ENCOMPASS

PERFECT STRANGERS

EPI-USE LABS

value through innovation

Name	Student Number
JULIANNA VENTER	U20433752
MORGAN BENTLEY	U18103007
KEABETSWE MOTHAPO	U21543462
RONIN BROOKES	U19069686
SAMEET KESHAV	U21479373

Coding Standards

Table of Contents

Introduction

The coding standards outlined in this document provide guidelines and best practices to be followed during the development of Encompass. Adhering to these standards will contribute to a consistent, maintainable, and efficient codebase. The following sub-topics are covered in this document:

1. Header Information in Files:

Header information provides important details about a file's purpose, authorship, and modification history. It is essential for clear documentation and version control. This section will define the required information and formatting for header comments in Encompass files.

2. Header Information for Functions:

Clear and informative function headers aid in understanding the purpose, parameters, and return values of functions. This section will specify the recommended format and content for function headers to ensure code readability and ease of maintenance.

3. Naming Conventions:

Consistent and meaningful naming conventions enhance code comprehension and maintainability. This section will outline the preferred naming conventions for variables, functions, classes, and other elements in the Encompass codebase.

4. API Responses:

Well-structured and consistent API responses improve interoperability and ease of integration with other systems. This section will define the guidelines for formatting API responses, including status codes, error handling, and data structures.

5. File Structure:

An organized file structure simplifies navigation and promotes modular design. This section will provide recommendations for organizing files within Encompass, including the placement of source code files, configuration files, and assets.

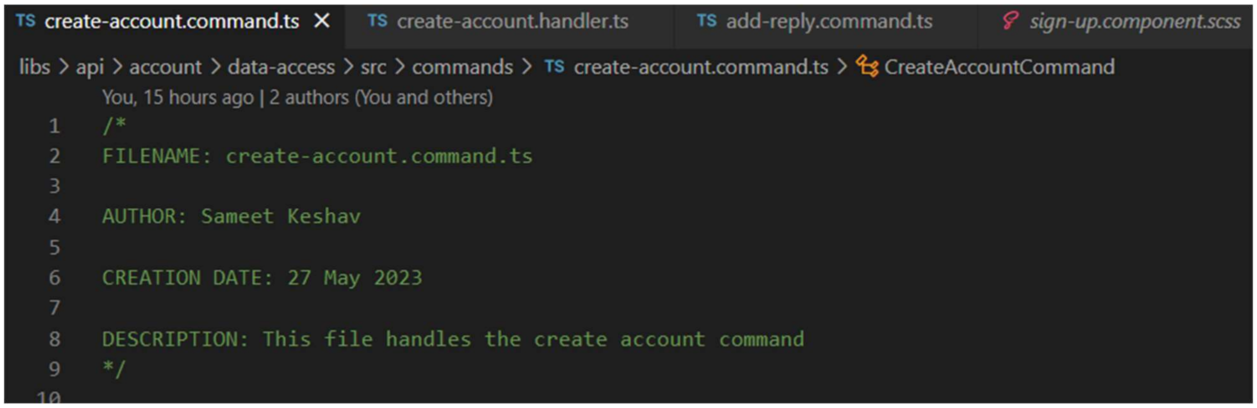
6. Repository Structure:

An efficient repository structure facilitates collaboration, version control, and code sharing. This section will outline the suggested structure for the Encompass repository, including branches, directories, and file organization.

Coding Standards

Header Information in files:

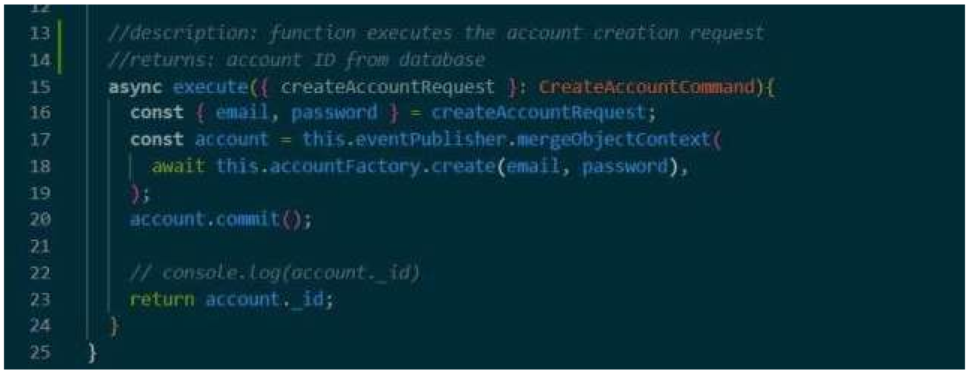
- Filename
- Authour(s)
- Date Created
- Description of File


