

Project 1: API Back-End Development

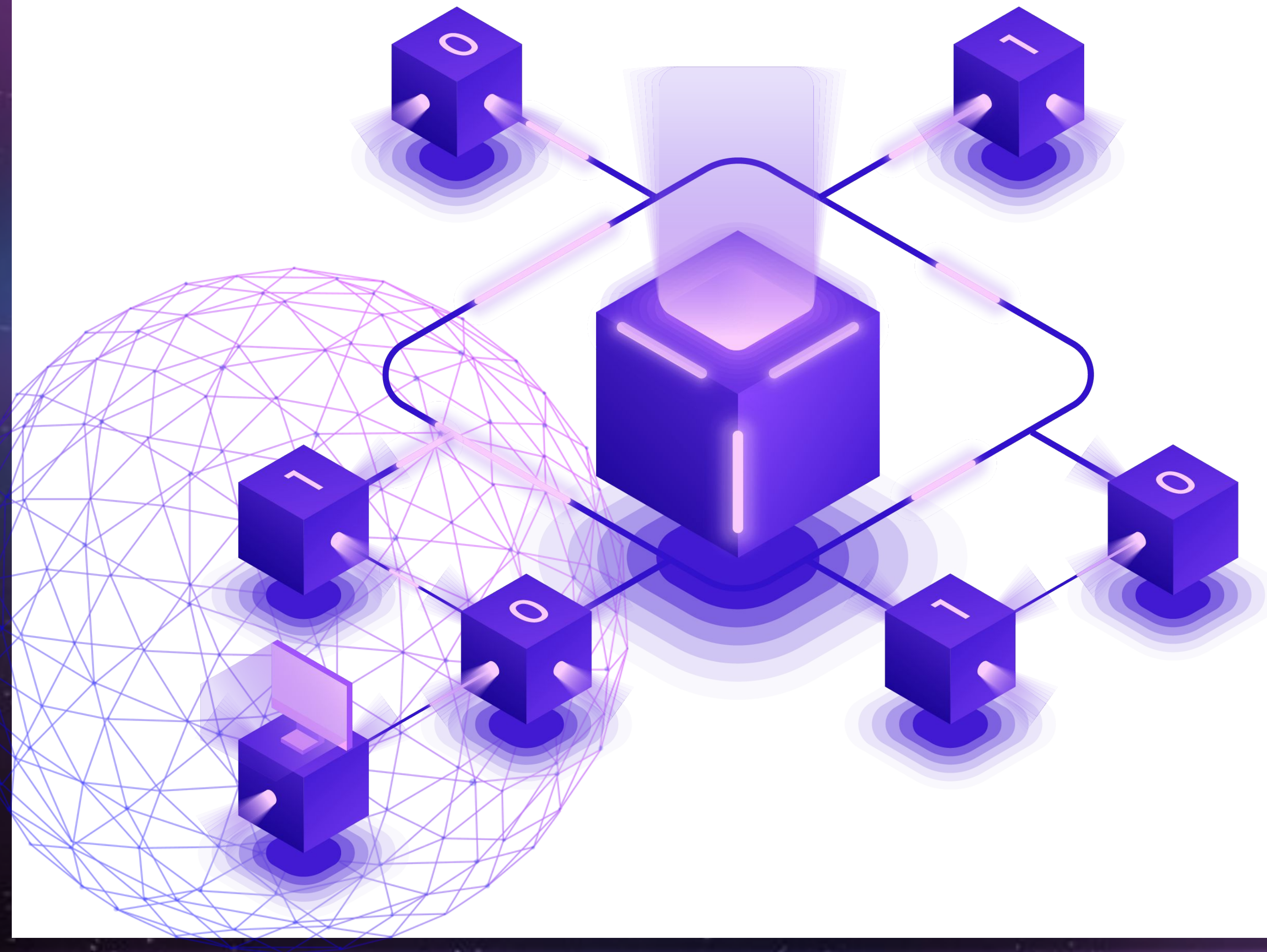
University of British Columbia

Wasi-ul-Hassan Raza
Calvin Qu
Po-Chia Wei
Austin Wong





HelpYa Services Inc.



- Helpya is a trade service acquisition platform
- Clients
 - Harrison Larson
 - Thomas Pattison
- What is the problem?
- Back end API development
- Cross platform (web, ios, android)

