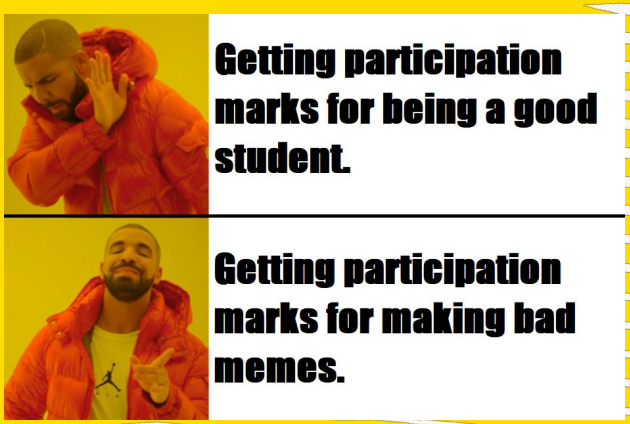
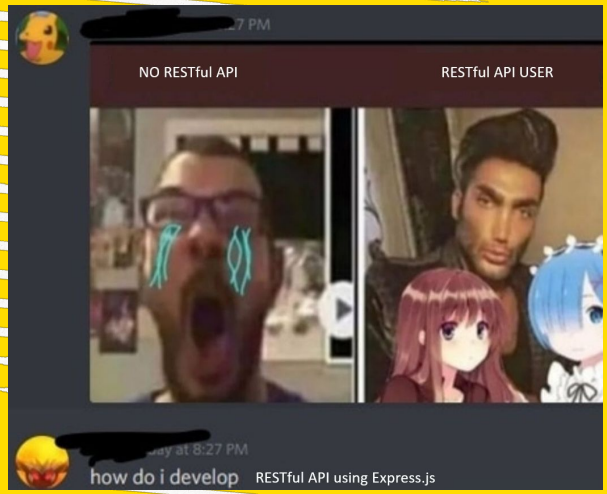


COMP1531 | T09B / H17B

Week 5

william.huynh3@unsw.edu.au



UNSW SYDNEY

Learning objectives

- Tips for Iteration 2
- Understanding Web Servers & HTTP
- Mini Iteration 2
- Testing our Backend with an API Client
- Writing HTTP Tests

UNSW SYDNEY

Iteration 2 Tasks

29 Functions (11 are old, 18 are new)

Major Tasks (50%)

TypeScript your project

Setup Continuous Integration *(pipeline)*

Write new HTTP Tests

Implement new functions

Write the API (server.ts)

Setup persistence

Power the frontend

Linting your code

Minor Tasks (50%)

Code quality (30%)

Git Practices (20%)

UNSW SYDNEY

Iteration 2 Tasks

Major Tasks (50%)

29 Functions (11 are old, 18 are new)

Write new HTTP Tests

Implement new functions

Write the API (server.ts)

Minor Tasks (50%)

Code quality (30%)

Git Practices (20%)

UNSW SYDNEY

Web Servers: What are they?

Inputs **(Requests)**

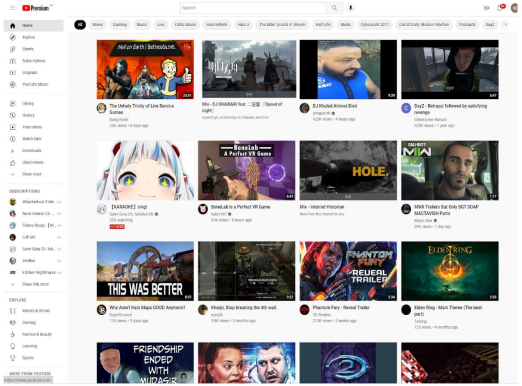
"cat pictures"

Client

Server

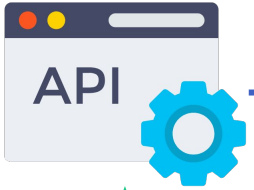
Outputs **(Responses)**

Client



Inputs (Requests)

