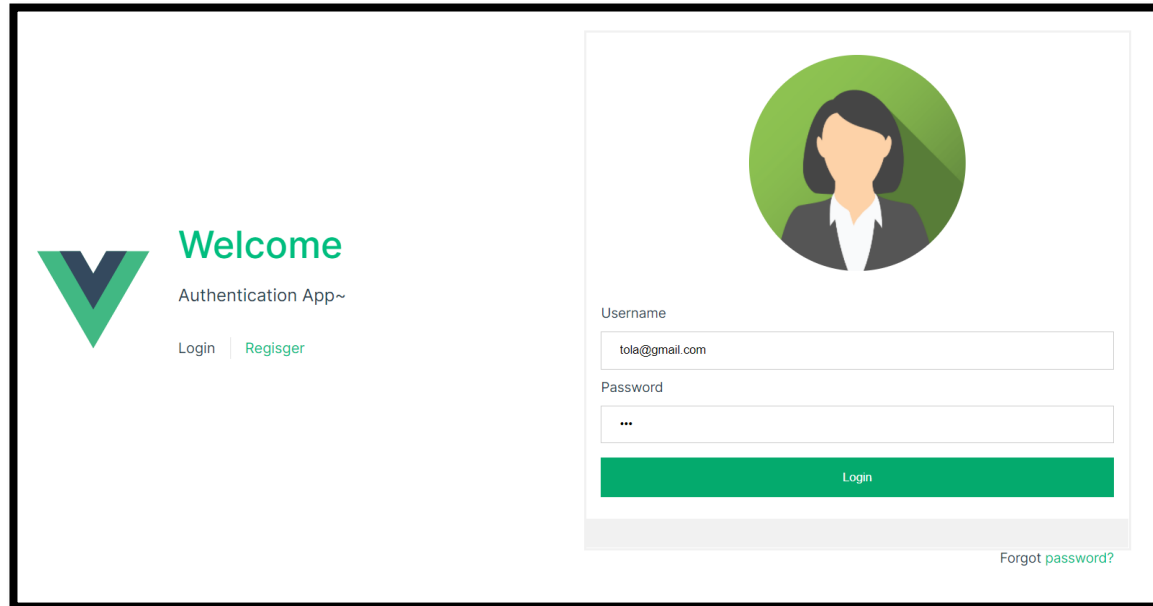
```
POST localhost:3001/register
Body
[{"username": "tola", "email": "tola@gmail.com", "firstName": "fName", "lastName": "lName", "password": "pwd"}]
```



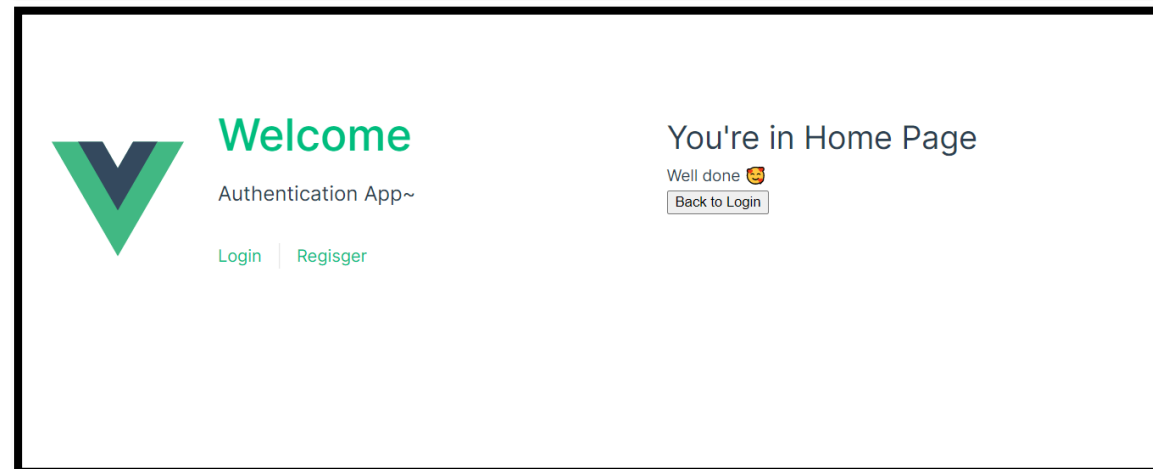
```
Body
[{"success": false, "error": "User is already existed-"}]
```

# NodeJS

## EX3: Integrate Login/Register app with APIs



A login form mockup with a green and blue logo on the left. The logo is a stylized 'V' shape. To the right of the logo, the text 'Welcome' is in green, and 'Authentication App~' is in black. Below this, there are two links: 'Login' and 'Regisger' (misspelled). The main form area has a circular profile picture placeholder with a green background and a person icon. Below the profile picture, there are two input fields: 'Username' with the value 'tola@gmail.com' and 'Password' with three dots indicating a masked password. A green 'Login' button is below the password field. At the bottom right, there is a link 'Forgot password?'.