Target Audience

- **Businesses**

- Creating advertisement
- Gaining clientele

- **Consumers**

- Seeking services
- Communicating with businesses



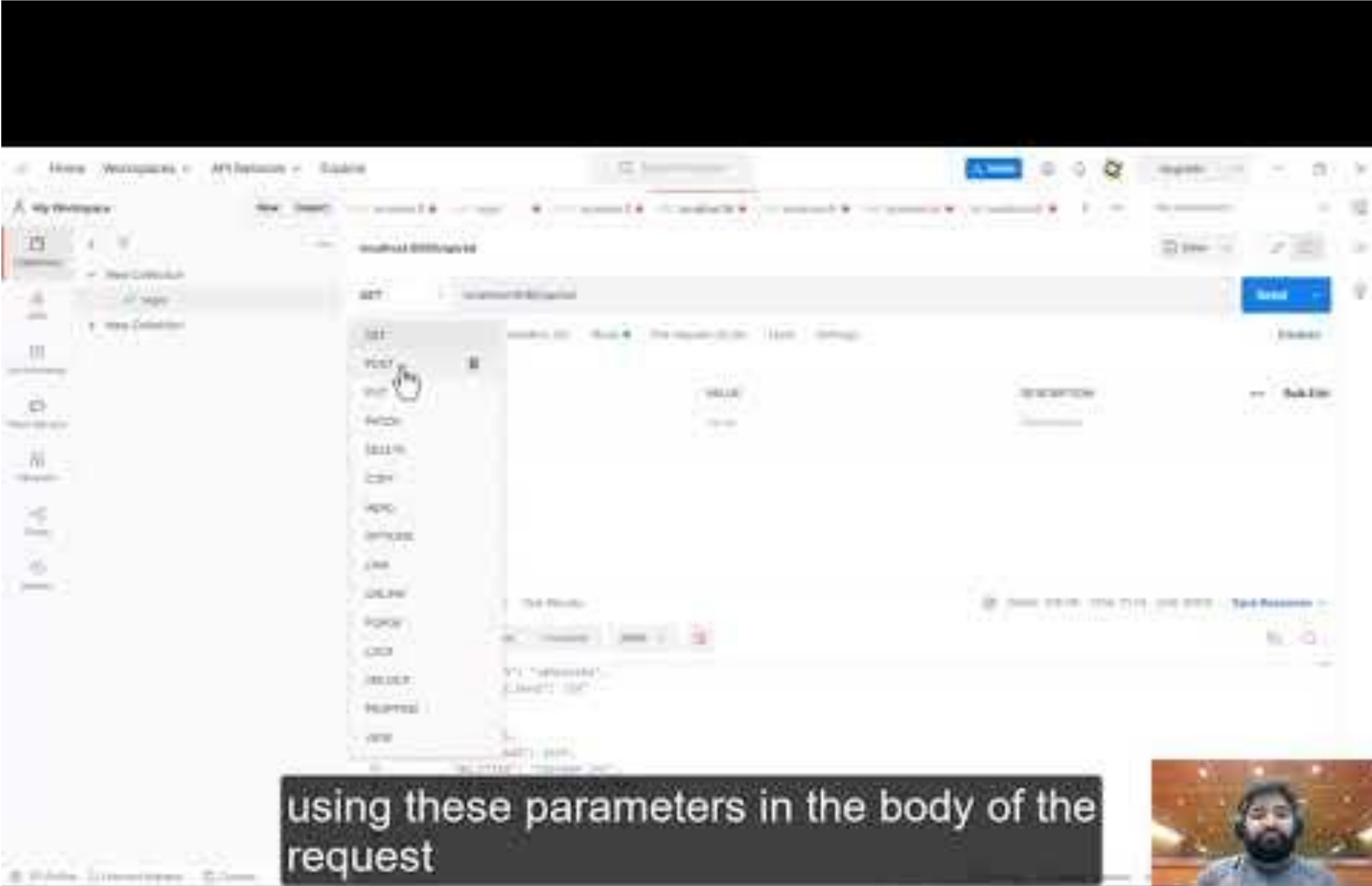
Requirements



The Helpya platform backend API needs to do the following:

- Store and manipulate user information
- Delete user accounts
- Search for a service
- Login
- Transactional endpoints
 - Stripe
- Messaging
- Four types of users

Video



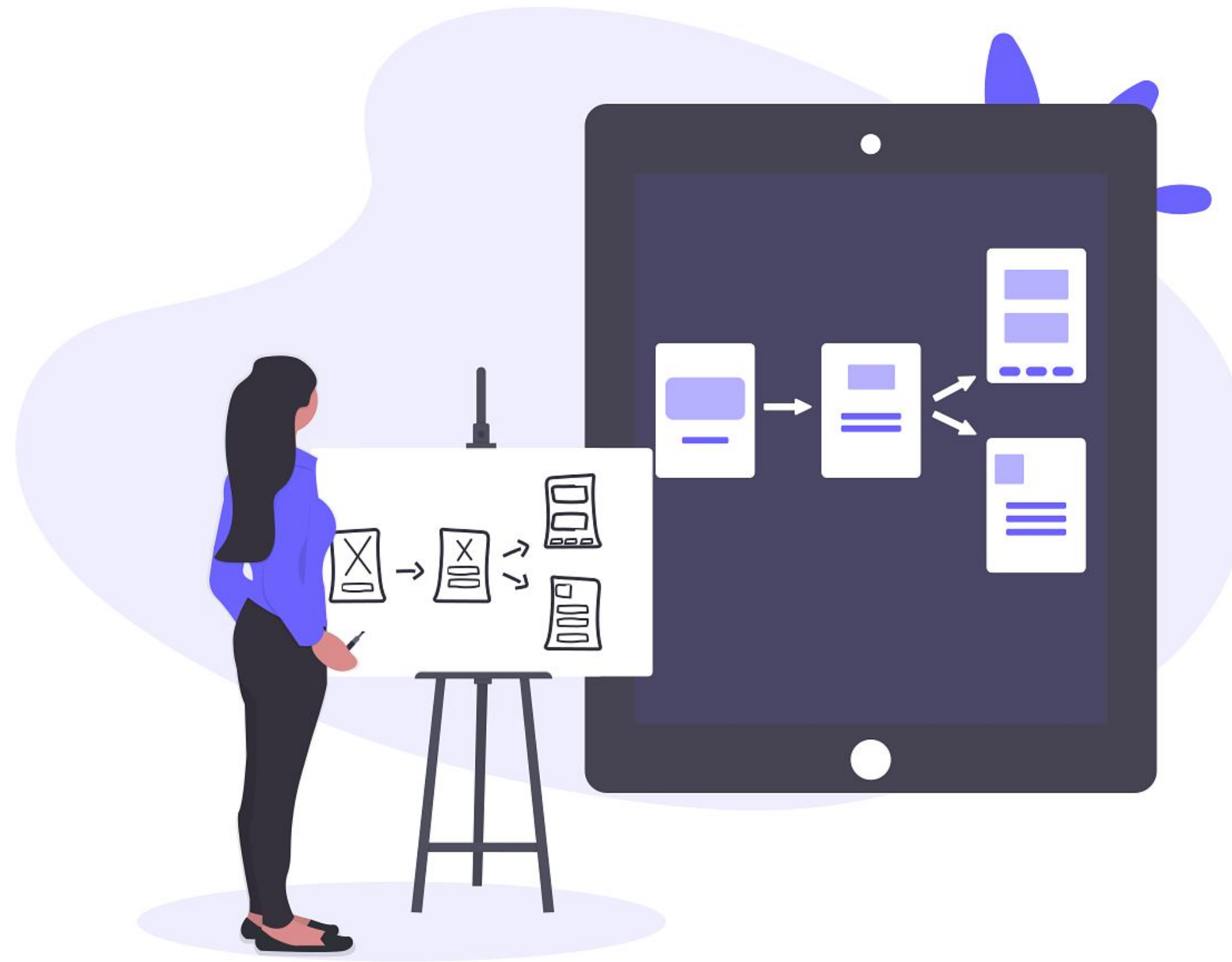
The screenshot shows a video player interface. The main content area displays a REST client application. On the left, there is a sidebar with a 'Parameters' section containing a list of parameters: GET, POST, PUT, PATCH, DELETE, OPTIONS, HEAD, TRACE, and CONNECT. A dropdown menu is open, showing these parameters. The main area shows a 'Send' button and a 'Body' tab. Below the video player, there is a black text box with white text that reads: 'using these parameters in the body of the request'. In the bottom right corner, there is a small video thumbnail of a person with a beard and headphones.

