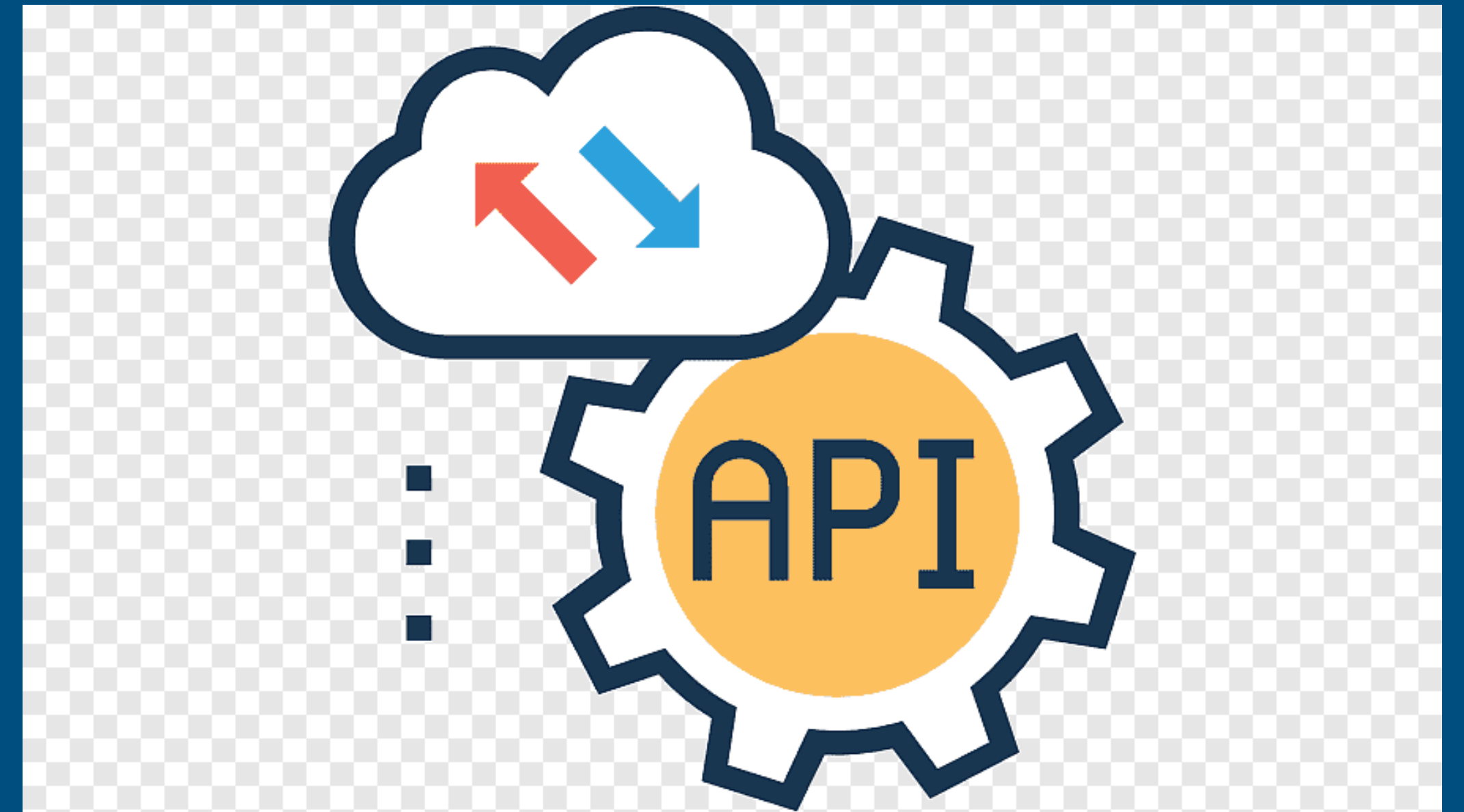


Playtime Rest



Full Stack Web Development

TDD



**Do The Simplest Thing That
Could Possibly Work**

- (if you're not sure what to do yet)

- A key practice in Agile Test Driven Development
- Start Simple
- Iterate incrementally
- Small steps