API




```
app.post('/auth/register/v2', (req: Request, res: Response) => {
  const { email, password, nameFirst, nameLast } = req.body;
  res.json(authRegisterV2(email, password, nameFirst, nameLast));
});
app.get('/channels/list/v2', (req: Request, res: Response) => {
  const token = req.query.token as string;
  res.json(channelsListV2(token));
});
```

Outputs (Responses)

Server

```
function authRegisterV2(email:String, nameFirst: String,
  return {authUserId: 1};
}

function authLoginV2(email:String, password: String) {
  return {authUserId: 1};
}

function authLogoutV2(token:String) {
  return {};
}

function userProfileV2(token:String, uId: number) {
  return {};
}
```

UNSW SYDNEY

Frontend Website:

https://treats-unsw.herokuapp.com/

Express: Write an API & create a HTTP Server!

A very popular npm library that allows you to
run your own HTTP server and write your own API
using NodeJS through the use of HTTP Methods.

```
npm install --save-dev express
```

Express contributes to 50% of Iteration 2

Express Basics: HTTP Methods

GET

PUT

POST

DELETE

Express Basics: HTTP Methods

GET	Reading data from the `dataStore`
PUT	Editing data in the `dataStore`
POST	Adding data to the `dataStore`
DELETE	Deleting data from the `dataStore`

GET PUT POST DELETE ??

authRegisterV2

GET PUT POST DELETE ??

channelMessagesV2

GET PUT POST DELETE ??

channelsCreateV2

GET PUT POST DELETE ??

clearV1

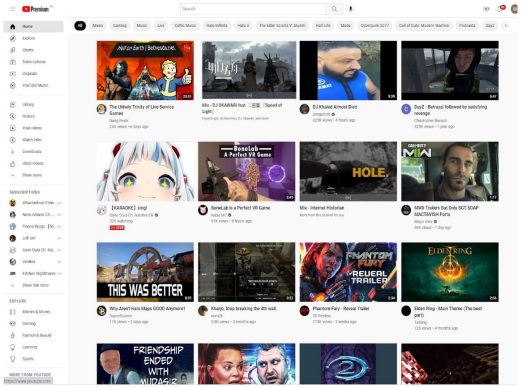
GET PUT POST DELETE ??

channelDetailsV2

GET PUT POST DELETE ??

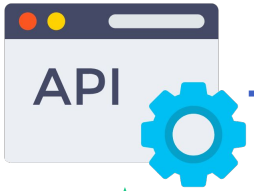
messageEditV1

Client



Inputs (Requests)

API




```
app.post('/auth/register/v2', (req: Request, res: Response) => {
  const { email, password, nameFirst, nameLast } = req.body;
  res.json(authRegisterV2(email, password, nameFirst, nameLast));
});
app.get('/channels/list/v2', (req: Request, res: Response) => {
  const token = req.query.token as string;
  res.json(channelsListV2(token));
});
```

Outputs (Responses)