using these parameters in the body of the request

API Development

- **Using Node js**

- Compatibility with other frameworks like react angular vue
- Scalability
- Large community support



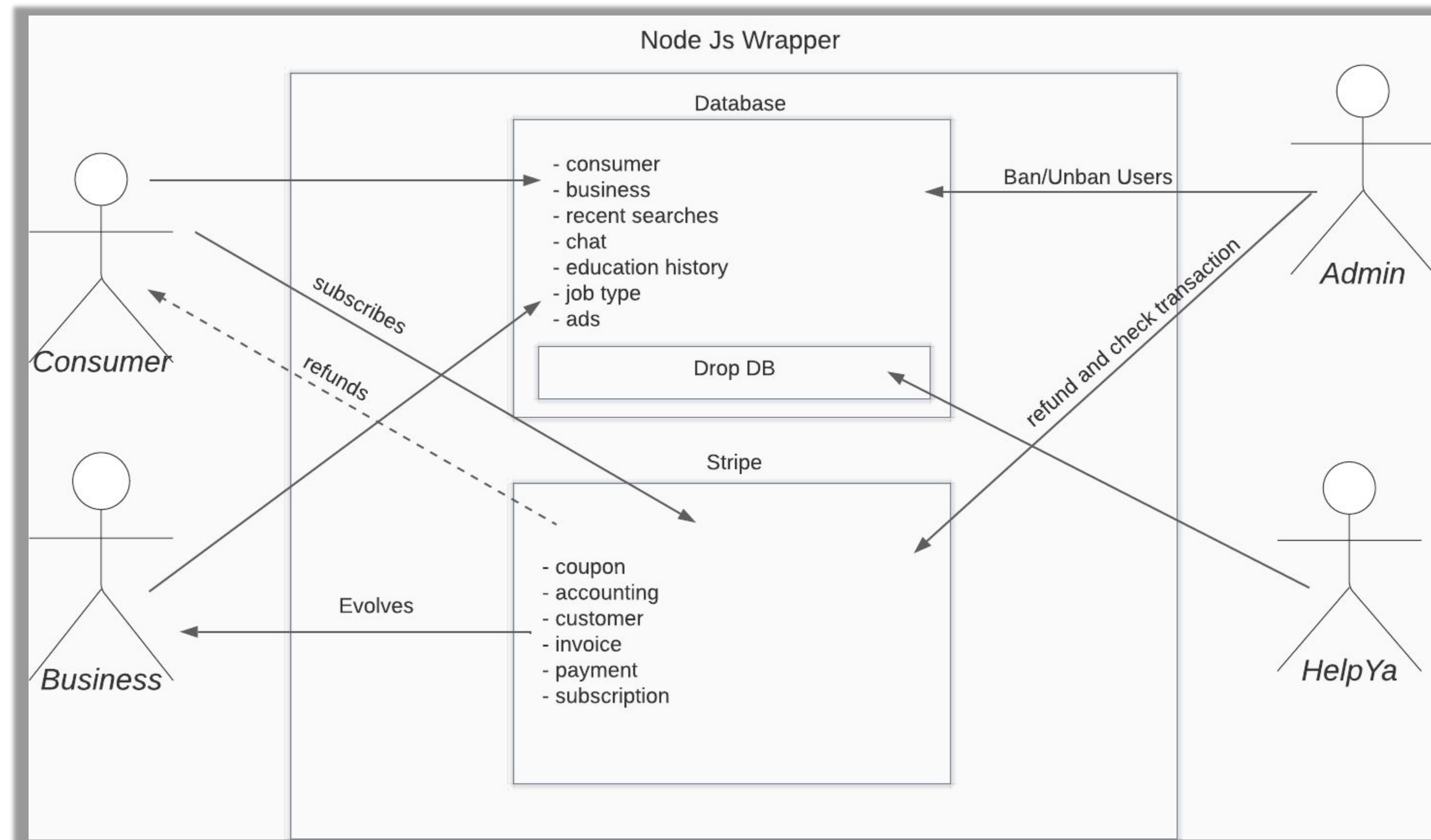
- **Key Components**

- Database (MySQL)
- Tokenization (Express.js)
- Transaction API (Stripe)