<https://wiki.c2.com/?DoTheSimplestThingThatCouldPossiblyWork>

Playlist API

- First attempt does nothing!
- Simple placeholders for API endpoints

playlist-api.js

```
import Boom from "@hapi/boom";
import { db } from "../models/db.js";

export const playlistApi = {
  find: {
    auth: false,
    handler: async function (request, h) {
    },
  },

  findOne: {
    auth: false,
    async handler(request) {
    },
  },

  create: {
    auth: false,
    handler: async function (request, h) {
    },
  },

  deleteOne: {
    auth: false,
    handler: async function (request, h) {
    },
  },

  deleteAll: {
    auth: false,
    handler: async function (request, h) {
    },
  },
};
```

api-routes.js

- Routes to expose the api endpoints

```
import { playlistApi } from "../api/playlist-api.js";
...

export const apiRoutes = [
  ...
  { method: "POST", path: "/api/playlists", config: playlistApi.create },
  { method: "DELETE", path: "/api/playlists", config: playlistApi.deleteAll },
  { method: "GET", path: "/api/playlists", config: playlistApi.find },
  { method: "GET", path: "/api/playlists/{id}", config: playlistApi.findOne },
  { method: "DELETE", path: "/api/playlists/{id}", config: playlistApi.deleteOne },
];
```

- playtimeService extended to connect to new endpoints

playtime-service.js

```
import axios from "axios";

import { serviceUrl } from "../fixtures.js";

export const playtimeService = {
  playtimeUrl: serviceUrl,

  async createPlaylist(playlist) {
    const res = await axios.post(`${this.playtimeUrl}/api/playlists`, playlist);
    return res.data;
  },

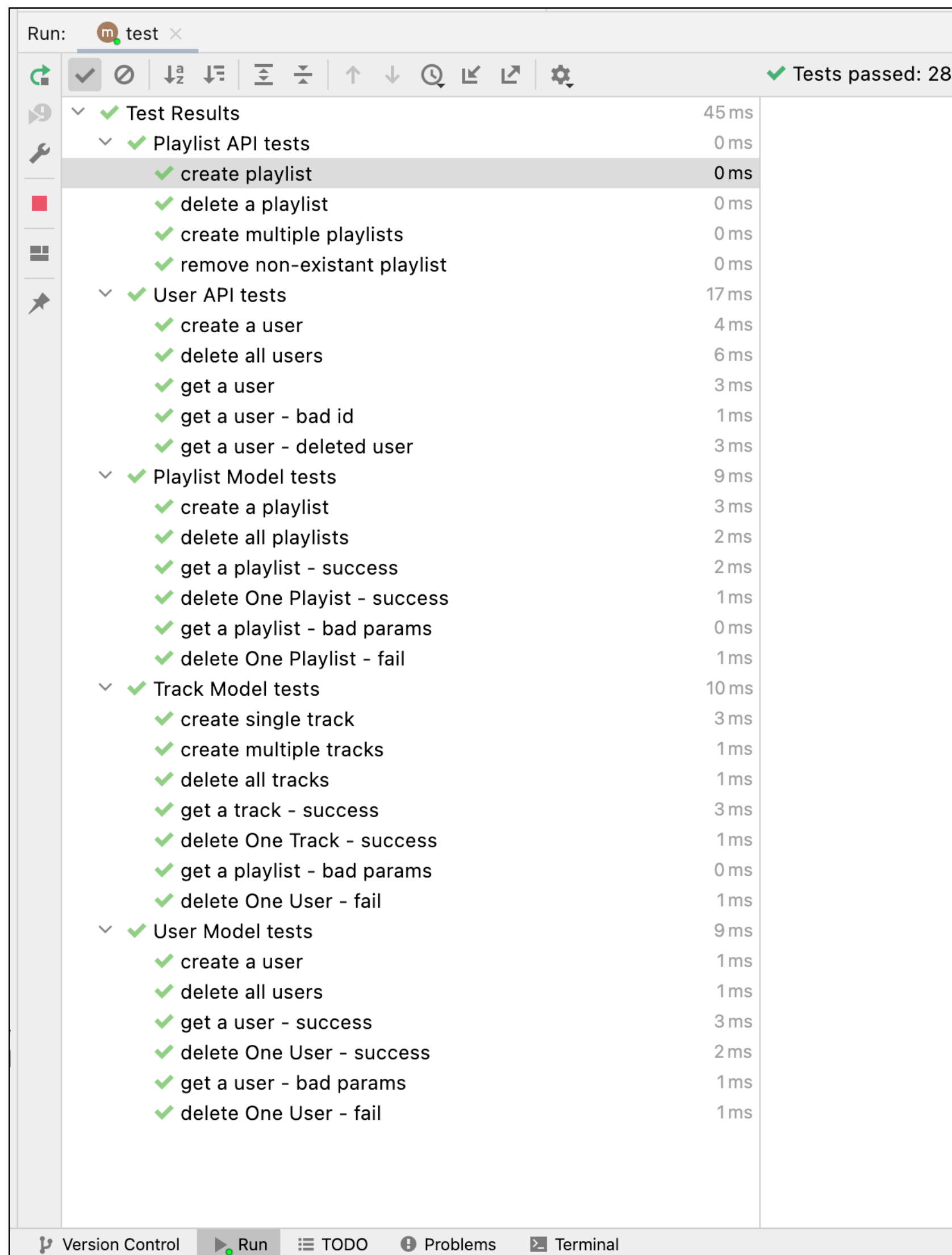
  async deleteAllPlaylists() {
    const response = await axios.delete(`${this.playtimeUrl}/api/playlists`);
    return response.data;
  },

  async deletePlaylist(id) {
    const response = await axios.delete(`${this.playtimeUrl}/api/playlists/${id}`);
    return response;
  },

  async getAllPlaylists() {
    const res = await axios.get(`${this.playtimeUrl}/api/playlists`);
    return res.data;
  },

  async getPlaylist(id) {
    const res = await axios.get(`${this.playtimeUrl}/api/playlists/${id}`);
    return res.data;
  },
};
```