Server

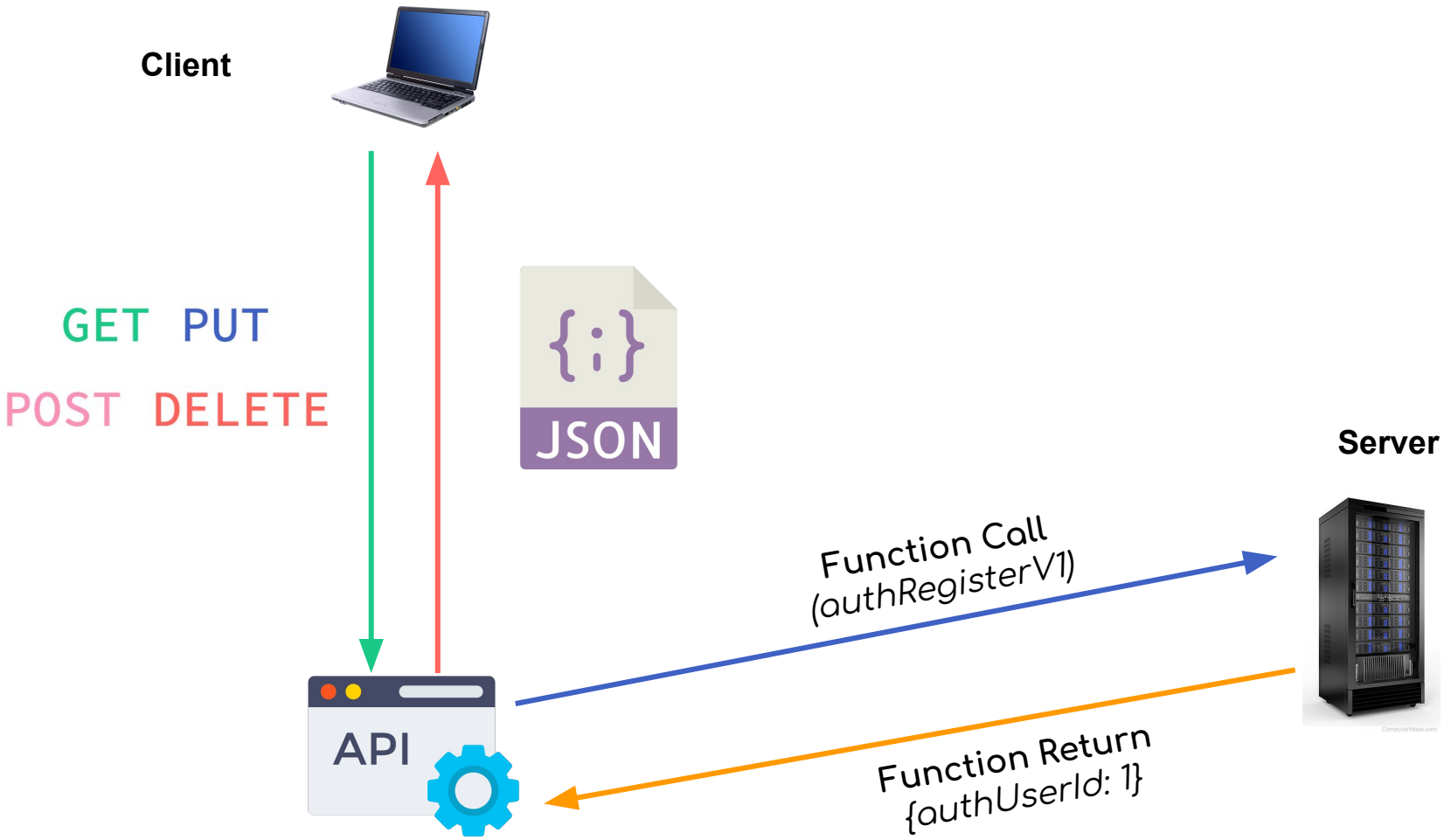
```
function authRegisterV2(email:String, nameFirst: String,
  return {authUserId: 1};
}

function authLoginV2(email:String, password: String) {
  return {authUserId: 1};
}

function authLogoutV2(token:String) {
  return {};
}

function userProfileV2(token:String, uId: number) {
  return {};
}
```

UNSW SYDNEY


Client
GET PUT
POST DELETE
JSON
Server
Function Call
(authRegisterV1)
API
Function Return
{authUserId: 1}