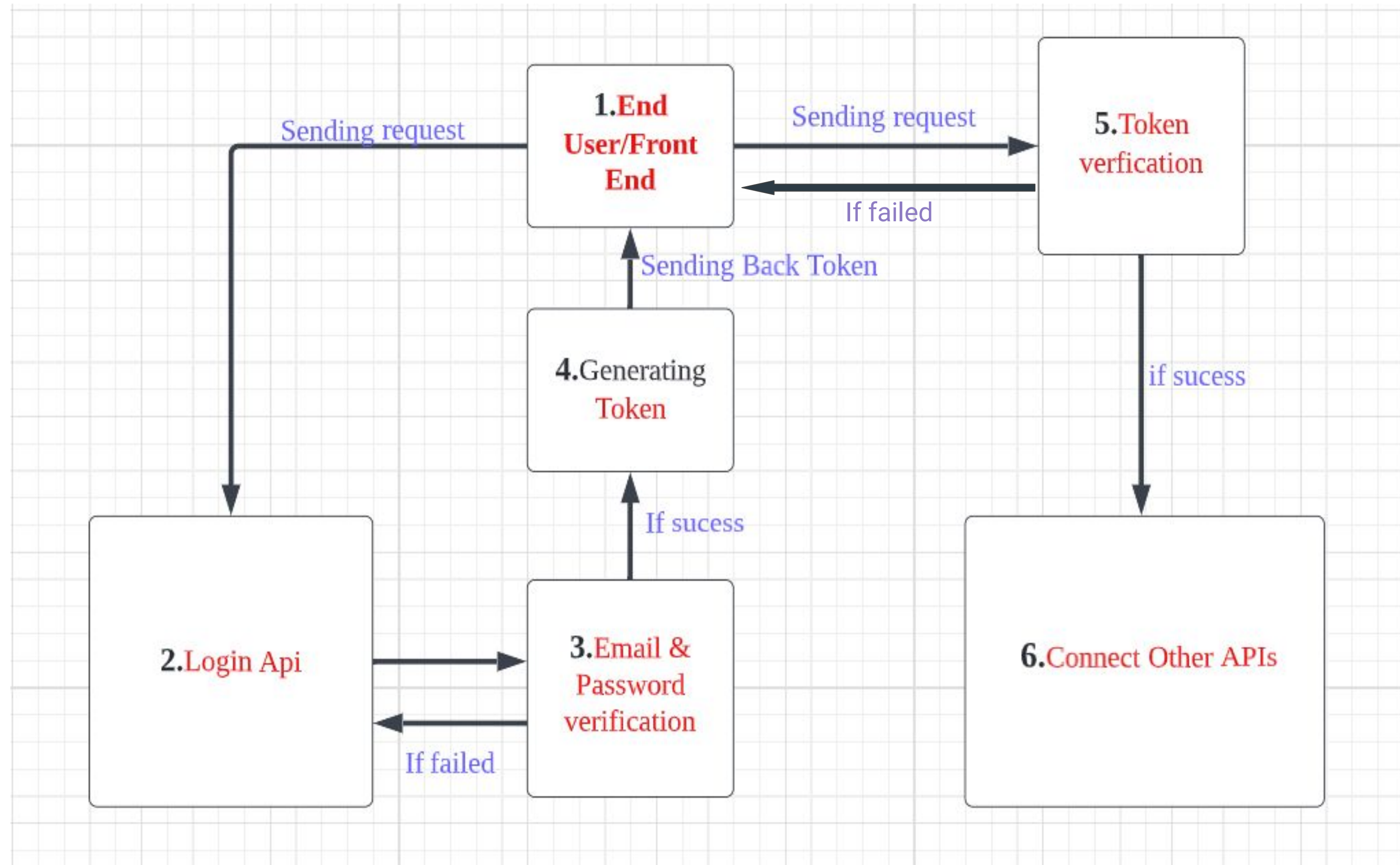
- **CI/CD**

- Heroku
- Drone

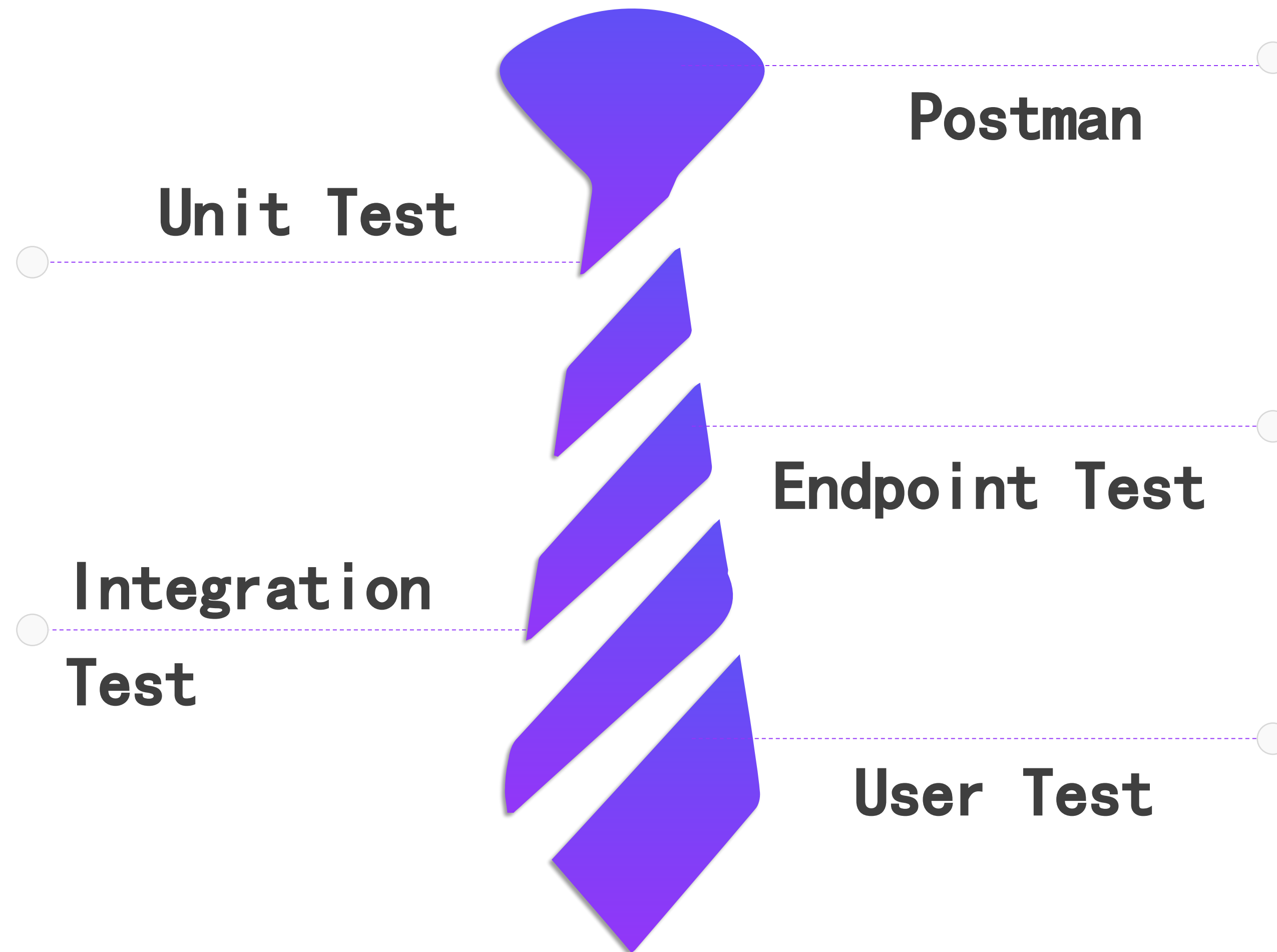
User Case Diagram



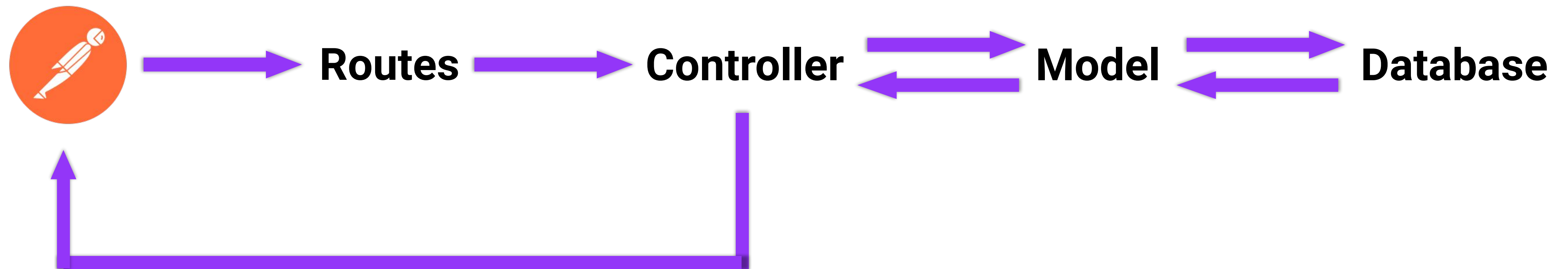
Login



Testing