The Tests are 'stubbed' out initially



The screenshot shows the 'Run: test' window in VS Code. The top bar indicates 'Tests passed: 28'. The test results are organized into a tree structure with the following categories and items:

Category	Item	Duration
Test Results		45 ms
Playlist API tests		0 ms
	create playlist	0 ms
	delete a playlist	0 ms
	create multiple playlists	0 ms
	remove non-existent playlist	0 ms
User API tests		17 ms
	create a user	4 ms
	delete all users	6 ms
	get a user	3 ms
	get a user - bad id	1 ms
	get a user - deleted user	3 ms
Playlist Model tests		9 ms
	create a playlist	3 ms
	delete all playlists	2 ms
	get a playlist - success	2 ms
	delete One Playlist - success	1 ms
	get a playlist - bad params	0 ms
	delete One Playlist - fail	1 ms
Track Model tests		10 ms
	create single track	3 ms
	create multiple tracks	1 ms
	delete all tracks	1 ms
	get a track - success	3 ms
	delete One Track - success	1 ms
	get a playlist - bad params	0 ms
	delete One User - fail	1 ms
User Model tests		9 ms
	create a user	1 ms
	delete all users	1 ms
	get a user - success	3 ms
	delete One User - success	2 ms
	get a user - bad params	1 ms
	delete One User - fail	1 ms

playlist-api-test.js

```
import { assert } from "chai";
import { playtimeService } from "../playtime-service.js";
import { assertSubset } from "../test-utils.js";
```

```
suite("Playlist API tests", () => {
```

```
  setup(async () => {
```

```
    teardown(async () => {});
```

```
    test("create playlist", async () => {
```

```
      test("delete a playlist", async () => {
```

```
        test("create multiple playlists", async () => {
```