Tute Task: Mini Iteration 2

1. Writing the API

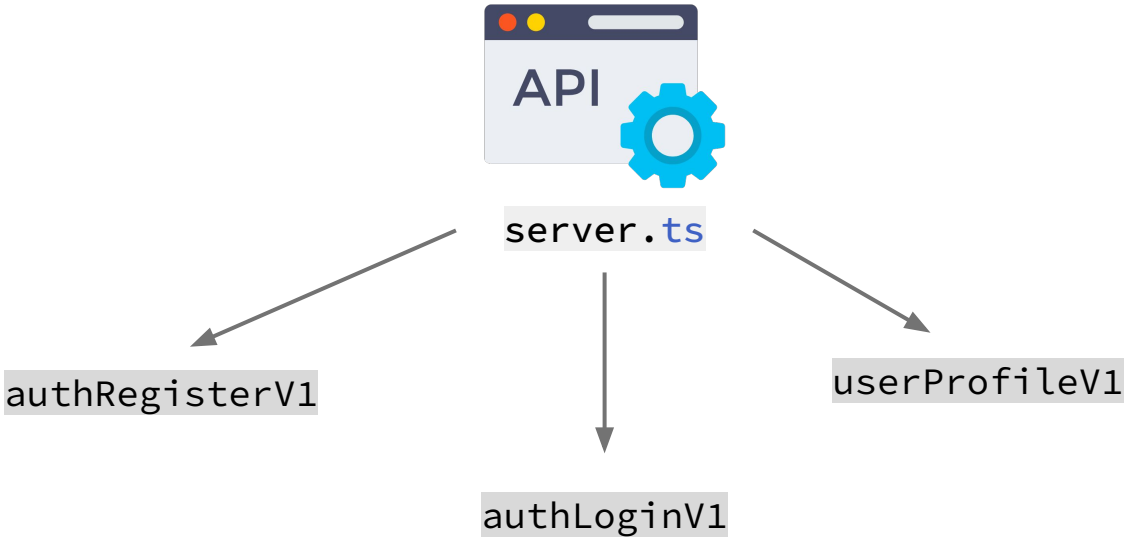
2. Using an API Client to test our server

3. Writing HTTP Tests

API: What the hell is an API?

An API consists of a series of routes, where each route connects to a function.

Each route is stored in a file called `server.ts`



API: What the hell is a Route?

```
40 app.post('/auth/register/v2', (req: Request, res: Response) => {
41     const { email, password, nameFirst, nameLast } = req.body;
42     res.json(authRegisterV2(email, password, nameFirst, nameLast));
43 });
```

- HTTP Method
- Route
- Request/Response *(req & res)*

API: What the hell is a Route?

HTTP Method

```
40 app.post('/auth/register/v2', (req: Request, res: Response) => {
41     const { email, password, nameFirst, nameLast } = req.body;
42     res.json(authRegisterV2(email, password, nameFirst, nameLast));
43 });
```

API: What the hell is a Route?

Route

```
40 app.post('/auth/register/v2', (req: Request, res: Response) => {
41     const { email, password, nameFirst, nameLast } = req.body;
42     res.json(authRegisterV2(email, password, nameFirst, nameLast));
43 });
```

API: What the hell is a Route?

```
40 app.post('/auth/register/v2', (req: Request, res: Response) => {
41     const { email, password, nameFirst, nameLast } = req.body;
42     res.json(authRegisterV2(email, password, nameFirst, nameLast));
43 });
```


Request: The data that the CLIENT sends to the API

Response: The data that the API sends back to the CLIENT

Express Basics: How does Express communicate?

For inputs:

Request Query	Typically **strings**	GET and DELETE methods
Request Body	Typically **objects**	PUT and POST methods

```
40 app.post('/auth/register/v2', (req: Request, res: Response) => {
41     const { email, password, nameFirst, nameLast } = req.body;
42     res.json(authRegisterV2(email, password, nameFirst, nameLast));
43 });
```

```
44 app.get('/channels/list/v2', (req: Request, res: Response) => {
45     const token = req.query.token as string;
46     res.json(channelsListV2(token));
47 });
```

Activity

auth/login/v2

auth/register/v2

channels/create/v2

channel/join/v2

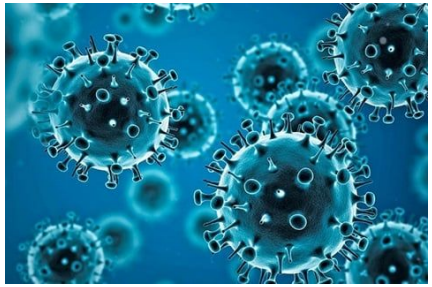
channel/invite/v2

user/profile/v2

Word on the street is that Iteration 2 is absolutely disgusting, and has caused the contraction of ligma 🦠 by numerous 1531 students.