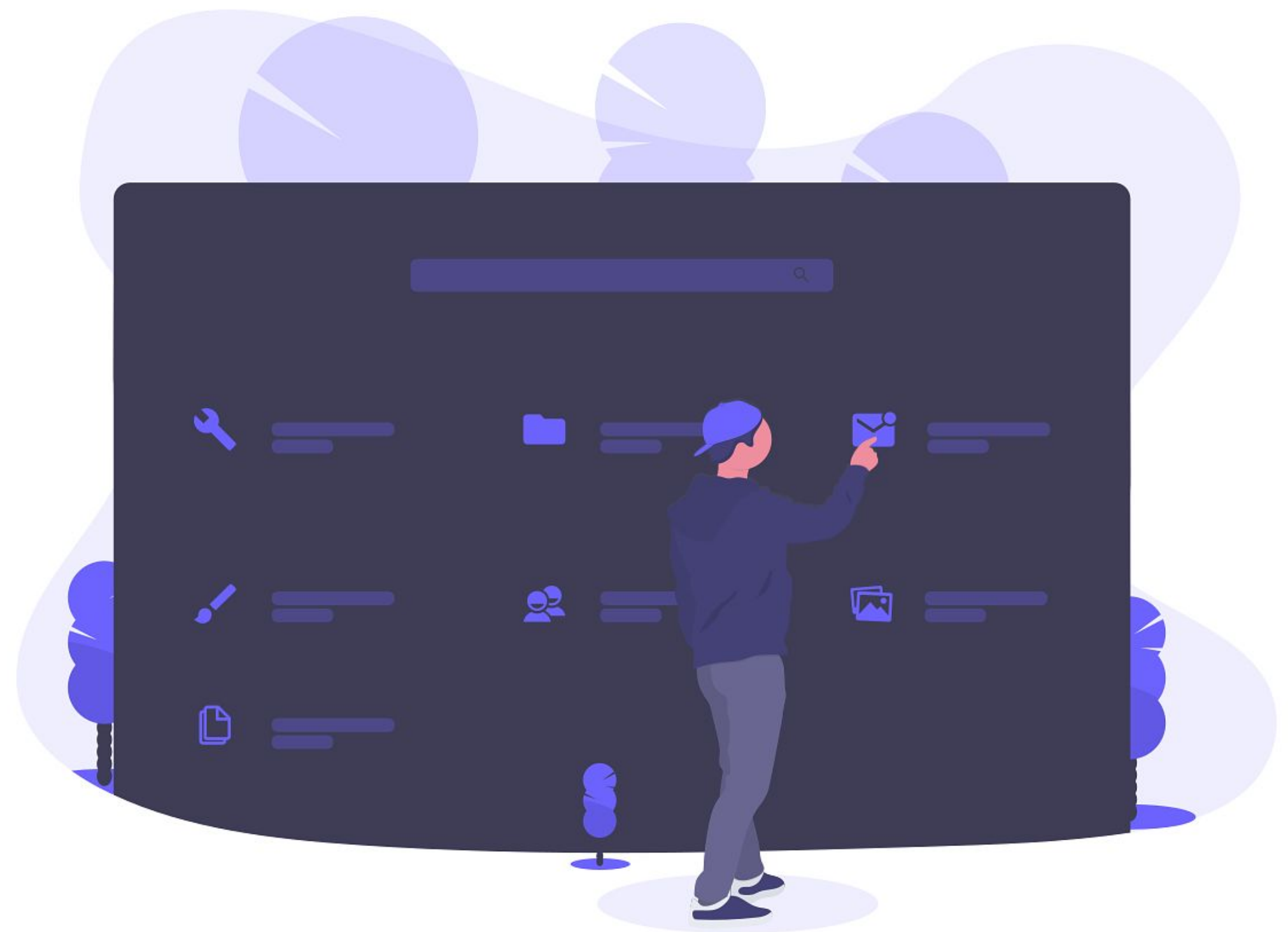


Postman



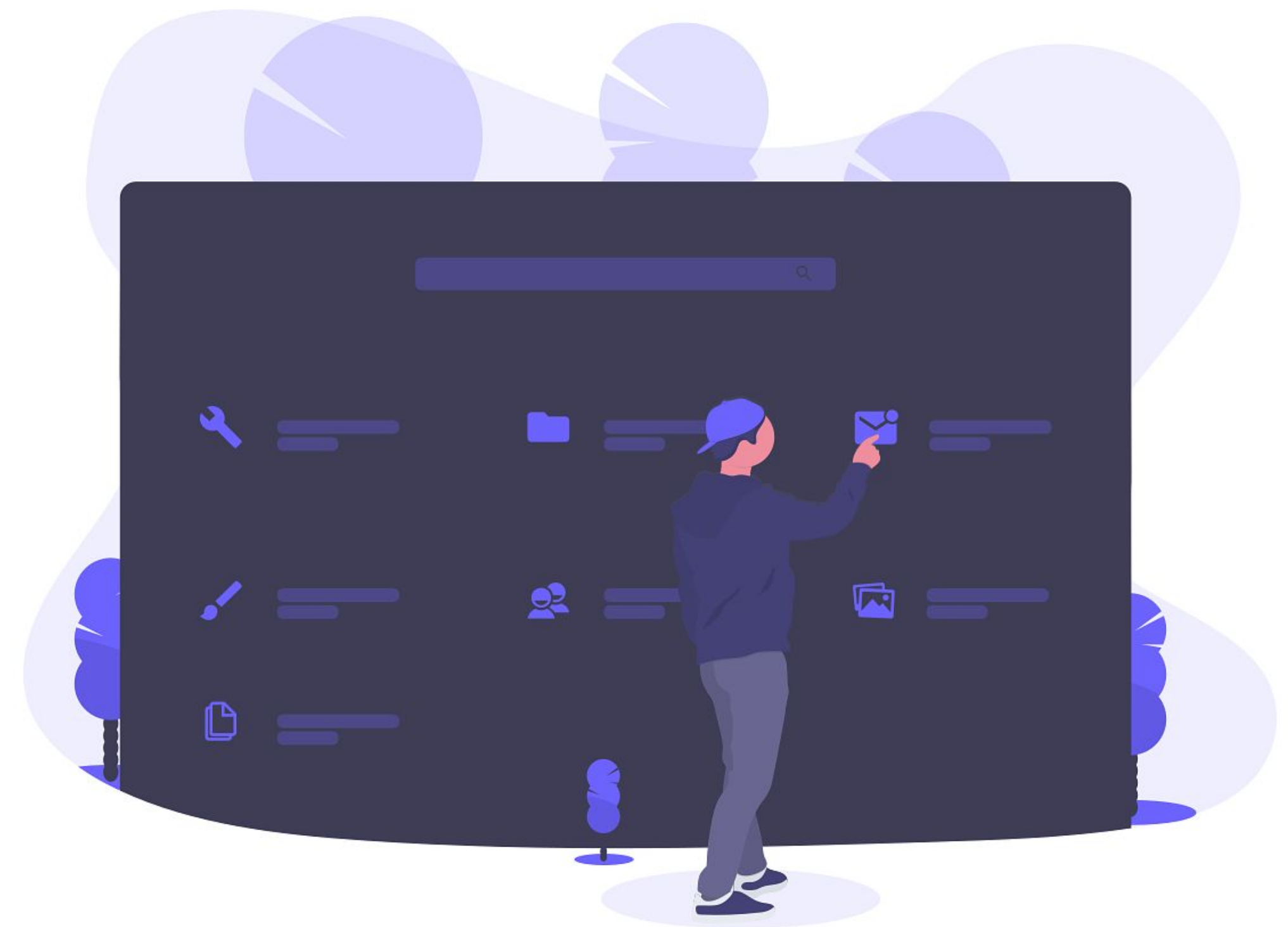
Unit Test

- Mocha Chai method
- Using mock-up Cache Database
- Test every independent module
