FEC Form 3X Bundle Assessment

Last Updated: Nov 8, 2021

# Introduction

In 2017, the Federal Election Commission (FEC) published a study of its current electronic platform, including a survey of the existing functionality of the FEC’s free filing software and an in-depth investigation of needs expressed by filers. The recommendations of this study served as the foundation for FEC’s new electronic filing platform. In July 2018, FEC awarded a contract to develop a “bundle” of three applications and APIs (fecfile-online, fecfile-ImageGenerator, fecfile-Validate) based on the Form 3X, which was branded as NextGen eFiling Phase 1. Phase 2 will be rebranded as FECFile Online and will continue the work of developing a secure, custom-built, open-source, cloud-based, electronic-filing-data application.

This assessment supports FEC’s ability to proceed into Phase 2 by 1) evaluating the previous development for the Form 3X bundle using a Quality Assurance Surveillance Plan (QASP), covering thirteen deliverable areas supported by multiple evaluation factors; and 2) determining the feasibility of integrating previously developed code into Phase 2 work.

# Overview and Findings Discussion

For the first part of this study, the assessment team reviewed nine of the thirteen deliverable areas, namely:

* Tested code
* Properly styled code
* Accessibility
* Deployment
* Documentation
* Security
* Design and Usability
* Open Source
* Cloud-native application

While each of the deliverable areas had a mixture of successes, marginal areas for remediation, and significant gaps, our assessment ultimately found that the performance standards for all nine areas were each significantly not met.

Before discussing the reusability of the NextGen eFiling Phase 1 applications, it is important to contextualize the performance measures mentioned above. Based on our understanding, these measures are aspirational—they are prospective goals for the next phase of development and the assessment is intended as a gap analysis against that future state. It is clear from the level of nonconformance found that these performance measures were not in place during Phase 1 development. As such, the assessment results should not necessarily be interpreted as the team’s qualitative judgment of “success” or “failure” on the overall Phase 1 project—many successful projects choose to make tradeoffs and compromises that could result in not meeting these performance standards.

The above discussion helps to rationalize the recommendations provided in the second part of this assessment: While major refactoring is needed to bring the application up to the QASP performance standards, there are significant opportunities for reuse. The technology stack (Angular, Django, Python) is workable and consists of production-level, widely-supported, industry-standard frameworks. Refactoring efforts can focus on reducing code duplication through object-oriented programming paradigms and reducing code complexity by leveraging frameworks (e.g. object relationship mapping in Django). By addressing technical deficiencies through refactoring, we can preserve the business logic that has been captured for the existing forms, fields, and processes. Examples and sample areas for refactoring are provided in the full text below.

While further work needs to be done in several areas—more granular specification and prioritization of the refactoring work, further user research and design around user experience, validation of existing features and user stories—overall, we believe that a significant portion of the existing application can be reused and repurposed.

# Assessment, Part 1: Evaluation Against Quality Assurance Surveillance Plan

## Assessment Ratings Key

|  |  |  |
| --- | --- | --- |
| Assessment | Description | Example |
| Pass | Fully passes specified Acceptable Quality Level (AQL) criteria. | *90% code coverage (vs. 85% required for AQL).* |
| Not Met (Marginal) | Does not meet AQL. With relatively minor adjustments to deliverable or criteria, this deliverable could meet the AQL. | *A large number of linting warnings are found, but are either trivial or could be configured in the scanning tool.* |
| Not Met (No Evidence Presented) | Does not meet AQL. Evidence of compliance is not consistently presented. | *No code reviews on merge exist in the public repository, but other evidence is inconclusive.* |
| Not Met (Significant) | Does not meet AQL. Evidence presented shows clear gaps. | *50% code coverage (vs. 85% required for AQL).* |