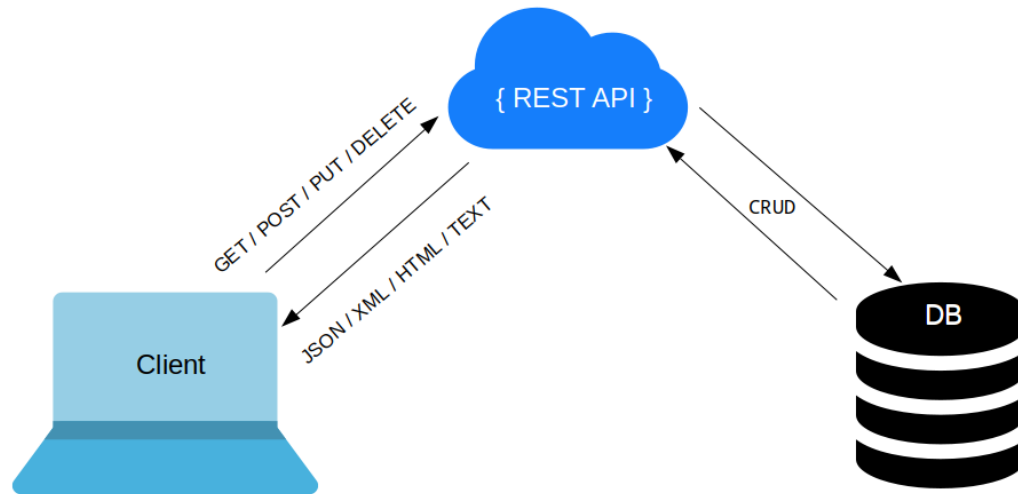


A home page mockup with the same green and blue logo on the left. The logo is a stylized 'V' shape. To the right of the logo, the text 'Welcome' is in green, and 'Authentication App~' is in black. Below this, there are two links: 'Login' and 'Regisger' (misspelled). The main content area has the text 'You're in Home Page' in black. Below this, there is a message 'Well done 🎉' and a button 'Back to Login'.

**Getting to understand**

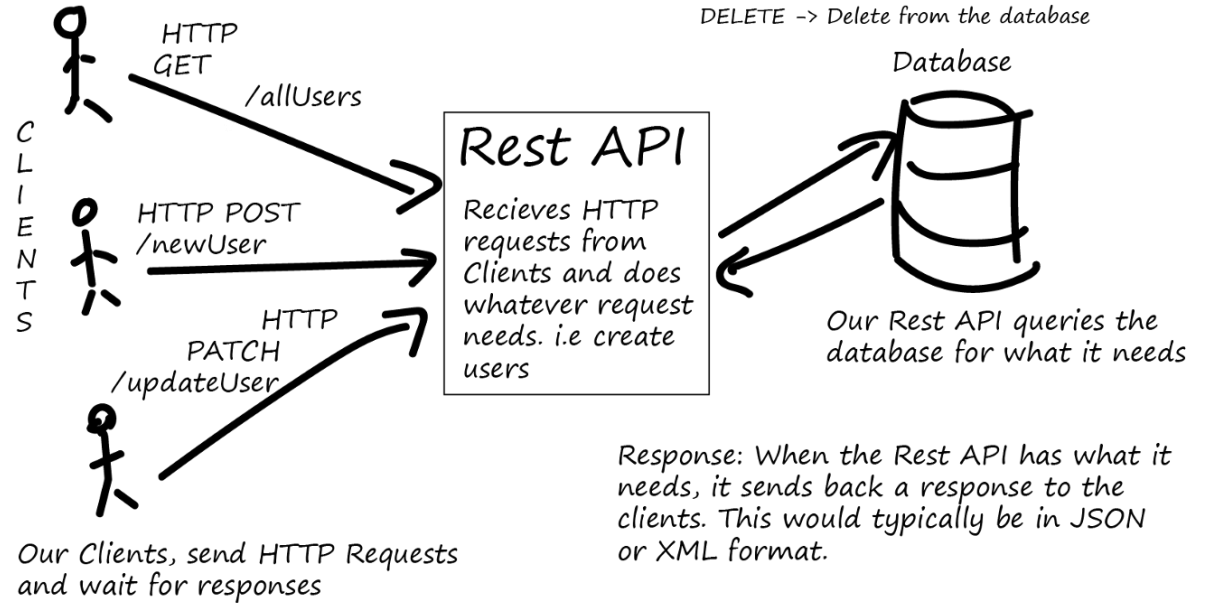
**“ExpressJS”**

# Rest APIs



## Rest API Basics

Typical HTTP Verbs:  
GET -> Read from Database  
PUT -> Update/Replace row in Database  
PATCH -> Update/Modify row in Database  
POST -> Create a new record in the database  
DELETE -> Delete from the database



