DATSR Developer's Manual

Release 1.1

Dux D-Zine

Oct 30, 2022

CONTENTS:

CHAPTER

ONE

OVERVIEW

1.1 Technologies Used

- **Angular:** Used as a front end framework in building the website.
- **Flask:** Used as a backend framework to connect database to python scripts and python scripts to the user-facing side of the application.
- **MongoDB:** Used as the database for the application, supplied Python libraries to interface with remote storage.
- **Sphinx:** Used in documentation generation (i.e., turning plaintext reStructuredText files into pdf and html outputs).
- **Microsoft Visio:** Used to create diagrams in documentation.
- **Jira:** Used for team management and organization of tasks.

1.2 Programming Languages Used

- **Python:** Used primarily in the backend to handle data in the web app as well as do computations like score calculation and data transformation.
- **Typescript:** Used primarily in the frontend to build the site's structure, generate HTML/CSS/JS from database outputs, and add dynamic features to the site.
- **HTML/CSS/Javascript:** Primarily generated by Angular. Some HTML and CSS files modified afterwards to add Bootstrap styling.
- **reStructuredText:** Used as plaintext markup language for documentation.

CHAPTER

TWO

FRONTEND DEVELOPMENT GUIDE

2.1 Overview

This section describes how a developer would go about contributing to the frontend of DATSR if they wanted to expand on this project.

2.2 Getting Started

2.2.1 Necessary Software

To setup a development environment for the front end developers should have node.js (version 18.0.0), Angular CLI (version 14.2.7), and firebase tools. For more information on these technologies check out the following links: nodejs ; Angular CLI ; firebase tools .

2.2.2 Proficiencies

Developers of this project should expect to need moderate experience in: Angular CLI Development TypeScript, HTML, and CSS. They should have some experience with JSON files and must be familiar with command-line interaction. In order to effectively contribute to the GitHub repo, programmers should also be proficient in version control.

2.3 Possible Areas of Improvement

- Expand Angular testing suite with continuous integration testing
- More refined UX
- Further styling using HTML, CSS, and Bootstrap .

2.4 Best Practices

Maintaining an agreed upon style guide is imperative to a solid Angular web application. Of course, following regular language style guides is a good way to establish cohesive styles across the project. Developers should run, build and test their code often to ensure no bugs are merged into the project.

CHAPTER THREE

BACKEND DEVELOPMENT GUIDE

3.1 Overview

This section is intended to aid developers who would like to contribute to or modify their own version of the DATSR backend functionality. The backend modules can be found in the "backend" folder in the root directory.

3.2 Getting Started

3.2.1 Necessary Software

To run the backend python modules, and be fully capable of modifying any/all components, on your local machine you will need the following python packages installed: pymongo (version 4.2.0), Flask (version 2.2.2), python (version 3.9.0), numpy (version 1.23.1), and bson (version 0.5.10). For more information on downloading Python, you can visit their download page ; you should be able to download most python libaries using pip in the command line, as described in this link .

3.2.2 Proficiencies

To add to, or modify, the backend codebase a developer should be proficient in python programming and also have substantial familiarity with locating (and interpreting) documentation for a given python library. Specifically, the developer would need familiarity with the pymongo documentation and related developer guides.

In order to fully implement the backend/database functionality such that it can be utilized on the UI (the web-hosted app) a developer would also require proficiency in the basics of app routing with Flask.

3.3 Possible Areas of Improvement

- Adding backend compatibility for more file types (e.g., '.json', .'xlx', .'dat', '.txt', etc.)
- More API routed compatibility to facilitate DATSR users filtering through database results
- Backend production of plots (of the time series themselves and also of the predicted/modeled forecasting task compared to the actual sample_test data)

3.4 Best Practices

Developers should adhere to Python style guidelines as well as maintain adequate version control of altered elements to ensure that the codebase remains functional.

CHAPTER

FOUR

DOCUMENTATION DEVELOPMENT GUIDE

4.1 Overview

This section is intended to aid those who wish to contribute to the DATSR project's documentation. The documentation for this project can be found in the "docs" folder in the root directory.

4.2 What You Need to Start

4.2.1 Software Necessary

To generate builds in the documentation folders, you must have Sphinx downloaded and the Read-The-Docs theme if you wish to create html versions. For more information check out these links to documentation for Sphinx and Read The Docs .

4.2.2 Proficiencies

To add to the documentation a developer should be proficient in reStructuredText for modifying content, Sphinx for generating builds, and Python for configuring Sphinx.

4.3 Possible Areas of Improvement

- More robust documentation of source code
  - Backend modules
  - Frontend modules
- Further documentation of frameworks used and how they have been implemented in DATSR

4.4 Best Practices

Developers should follow standards of reStructuredText as a style guide. They should keep section formatting consistent with existing docs. It is also important to test builds with Sphinx often and to not push anything that has Sphinx errors or warnings.