- Coding Difficulty: Easy



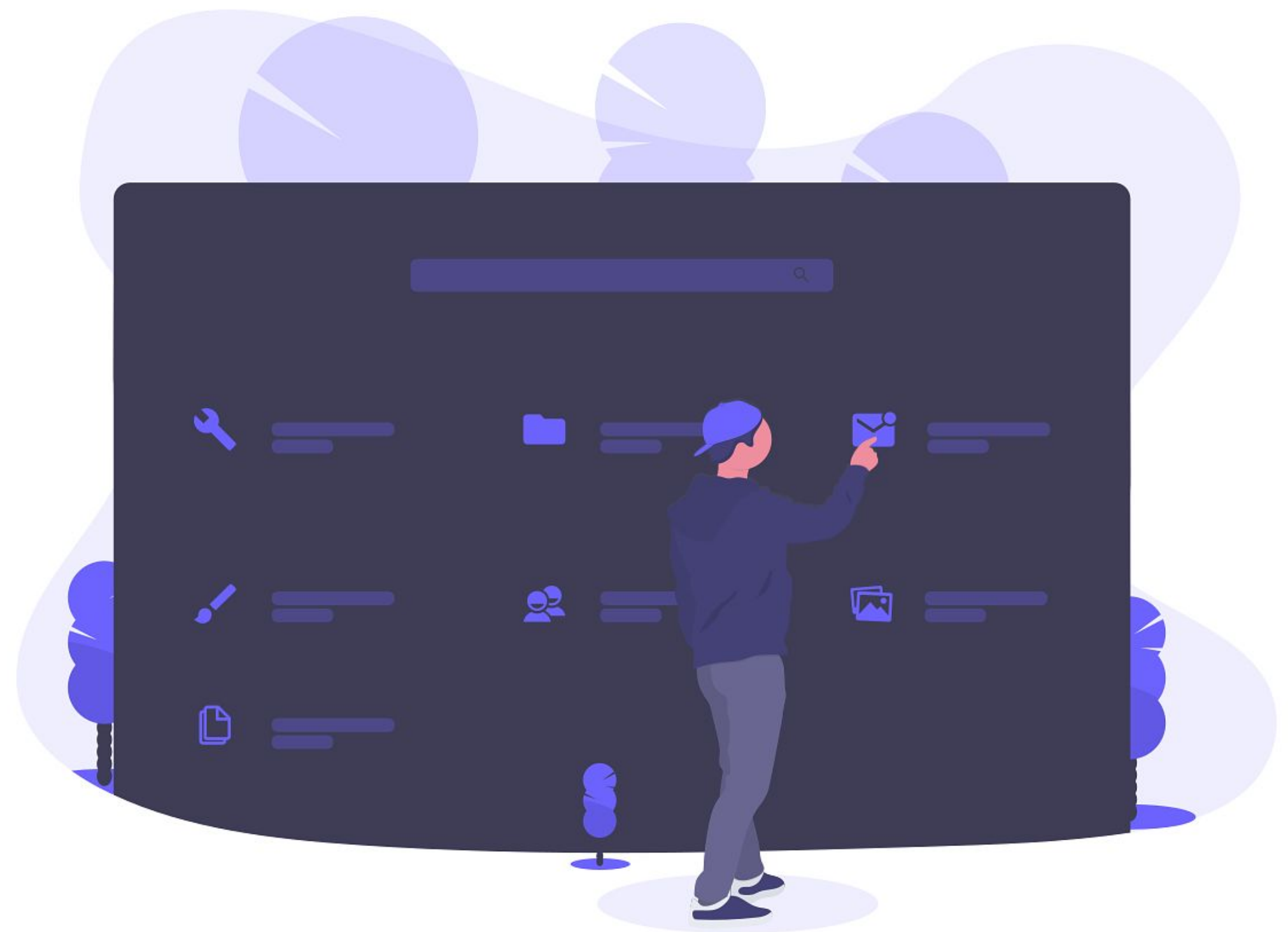
Endpoint Test

- Mocha Chai method
- Test all endpoints
- Able to test without database
- Coding Difficulty: Medium