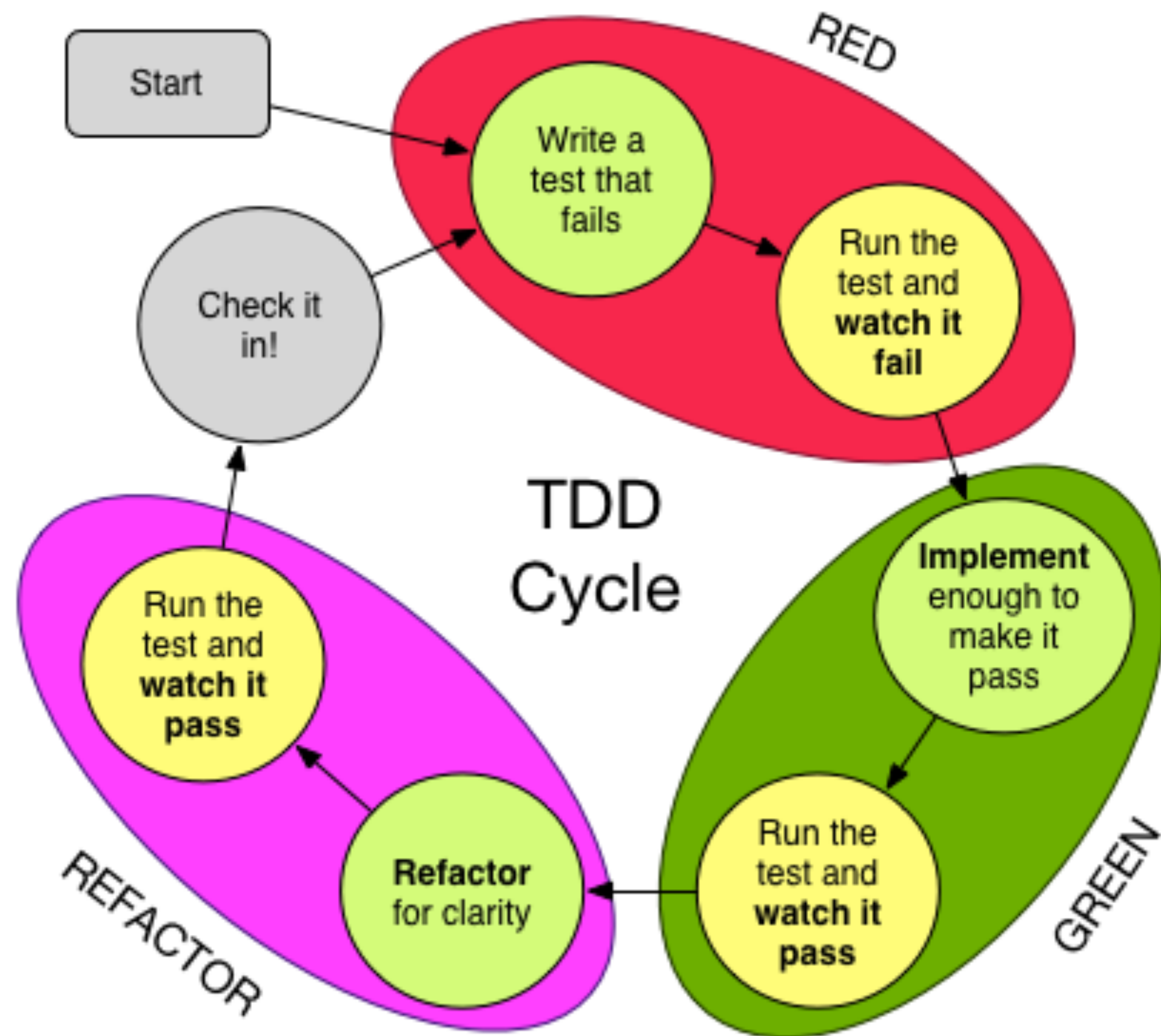
```
          test("remove non-existent playlist", async () => {
```


- Import & initialise fixtures
- Clear data stores
- Create sample user
- Initialise test playlist

playlist-api-test.js

```
...  
import { maggie, mozart, testPlaylists } from "../fixtures.js";  
...
```

```
...  
suite("Playlist API tests", () => {  
  
  let user = null;  
  
  setup(async () => {  
    await playtimeService.deleteAllPlaylists();  
    await playtimeService.deleteAllUsers();  
    user = await playtimeService.createUser(maggie);  
    mozart.userid = user._id;  
  });  
...  
}
```




```
test("create playlist", async () => {
  const returnedPlaylist = await playtimeService.createPlaylist(mozart);
  assert.isNotNull(returnedPlaylist);
  assertSubset(mozart, returnedPlaylist);
});
```

The screenshot shows the VS Code interface with the Run and Test Results window open. The test results show that 1 test failed and 27 tests passed. The failed test is 'create playlist' under 'Playlist API tests'. The error message is 'Error: Request failed with status code 500'.