## Assessment Results: Summary

| Deliverable | Assessment | Acceptable Quality Level (AQL) | AQL Result |
| --- | --- | --- | --- |
| Tested code | Not Met (Significant) | Minimum of 85% test coverage of all code with meaningful tests as determined by the agency. For example, see testing best practices and section on test-driven development (TDD) in FEC efiling study. | < 13% |
| Tests are run as part of the CI/CD pipeline | No evidence found |
| At least one approving review before code changes are merged. | No evidence found |
| Properly styled code | Not Met (Significant) | 0 linting errors and 0 warnings with automated linter tests (such as Flake8 and ESLint) | 53,350 Total; 2,014 Moderate+ |
| Passing tests are required in CI/CD pipeline to deploy code. | Passing tests not enforced |
| Accessibility | Not Met (Significant) | Less than 10 minimal errors and no major errors reported using an automated scanner and manual testing | 203 errors; 296 warnings |
| Easily distinguishable/high contrast colors, type, and interactive elements | 4 manual findings |
| Functional keyboard/tab navigation | 3 manual findings |
| All non-text content has a text alternative that serves the equivalent purpose. | 2 manual findings |
| Clear webpage title that describes their topic and function/purpose | Single title for entire app |
| Deployment | Not Met (Significant) | Successful build with a single command | Pass |
| Deployable to cloud.gov in a single command | Possible with minor fixes |
| CI/CD scripts are public | CI/CD scripts in repo root |
| Branching strategy is followed. | See discussion |
| Documentation | Not Met (Significant) | Major functionality | Not available |
| Dependencies | License missing |
| Inline code documentation | Limited, no automation |
| System diagram | Not available |
| Database | Not available |
| Production-level decisions | Not public |
| Security | Not Met (Significant) | Code submitted must be free of medium and high-level static and dynamic security vulnerabilities. | Significant work required to meet quality level |
| FEC data security policies should be followed. | N/A |
| Clean tests when dependencies are scanned with an automated tool such as Snyk and npm audit for vulnerabilities | Issues found with 13 dependencies |
| Clean tests when static code analysis using Bandit and passive scanning with a scanner such as OWASP ZAP along with documentation explaining any false positives | Significant work required to meet quality level |
| Not more than one major version behind on essential underlying technology | Typescript and Angular outdated |
| User research | N/A | N/A | N/A |
| Open source | Not Met (Significant) | Public repository is the working code and updated every workday. | Code managed in private repository |
| Issues/tickets are in a public repository and follow issue guidance. | Code managed in private repository |
| Pull request documentation is followed. | None available |
| Design and usability | Not Met (Significant) | Industry best practices for UI design for modern websites. UI considerations include: proper alignment of elements, proper spacing, consistency in sizing, typography, color(s) throughout the product. | FEC Pattern Library not Followed |
| Clear user flow and information architecture | Mixed results for UX heuristics |
| Clear integration of user research and usability best practices | N/A |
| Friendly, easily understood and actionable content | Mixed results for UX heuristics |
| Cloud-native application | Not Met (Significant) | 12-factor application framework is followed. | Majority compliant |
| Applications each have their own codebase. | Frontend and backend need separation |
| Config is stored in the environment. | Env Config Refs in App |
| Database design/structure best practices are documented and followed. | No evidence |
| Model-view-controller (MVC) architectural model and application ORM are used to interact with database. | MVC pass; ORM not met |
| Software is built using object-oriented programming. | OOP not well utilized |
| Performance | N/A | N/A | N/A |
| Mobile accessible | N/A | N/A | N/A |
| Accepted features | N/A | N/A | N/A |

# Deliverable: Tested Code

**Overall Performance Standard:**

* *Code delivered must have substantial test code coverage and a clean code base.*
* *Code goes through peer review before merge.*

**Overall Assessment:** Not Met (Significant)

## AQL: Test Coverage

**AQL**: *Minimum of 85% test coverage of all code with meaningful tests as determined by the agency. For example, see testing best practices and section on test-driven development (TDD) in FEC efiling study.*

**Assessment:** Not Met (Significant)

**Assessment Method:** Manual run of existing unit tests.

**Discussion:** Some minor modifications were required to get the application and unit tests running. While modifications were not significant, this could potentially indicate that unit tests were not maintained throughout the project.

#### Unit Test Coverage by Repository

|  |  |  |
| --- | --- | --- |
| Repository | Language | Coverage |
| fecfile-online Django | Python | 13% |
| fecfile-online Angular | Typescript | Test not running successfully. |
| fecfile-ImageGenerator | Python | 0% |
| fecfile-Validate | Python | 0% |

## AQL: CI/CD Pipeline

**AQL:** *Tests are run as part of the CI/CD pipeline.*

**Assessment:** Not Met (No Evidence Found)

**Assessment Method:** Manual review of Jenkins scripts.

**Discussion:** We were unable to find specific evidence that automated tests were run in a CI/CD pipeline. In the Jenkins CI/CD scripts in the repository, we identified where static code analysis is being run, but automated unit tests were not called here, which would have been a logical place to do so. Given the lack of evidence of unit testing at this step (and anywhere else), it appears that unit testing is not being executed as part of the CI/CD pipeline.

