1. Mục đích

Một đồ án tốt nghiệp là phần mềm hỗ trợ phát hiện đạo văn (plagism detech). Phần mềm này tạo ra nhằm giúp cho các bạn có thể kiểm tra và phát hiện mình có đang sao chép tài liệu của người khác hay không.

Người sử dụng sẽ là các giảng viên, các bạn sinh viên, và tất cả mọi người có nhu cầu kiểm tra.

Phần mềm giúp phát hiện và ngăn chặn kịp thời các tài liệu trước khi được phổ biến ra các phương tiện thông tin đại chúng.

1. Phạm vi

Sản phẩm trước mắt sẽ chỉ được giới hạn là một đồ án tốt nghiệp, nên sẽ nói về quy trình thực hiện và mô tả sản phẩm.

Sản phẩm có thể được cải tiến thêm để trở thành một sản phẩm thương mại.

1. Giới hạn

Đồ án thực hiện từ tháng 1. Ý tưởng từ việc kiểm tra đạo văn giúp các bạn sinh viên tránh sao chép các tài liệu gốc.

Dự kiến sẽ kết thúc trong đợt bảo vệ đồ án tốt nghiệp vào tháng 7.

1. Đối tượng

Các sinh viên thực hiện đồ án này sẽ chịu trách nhiệm chính bao gồm: Lê Đức Thuận, Lê Trần Lâm An, Nguyễn Hoàng Tuấn Cường, Diệp Hải Bình, Nguyễn Thanh Bình.

Các giáo viên hướng dẫn sẽ giúp chỉnh sửa sản phẩm theo tiến độ.

1. Kết quả dự kiến

Sản phẩm là một trang web giúp phát hiện đạo văn. Người dùng sẽ được tạo tài khoản và sử dụng.

Kết quả cuối cùng là trang web sẽ hoạt động với độ chính xác tương đối cao.

1. Hướng dẫn cài đặt

Front-end

**CLCDaoVan**

**Getting started**

1. Go to project folder and install dependencies:

npm install

1. Launch development server, and open localhost:4200 in your browser:

npm start

**Project structure**

dist/ web app production build

docs/ project docs and coding guides

e2e/ end-to-end tests

src/ project source code

|- app/ app components

| |- core/ core module (singleton services and single-use components)

| |- shared/ shared module (common components, directives and pipes)

| |- app.component.\* app root component (shell)

| |- app.module.ts app root module definition

| |- app-routing.module.ts app routes

| +- ... additional modules and components

|- assets/ app assets (images, fonts, sounds...)

|- environments/ values for various build environments

|- theme/ app global scss variables and theme

|- translations/ translations files

|- index.html html entry point

|- main.scss global style entry point

|- main.ts app entry point

|- polyfills.ts polyfills needed by Angular

+- test.ts unit tests entry point

reports/ test and coverage reports

proxy.conf.js backend proxy configuration

**Main tasks**

Task automation is based on [NPM scripts](https://docs.npmjs.com/misc/scripts).

| **Task** | **Description** |
| --- | --- |
| npm start | Run development server on http://localhost:4200/ |
| npm run serve:sw | Run test server on http://localhost:4200/ with service worker enabled |
| npm run build [-- --configuration=production] | Lint code and build web app for production (with [AOT](https://angular.io/guide/aot-compiler)) in dist/ folder |
| npm test | Run unit tests via [Karma](https://karma-runner.github.io/) in watch mode |
| npm run test:ci | Lint code and run unit tests once for continuous integration |
| npm run e2e | Run e2e tests using [Protractor](http://www.protractortest.org/) |
| npm run lint | Lint code |
| npm run translations:extract | Extract strings from code and templates to src/app/translations/template.json |
| npm run docs | Display project documentation |
| npm run prettier | Automatically format all .ts, .js & .scss files |

When building the application, you can specify the target configuration using the additional flag --configuration <name> (do not forget to prepend -- to pass arguments to npm scripts).

The default build configuration is prod.

**Development server**

Run npm start for a dev server. Navigate to http://localhost:4200/. The app will automatically reload if you change any of the source files. You should not use ng serve directly, as it does not use the backend proxy configuration by default.

**Code scaffolding**

Run npm run generate -- component <name> to generate a new component. You can also use npm run generate -- directive|pipe|service|class|module.

If you have installed [angular-cli](https://github.com/angular/angular-cli) globally with npm install -g @angular/cli, you can also use the command ng generate directly.

**Additional tools**

Tasks are mostly based on the angular-cli tool. Use ng help to get more help or go check out the [Angular-CLI README](https://github.com/angular/angular-cli).

**Code formatting**

All .ts, .js & .scss files in this project are formatted automatically using [Prettier](https://prettier.io/), and enforced via the test:ci script.

A pre-commit git hook has been configured on this project to automatically format staged files, using (pretty-quick)[<https://github.com/azz/pretty-quick>], so you don't have to care for it.