Test Results	Duration
Test Results	49 ms
Playlist API tests	3 ms
create playlist	2 ms
delete a playlist	0 ms
create multiple playlists	0 ms
remove non-existent playlist	1 ms
User API tests	20 ms
Playlist Model tests	8 ms
Track Model tests	7 ms
User Model tests	11 ms

Error: Request failed with status code 500

```
at createError (node_modules/axios/lib/core/createError.js:16:15)
at settle (node_modules/axios/lib/core/settle.js:17:12)
at IncomingMessage.handleStreamEnd (node_modules/axios/lib/adapters/http.js:312:11)
at IncomingMessage.emit (node:events:402:35)
at endReadableNT (node:internal:streams:245:12)
at processTicksAndRejections (node:internal:process:200:35)
```

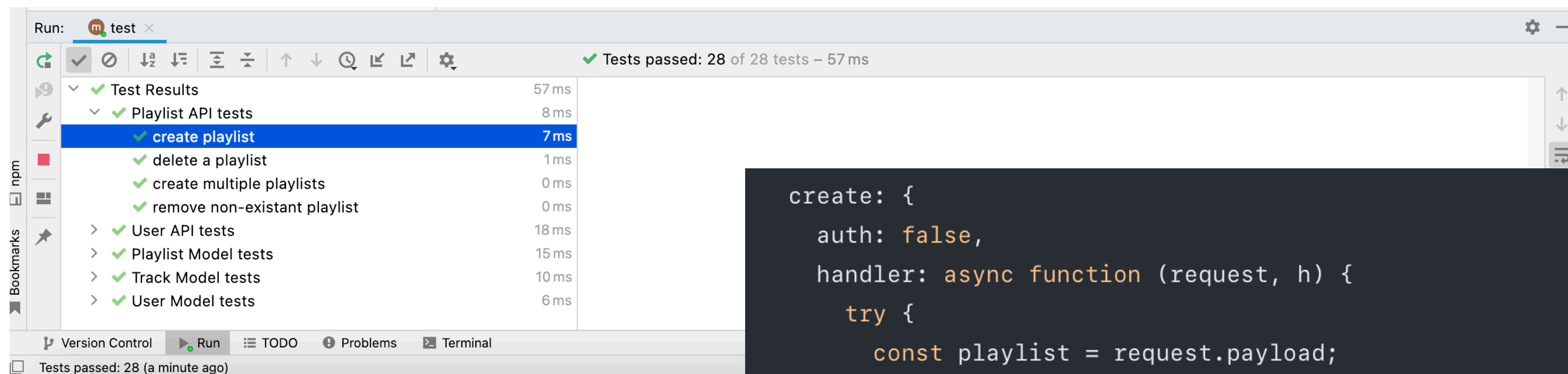
Tests failed: 1, passed: 27 (moments ago)