Integration Test

- Mocha Chai method
- Test interaction between integrated units
- Connected to database
- Process dummy data
- Coding Difficulty: Hard



Testing Example

```
POST /api/consumer
  ✓ consumer create
{ type: 'consumer', id: '1', iat: 1660114694, exp: 1660201094 }

GET /api/consumer
  ✓ get all consumer
{ type: 'consumer', id: '1', iat: 1660114694, exp: 1660201094 }

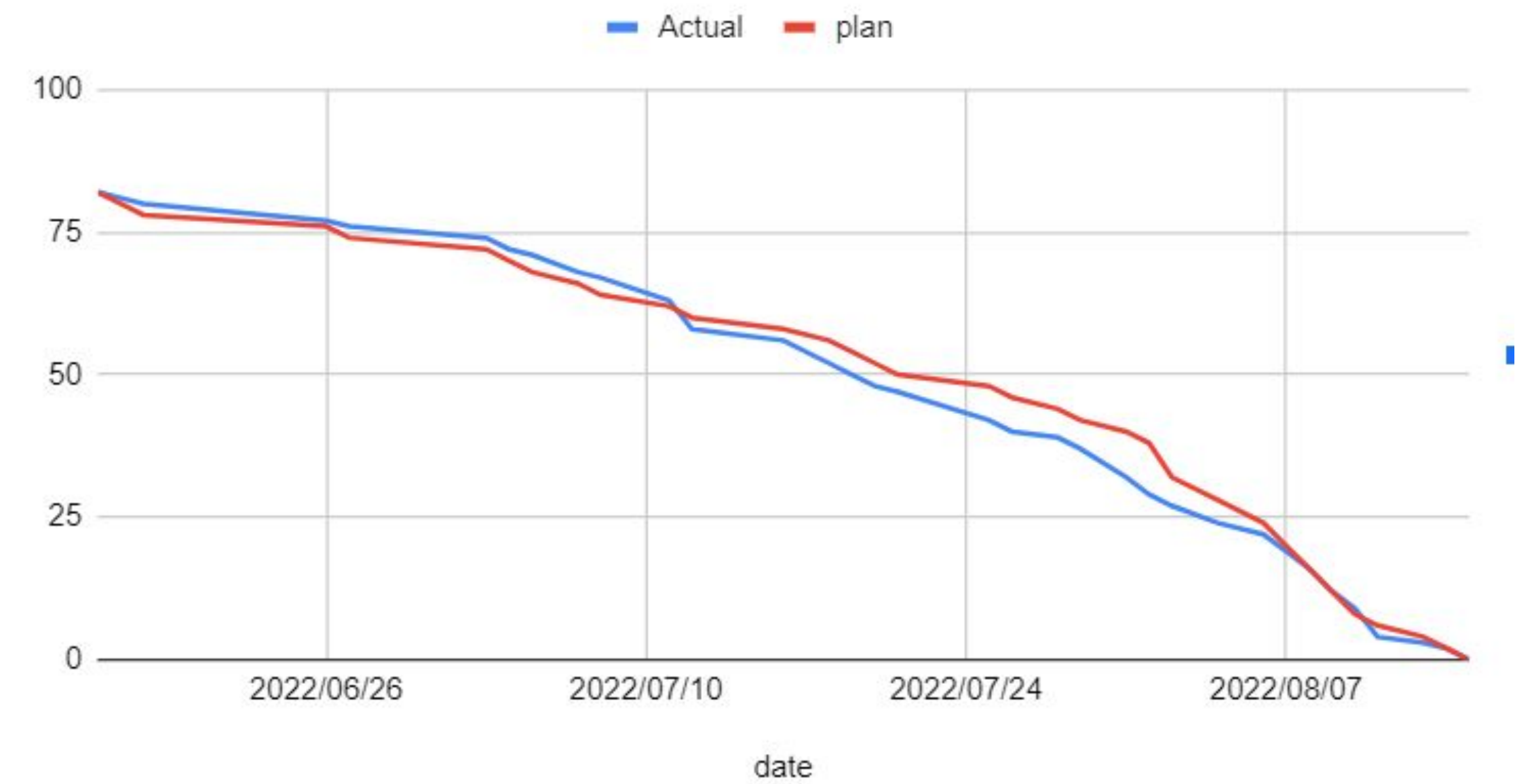
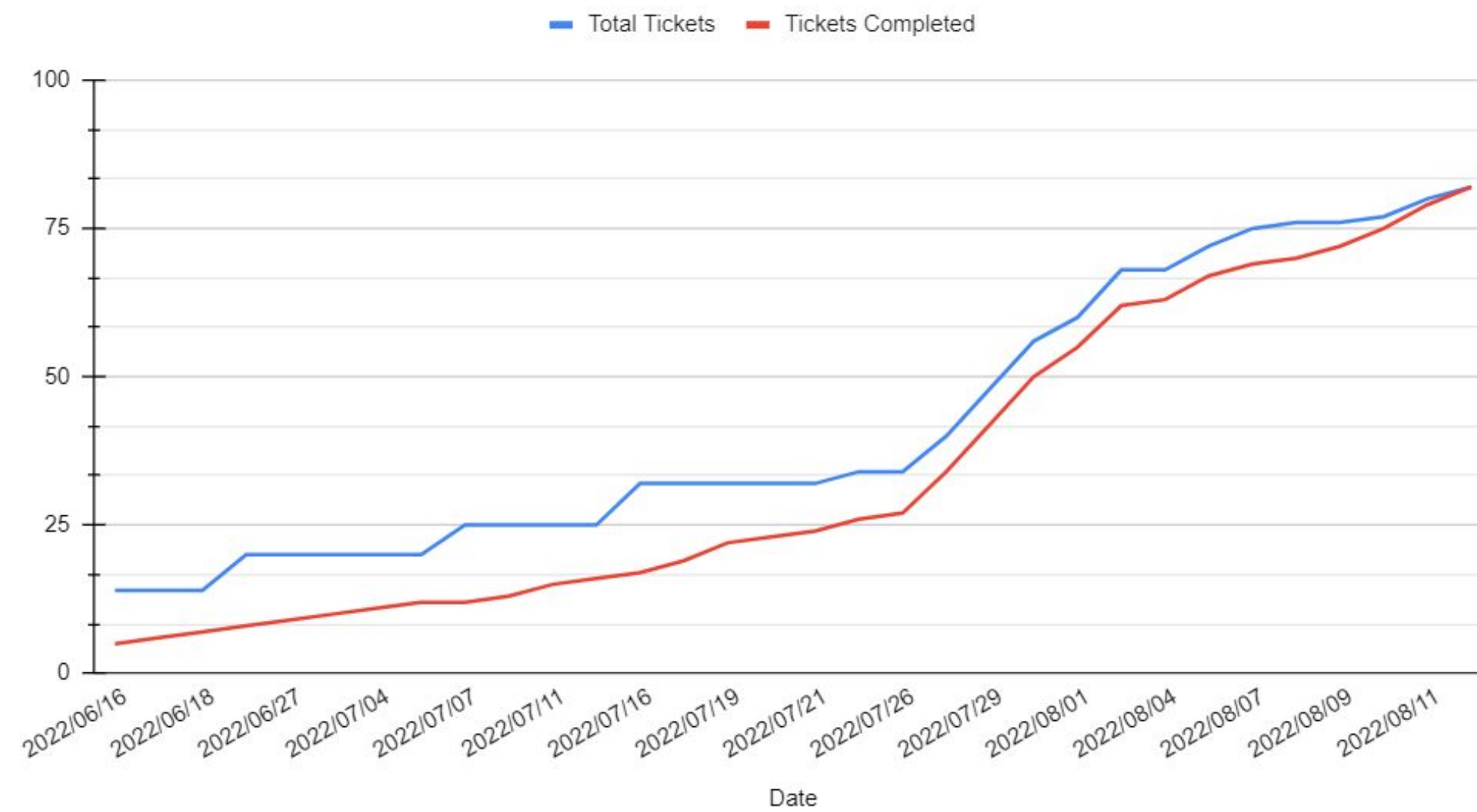
GET /api/consumer
  ✓ consumer findOne
{ type: 'consumer', id: '1', iat: 1660114694, exp: 1660201094 }

PUT /api/consumer
  ✓ consumer update
{ type: 'consumer', id: '1', iat: 1660114694, exp: 1660201094 }

4 passing (51ms)
```