Fearing for your life, you decide to start early and write the API routes with your group!

Can you write the following routes in your `server.ts` file before you and your group contracts ligma???



Tute Task: Mini Iteration 2

1. ~~Writing the API~~

2. Using an API Client to test our server

3. Writing HTTP Tests

Testing our API: API Clients

An API client allows you to call your API routes, and view the responses through a **Graphical User Interface** (GUI).

It is an excellent debugging tool when writing your API's

POSTMAN

Insomnia
REST API Client

Testing our API: Status Codes

200

400

404

500

Testing our API: Status Codes

200 – All good!

400 – Bad Request

404 – Route Not Found

500 – Server Error

Tute Task: Mini Iteration 2

1. ~~Writing the API~~

2. ~~Using an API Client to test our server~~

3. Writing HTTP Tests

Testing: Iteration 1 Tests

```
test('Test sucessful authRegister 1', () => {
  const checkAuthUserID1 = authRegisterV1('morganau@example.com', 'morgan123456', 'Morgan', 'Au');
  expect(checkAuthUserID1).toStrictEqual({ authUserId: checkAuthUserID1.authUserId });
});

test('Test sucessful authRegister 2', () => {
  const checkAuthUserID2 = authRegisterV1('chosun@education.com', 'handsomechosun', 'Cho', 'Sun');
  expect(checkAuthUserID2).toStrictEqual({ authUserId: checkAuthUserID2.authUserId });
});
```

Testing: Iteration 2 HTTP Tests

```
23  test('succesful authRegister test', () => {
24    const res = request(
25      'POST',
26          `${url}:${port}/auth/register/v2`,
27          {
28            body: JSON.stringify({
29              email: 'john@email.com',
30              password: 'password',
31              nameFirst: 'john',
32              nameLast: 'bob'
33            }),
34
35          }
36    );
37    const body1 = JSON.parse(res.body as string);
38    expect(res.statusCode).toBe(OK);
39    expect(body1).toEqual(expect.objectContaining({
40      token: expect.any(String),
41      authUserId: expect.any(Number),
42    }));
```

Testing: Iteration 2 HTTP Tests

1. Making a request
2. Converting the response to a JavaScript object.
3. Doing our normal `expect` & `toEqual` stuff

```
23 test('succesful authRegister test', () => {
24   const res = request(
25     'POST',
26         `${url}:${port}/auth/register/v2`,
27         {
28           body: JSON.stringify({
29             email: 'john@email.com',
30             password: 'password',
31             nameFirst: 'john',
32             nameLast: 'bob'
33           }),
34
35         }
36   );
37   const body1 = JSON.parse(res.body as string);
38   expect(res.statusCode).toBe(OK);
39   expect(body1).toEqual(expect.objectContaining({
40     token: expect.any(String),
41     authUserId: expect.any(Number),
42   }));
```

Step 1: lines 24 - 36
Step 2: line 37
Step 3: lines 38 - 42

1. Making a request
2. Converting the response to a JavaScript object.
3. Doing our normal `expect` & `toEqual` stuff

Step 1: lines 24 - 36
Step 2: line 37
Step 3: lines 38 - 42

```
23 test('succesful authRegister test', () => {
24   const res = request(
25     'POST',
26         `${url}:${port}/auth/register/v2`,
27         {
28           body: JSON.stringify({
29             email: 'john@email.com',
30             password: 'password',
31             nameFirst: 'john',
32             nameLast: 'bob'
33           }),
34
35         }
36   );
37   const body1 = JSON.parse(res.body as string);
38   expect(res.statusCode).toBe(OK);
39   expect(body1).toEqual(expect.objectContaining({
40     token: expect.any(String),
41     authUserId: expect.any(Number),
42   }));
```

Activity

Word on the street is that Iteration 2 is absolutely disgusting, and has caused the contraction of ligma 🦠 by numerous 1531 students.

Fearing for your life, you decide to start early and write the HTTP Tests for authLogin & authRegister

Can you write the following tests in your `auth_http.ts` file before you and your group contracts ligma???