TS create-account.command.ts ×
TS create-account.handler.ts
TS add-reply.command.ts
sign-up.component.scss
libs > api > account > data-access > src > commands > TS create-account.command.ts > CreateAccountCommand
You, 15 hours ago | 2 authors (You and others)

```
/*
FILENAME: create-account.command.ts

AUTHOR: Sameet Keshav

CREATION DATE: 27 May 2023

DESCRIPTION: This file handles the create account command
*/
```

Header Information for Functions:

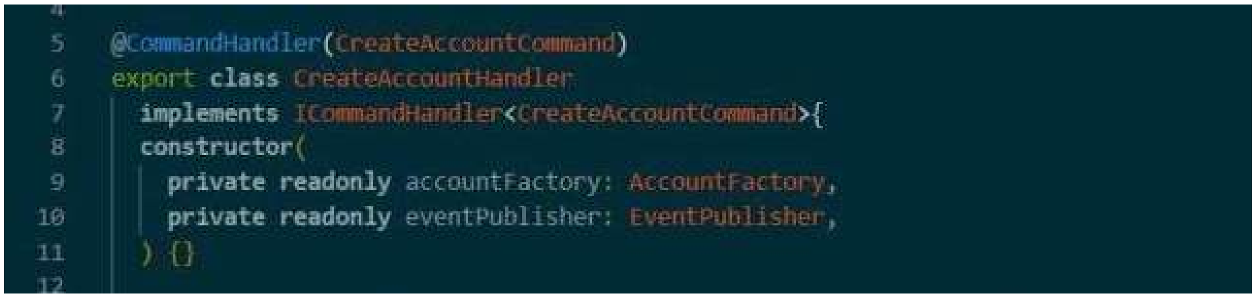
- Description
- Return type & value



```
  //description: function executes the account creation request
  //returns: account ID from database
  async execute({ createAccountRequest }: CreateAccountCommand){
    const { email, password } = createAccountRequest;
    const account = this.eventPublisher.mergeObjectContext(
      await this.accountFactory.create(email, password),
    );
    account.commit();

    // console.log(account._id)
    return account._id;
  }
}
```

Naming conventions:

- All our variables are named with small-letter words, at times it is phrases, merged into one word.
- Object types are named with words, starting with capital letters.



```
@CommandHandler(CreateAccountCommand)
export class CreateAccountHandler
  implements ICommandHandler<CreateAccountCommand>{
  constructor(
    private readonly accountFactory: AccountFactory,
    private readonly eventPublisher: EventPublisher,
  ) {}
```

API Responses:

- 200 – no errors returned
- 400 – incomplete request error message
- 403 – user authentication failed
- 404 – resource not found in database
- 409 – user already exists
- 500 – internal server errors occur

File Structure:

- The working code is in two folders:
  - App
  - Api

- Each feature has a subfolder in the App (front-end code)and Api(back-end) folders.
- In API folder, each feature has the following folders:
  - commands
  - database
  - dto
  - events
  - lib
  - queries
- In APP folder, each feature has the following folders:
  - data-access
  - feature
  - util
- Folders have respective .ts, .scss, .html, .json files.

Repository Structure:

- Dev branch is the main branch.
- Each feature has its own branch, which is merged into the dev branch after development. This is to safe keep code in the dev branch and heighten the coding consistency.

Conclusion

By adhering to the above coding standards, the Encompass development team will create a cohesive and maintainable codebase that supports scalability, extensibility, and ease of collaboration. Consistency in coding practices will streamline code reviews, enhance productivity, and contribute to the long-term success of the Encompass project.

PERFECT STRANGERS