You can also force code formatting by running the command npm run prettier.

**What's in the box**

The app template is based on [HTML5](http://whatwg.org/html), [TypeScript](http://www.typescriptlang.org/) and [Sass](http://sass-lang.com/). The translation files use the common [JSON](http://www.json.org/) format.

**Tools**

Development, build and quality processes are based on [angular-cli](https://github.com/angular/angular-cli) and [NPM scripts](https://docs.npmjs.com/misc/scripts), which includes:

* Optimized build and bundling process with [Webpack](https://webpack.github.io/)
* [Development server](https://webpack.github.io/docs/webpack-dev-server.html) with backend proxy and live reload
* Cross-browser CSS with [autoprefixer](https://github.com/postcss/autoprefixer) and [browserslist](https://github.com/ai/browserslist)
* Asset revisioning for [better cache management](https://webpack.github.io/docs/long-term-caching.html)
* Unit tests using [Jasmine](http://jasmine.github.io/) and [Karma](https://karma-runner.github.io/)
* End-to-end tests using [Protractor](https://github.com/angular/protractor)
* Static code analysis: [TSLint](https://github.com/palantir/tslint), [Codelyzer](https://github.com/mgechev/codelyzer), [Stylelint](http://stylelint.io/) and [HTMLHint](http://htmlhint.com/)
* Local knowledgebase server using [Hads](https://github.com/sinedied/hads)
* Automatic code formatting with [Prettier](https://prettier.io/)

**Libraries**

* [Angular](https://angular.io/)
* [Bootstrap 4](https://getbootstrap.com/)
* [ng-bootsrap](https://ng-bootstrap.github.io/)
* [Font Awesome](http://fontawesome.io/)
* [RxJS](http://reactivex.io/rxjs)
* [ngx-translate](https://github.com/ngx-translate/core)
* [Lodash](https://lodash.com/)

**Coding guides**

* [Angular](https://github.com/NLPFinalProject/Do_An_Tot_Nghiep_NLP/blob/main/FrontEnd/docs/coding-guides/angular.md)
* [TypeScript](https://github.com/NLPFinalProject/Do_An_Tot_Nghiep_NLP/blob/main/FrontEnd/docs/coding-guides/typescript.md)
* [Sass](https://github.com/NLPFinalProject/Do_An_Tot_Nghiep_NLP/blob/main/FrontEnd/docs/coding-guides/sass.md)
* [HTML](https://github.com/NLPFinalProject/Do_An_Tot_Nghiep_NLP/blob/main/FrontEnd/docs/coding-guides/html.md)
* [Unit tests](https://github.com/NLPFinalProject/Do_An_Tot_Nghiep_NLP/blob/main/FrontEnd/docs/coding-guides/unit-tests.md)
* [End-to-end tests](https://github.com/NLPFinalProject/Do_An_Tot_Nghiep_NLP/blob/main/FrontEnd/docs/coding-guides/e2e-tests.md)

**Other documentation**

* [I18n guide](https://github.com/NLPFinalProject/Do_An_Tot_Nghiep_NLP/blob/main/FrontEnd/docs/i18n.md)
* [Working behind a corporate proxy](https://github.com/NLPFinalProject/Do_An_Tot_Nghiep_NLP/blob/main/FrontEnd/docs/corporate-proxy.md)
* [Updating dependencies and tools](https://github.com/NLPFinalProject/Do_An_Tot_Nghiep_NLP/blob/main/FrontEnd/docs/updating.md)
* [Using a backend proxy for development](https://github.com/NLPFinalProject/Do_An_Tot_Nghiep_NLP/blob/main/FrontEnd/docs/backend-proxy.md)
* [Browser routing](https://github.com/NLPFinalProject/Do_An_Tot_Nghiep_NLP/blob/main/FrontEnd/docs/routing.md)

Back-end

* Preprocessor.py: gồm các hàm …  
  Xử lý các định dạng file khác nhau (pdf, doc, docx, pptx, csv, xlsx), từ văn bản chia ra các câu.  
  Input: file path.  
  Output: Danh sách các câu.
* Database:

Lưu các câu đã xử lý vào database và gán chỉ mục, id cho nó.

Input: Danh sách các câu

Output: Danh sách các bảng chứa các câu đã được đánh chỉ mục, gán id

* TF-IDF.py:

Rút trích từ khoá cho bài văn bản

Input: Văn bản

Output: từ khoá

* Crawl-data.py:

Kéo dữ liệu từ internet thông qua từ khoá

Input: từ khoá

Output: Danh sách văn bản

* Database:

Lưu danh sách câu của văn bản vào database và gán tên cho nó

1. Hướng dẫn sử dụng

gitclone <https://github.com/NLPFinalProject/Do_An_Tot_Nghiep_NLP.git>

Khởi động server bằng

1. Kết quả thực nghiệm