playlist-api.js

```
import Boom from "@hapi/boom";
import { db } from "../models/db.js";

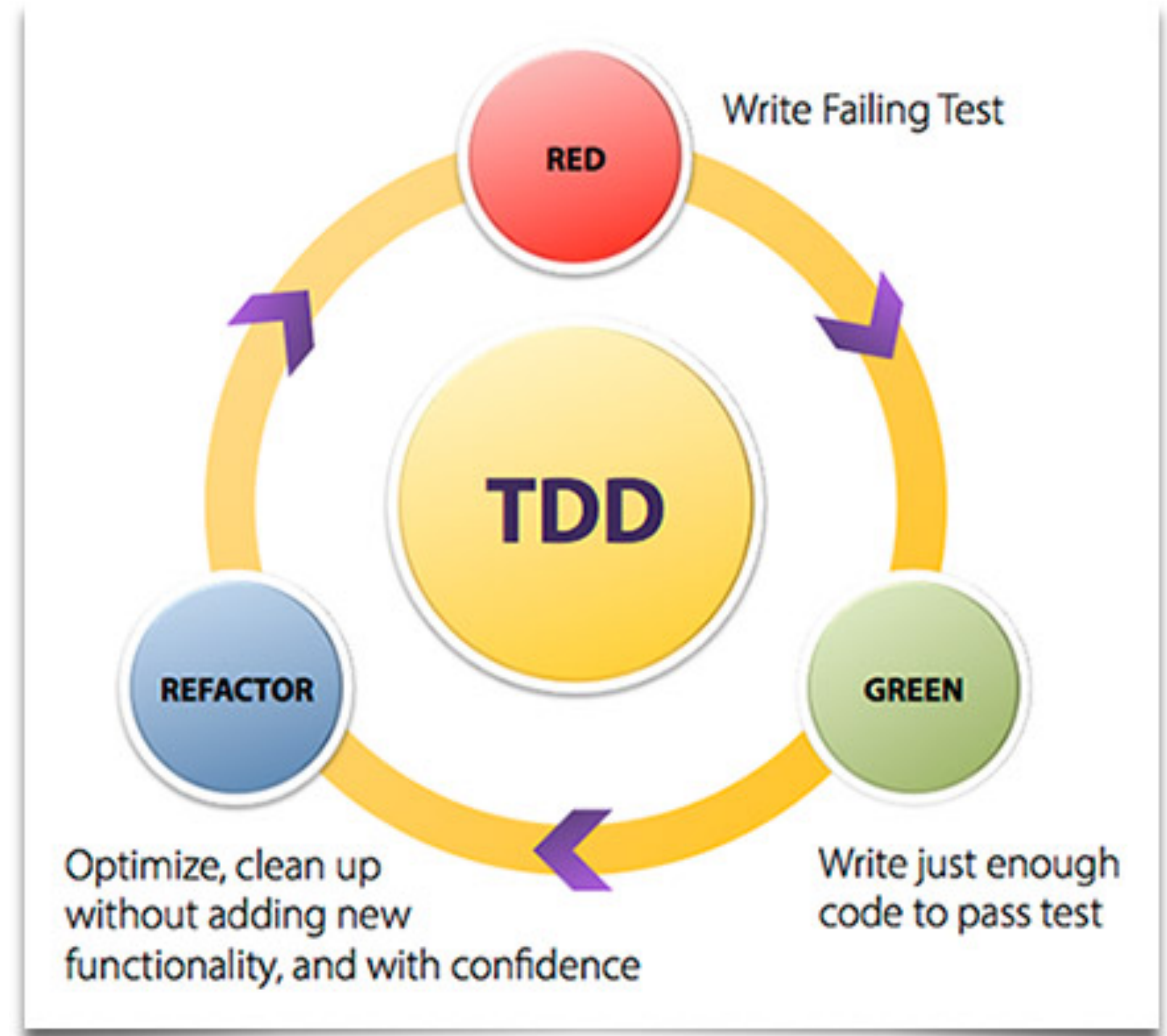
export const playlistApi = {
  create: {
    auth: false,
    handler: async function (request, h) {
      // ...
    },
  },
};
```

```
test("create playlist", async () => {
  const returnedPlaylist = await playtimeService.createPlaylist(mozart);
  assert.isNotNull(returnedPlaylist);
  assertSubset(mozart, returnedPlaylist);
});
```



```
create: {
  auth: false,
  handler: async function (request, h) {
    try {
      const playlist = request.payload;
      const newPlaylist = await db.playlistStore.addPlaylist(playlist);
      if (newPlaylist) {
        return h.response(newPlaylist).code(201);
      }
      return Boom.badImplementation("error creating playlist");
    } catch (err) {
      return Boom.serverUnavailable("Database Error");
    }
  },
},
```

- Write a test
- Test fails
- Do enough to make test pass
- Reflect on the implementation, perhaps simplify/refactor and re run test
- Test passes again



- Exception thrown in server

==> Caught in the test

```
test("delete a playlist", async () => {
  const playlist = await playtimeService.createPlaylist(mozart);
  const response = await playtimeService.deletePlaylist(playlist._id);
  assert.equal(response.status, 204);
  try {
    const returnedPlaylist = await playtimeService.getPlaylist(playlist.id);
    assert.fail("Should not return a response");
  } catch (error) {
    assert(error.response.data.message === "No Playlist with this id", "Incorrect Response Message");
  }
});
```

```
findOne: {
  auth: false,
  async handler(request) {
    try {
      const playlist = await db.playlistStore.getPlaylistById(request.params.id);
      if (!playlist) {
        return Boom.notFound("No Playlist with this id");
      }
      return playlist;
    } catch (err) {
      return Boom.serverUnavailable("No Playlist with this id");
    }
  },
},
```

```
deleteOne: {
  auth: false,
  handler: async function (request, h) {
    try {
      const playlist = await db.playlistStore.getPlaylistById(request.params.id);
      if (!playlist) {
        return Boom.notFound("No Playlist with this id");
      }
      await db.playlistStore.deletePlaylistById(playlist._id);
      return h.response().code(204);
    } catch (err) {
      return Boom.serverUnavailable("No Playlist with this id");
    }
  },
},
```

```

test("create multiple playlists", async () => {
  for (let i = 0; i < testPlaylists.length; i += 1) {
    testPlaylists[i].userid = user._id;
    // eslint-disable-next-line no-await-in-loop
    await playtimeService.createPlaylist(testPlaylists[i]);
  }
  let returnedLists = await playtimeService.getAllPlaylists();
  assert.equal(returnedLists.length, testPlaylists.length);
  await playtimeService.deleteAllPlaylists();
  returnedLists = await playtimeService.getAllPlaylists();
  assert.equal(returnedLists.length, 0);
});