Burn Up/Down Chart

Total Tickets and Tickets Completed



Project Management

- Workflow: Agile



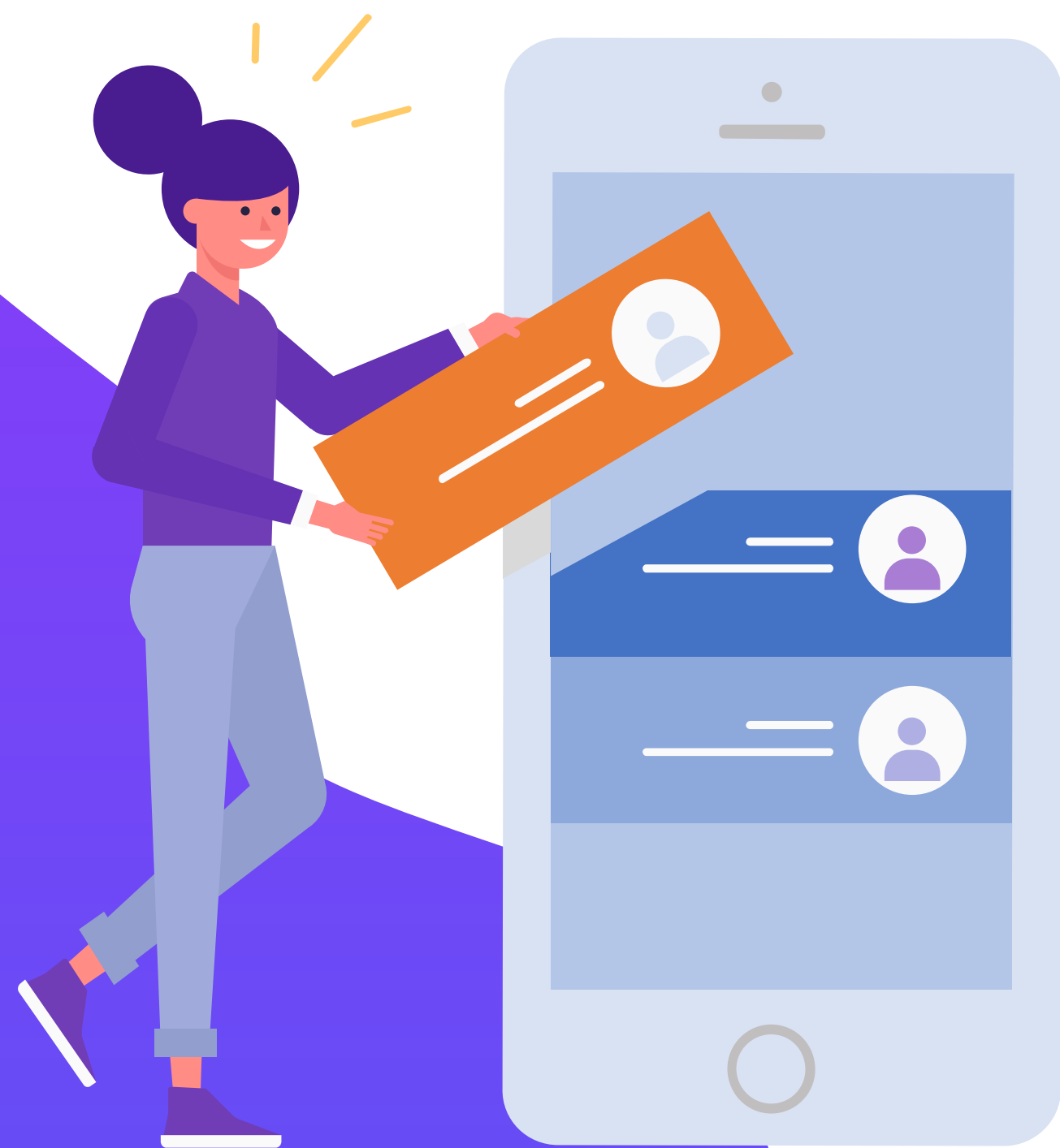
- Scrum meetings
- Tickets to manage tasks



Total Project Time



Challenges Throughout Development



Coding challenges

- Drone

Updating Requirements

- Evolving Requirements

Breaking Down Tasks

- Route Guards
- Pagination
- Insertion Guard

Conclusion



Any Questions?