# ExpressJS

```
const express = require('express' 4.17.3 )
const app = express()
const port = 3000

app.get('/', (req, res) => {
  res.send('Hello World!')
})

app.listen(port, () => {
  console.log(`Example app listening on port ${port}`);
})
```

```
$ node app.js
```

# ExpressJS

## Basic routing

```
app.get('/', (req, res) => {  
  res.send('Hello World!')  
})
```

Respond to POST request on the root route (/), the application's home page:

```
app.post('/', (req, res) => {  
  res.send('Got a POST request')  
})
```

Respond to a PUT request to the /user route:

```
app.put('/user', (req, res) => {  
  res.send('Got a PUT request at /user')  
})
```

Respond to a DELETE request to the /user route:

```
app.delete('/user', (req, res) => {  
  res.send('Got a DELETE request at /user')  
})
```



# ExpressJS

## Serving static files in Express

```
express.static(root, [options])
```

```
app.use(express.static('public'))
```

```
http://localhost:3000/images/kitten.jpg  
http://localhost:3000/css/style.css  
http://localhost:3000/js/app.js  
http://localhost:3000/images/bg.png  
http://localhost:3000/hello.html
```

```
const path = require('path')  
app.use('/static', express.static(path.join(__dirname, 'public')))
```

```
http://localhost:3000/static/images/kitten.jpg  
http://localhost:3000/static/css/style.css  
http://localhost:3000/static/js/app.js  
http://localhost:3000/static/images/bg.png  
http://localhost:3000/static/hello.html
```

# ExpressJS

## Middleware

```
var express = require('express');
var app = express();
```

HTTP method for which the middleware function applies.

Path (route) for which the middleware function applies.

The middleware function.

```
app.get('/', function(req, res, next) {
  next();
})
```

Callback argument to the middleware function, called "next" by convention.

HTTP response argument to the middleware function, called "res" by convention.

HTTP request argument to the middleware function, called "req" by convention.

```
app.listen(3000);
```

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

const myLogger = function (req, res, next) {
  console.log('LOGGED')
  next()
}

app.use(myLogger)

app.get('/', (req, res) => {
  res.send('Hello World!')
})

app.listen(3000)
```

```
const express = require('express')
const cookieParser = require('cookie-parser')
const cookieValidator = require('./cookieValidator')

const app = express()

async function validateCookies (req, res, next) {
  await cookieValidator(req.cookies)
  next()
}

app.use(cookieParser())

app.use(validateCookies)

// error handler
app.use((err, req, res, next) => {
  res.status(400).send(err.message)
})

app.listen(3000)
```

# ExpressJS

## Application-level middleware

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

app.use((req, res, next) => {
  console.log('Time:', Date.now())
  next()
})
```

```
app.use('/user/:id', (req, res, next) => {
  console.log('Request URL:', req.originalUrl)
  next()
}, (req, res, next) => {
  console.log('Request Type:', req.method)
  next()
})
```

```
function logOriginalUrl (req, res, next) {
  console.log('Request URL:', req.originalUrl)
  next()
}
```

```
function logMethod (req, res, next) {
  console.log('Request Type:', req.method)
  next()
}
```

```
const logStuff = [logOriginalUrl, logMethod]
app.get('/user/:id', logStuff, (req, res, next) => {
  res.send('User Info')
})
```

# ExpressJS

## Router-level middleware

```
const router = express.Router()
```

```
const express = require('express')
const app = express()
const router = express.Router()

// a middleware function with no mount path. This code is executed for every request to the router
router.use((req, res, next) => {
  console.log('Time:', Date.now())
  next()
})

// a middleware sub-stack shows request info for any type of HTTP request to the /user/:id path
router.use('/user/:id', (req, res, next) => {
  console.log('Request URL:', req.originalUrl)
  next()
}, (req, res, next) => {
  console.log('Request Type:', req.method)
  next()
})
```

## Built-in middleware

- `express.static` serves static assets such as HTML files, images, and so on.
- `express.json` parses incoming requests with JSON payloads. **NOTE: Available with Express 4.16.0+**
- `express.urlencoded` parses incoming requests with URL-encoded payloads. **NOTE: Available with Express 4.16.0+**

## Third-party middleware

```
const express = require('express')
const app = express()
const cookieParser = require('cookie-parser')

// load the cookie-parsing middleware
app.use(cookieParser())
```

## I want more about ExpressJS

 <https://expressjs.com/>

Good luck 🍀