```

```

test("remove non-existent playlist", async () => {
  try {
    const response = await playtimeService.deletePlaylist("not an id");
    assert.fail("Should not return a response");
  } catch (error) {
    assert(error.response.data.message === "No Playlist with this id", "Incorrect Response Message");
  }
});

```

```

find: {
  auth: false,
  handler: async function (request, h) {
    try {
      const playlists = await db.playlistStore.getAllPlaylists();
      return playlists;
    } catch (err) {
      return Boom.serverUnavailable("Database Error");
    }
  },
},

```


api-routes.js

```
{ method: "GET", path: "/api/tracks", config: trackApi.find },
{ method: "GET", path: "/api/tracks/{id}", config: trackApi.findOne },
{ method: "POST", path: "/api/playlists/{id}/tracks", config: trackApi.create },
{ method: "DELETE", path: "/api/tracks", config: trackApi.deleteAll },
{ method: "DELETE", path: "/api/tracks/{id}", config: trackApi.deleteOne },
```

track-api.js

```
import Boom from "@hapi/boom";
import { db } from "../models/db.js";

export const trackApi = {
  find: {
    auth: false,
    handler: async function (request, h) {
    },
  },

  findOne: {
    auth: false,
    async handler(request) {
    },
  },

  create: {
    auth: false,
    handler: async function (request, h) {
    },
  },

  deleteAll: {
    auth: false,
    handler: async function (request, h) {
    },
  },

  deleteOne: {
    auth: false,
    handler: async function (request, h) {
    },
  },
};
```

track-api.test

```
import { assert } from "chai";
import { assertSubset } from "../test-utils.js";
import { playtimeService } from "../playtime-service.js";
import { maggie, mozart, testPlaylists, testTracks, concerto } from "../fixtures.js";
```

```
suite("Track API tests", () => {
  let user = null;
  let beethovenSonatas = null;
```

```
  setup(async () => {
  });
```

```
  teardown(async () => {});
```

```
  test("create track", async () => {
  });
```

```
  test("create Multiple tracks", async () => {
  });
```

```
  test("Delete Track", async () => {
  });
```

```
  test("test denormalised playlist", async () => {
  });
});
```

api-routes.js

```
{ method: "GET", path: "/api/tracks", config: trackApi.find },
{ method: "GET", path: "/api/tracks/{id}", config: trackApi.findOne },
{ method: "POST", path: "/api/playlists/{id}/tracks", config: trackApi.create },
{ method: "DELETE", path: "/api/tracks", config: trackApi.deleteAll },
{ method: "DELETE", path: "/api/tracks/{id}", config: trackApi.deleteOne },
```

Bookmarks	✓ Test Results	80 ms
	> ✓ Playlist API tests	30 ms
	✓ Track API tests	0 ms
	✓ create track	0 ms
	✓ create Multiple tracks	0 ms
	✓ Delete Track	0 ms
	✓ test denormalised playlist	0 ms
	> ✓ User API tests	19 ms
	> ✓ Playlist Model tests	10 ms
	> ✓ Track Model tests	8 ms
	> ✓ User Model tests	13 ms
<div> <div>Git</div> <div>Run</div> <div>TODO</div> <div>Problems</div> <div>Terminal</div> </div> <div>Tests passed: 32 (moments ago)</div>		

track-api.test

```
import { assert } from "chai";
import { assertSubset } from "../test-utils.js";
import { playtimeService } from "../playtime-service.js";
import { maggie, mozart, testPlaylists, testTracks, concerto } from "../fixtures.js";
```

```
suite("Track API tests", () => {
  let user = null;
  let beethovenSonatas = null;

  setup(async () => {
  });

  teardown(async () => {});

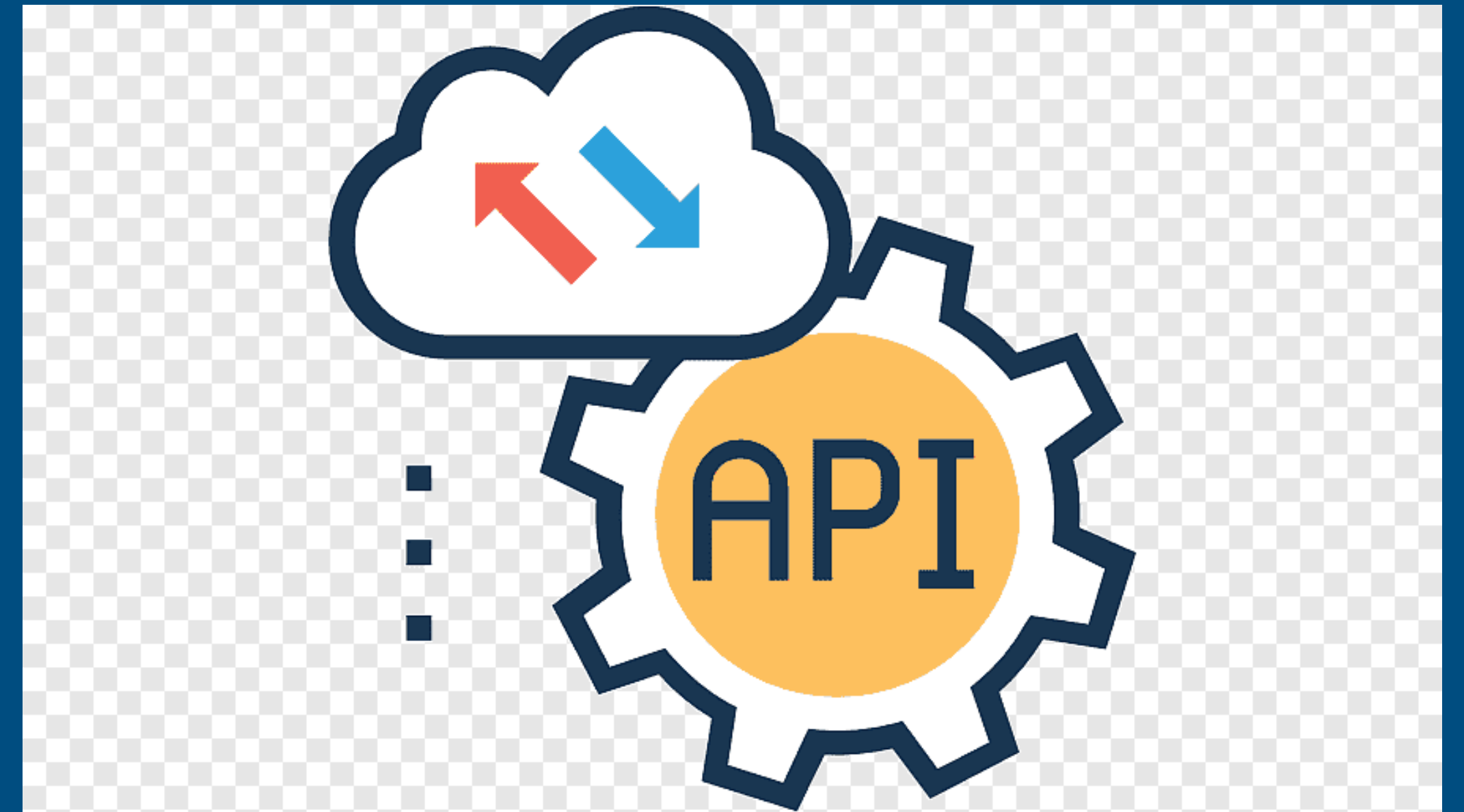
  test("create track", async () => {
  });

  test("create Multiple tracks", async () => {
  });

  test("Delete Track", async () => {
  });

  test("test denormalised playlist", async () => {
  });
});
```

Playtime Rest



Full Stack Web Development