However, it is possible that automated unit testing was configured elsewhere in the Jenkins pipeline. For instance, Jenkins does allow for tests and plug-ins to be configured solely in the UI, i.e. not through Jenkins files or other artifacts that would leave a footprint in the repository. This is a common practice. As such, we are assessing this factor as not met due to lack of positive evidence otherwise.

## AQL: Code Review

***AQL****: At least one approving review before code changes are merged.*

**Assessment:** Not Met (No Evidence Found)

**Assessment Method:** Manual review of repository, comments, and pull requests.

**Discussion:** Examination of pull requests and commit history in the GitHub repository did not identify consistent code review processes or comments reflecting code review (e.g. comments on merge). There were only 2 pull requests on fecfile-online, 2 on fecfile-validate, and 15 on fecfile-image generator. The majority of these pull requests correspond to sprint tagging activities as opposed to individual code review activities.

However, looking at the history stored in GitHub, we can see that commit messages do reference pull requests. This evidence is consistent with a scenario in which the development team used a git repository hosted somewhere other than GitHub, and then imported that repository into GitHub. Given that we do not have access to the pull request notes or comments from the original repository, it is impossible to say if this AQL was met or not.

Documentation of peer review on pull requests and comments on merge in a git repository would be the most visible evidence of required code reviews. However, it seems possible that code review occurred and was documented elsewhere. For example, looking at the history stored in GitHub, we can see that commit messages reference pull requests and ticket numbers, but those requests are not in GitHub. This evidence is consistent with a scenario in which the development team used a git repository hosted somewhere other than GitHub, and then imported that repository into GitHub.

In addition to a separate, non-GitHub repository, there was a separate JIRA instance maintained by the prior vendor (e.g. <https://salientcrgt-fecefiling.atlassian.net/browse/FNE-2181>). That JIRA instance, or another plugin therein, may reflect code review processes not visible here. In another example, there are 30 tickets (out of 1000+) in the FEC New eFiling JIRA project that utilize the “In Code Review” status. All of these may indicate that a code review procedure was utilized but was not well documented or visible to FEC or the public. As such, we are assessing this factor as not met due to lack of positive evidence otherwise.

# Deliverable: Properly Styled Code

**Overall Performance Standard**:

* *Code is automatically checked for linting errors.*

**Overall Assessment:** Not Met (Significant)

### AQL: Linting

**AQL*:*** *0 linting errors and 0 warnings with automated linter tests (such as Flake8 and ESLint)*

**Assessment:** Not Met (Significant)

**Assessment Method:** Flake8 and ESLint automated tests with manual adjudication of scan results.

**Discussion:** Overall, the code base does not meet the AQL for linting. Currently, linting scans result in 53,350 total errors and warnings. However, upon manual review and adjudication of these findings, we would recommend only 2,014 as either Moderate or High priority—these require actual developer review and remediation. The vast majority of linting errors and warnings are Low or Trivial priority, with the majority of those errors being addressable through automated script fixes (e.g. spacing and other non-functional style violations that can be fixed by linting tools or IDEs). The full list of linting errors for each repository and tool are listed in the Appendix.

## AQL: Passing Code Style Required for Deployment

**AQL**: *Passing tests are required in CI/CD pipeline to deploy code.*

**Assessment:** Not Met (Significant)

**Assessment Method:** Manual review of deployment scripts.

**Discussion:** There is no evidence that linting tools are run during CI/CD deployment, nor are passing tests required for a deployment as part of the Jenkins scripts provided.

# Deliverable: Accessibility

**Overall Performance Standard:**

* *Web Content Accessibility Guidelines 2.1 AA and section 508 standards.*

**Overall Assessment:** Not Met (Significant)

## AQL: Automated Scanning

**AQL***: Less than 10 minimal errors and no major errors reported using an automated scanner and manual testing.*

**Assessment:** Not Met (Significant)

**Assessment Method:** HTML Code Sniffer execution of both Section 508 and WCAG AA criteria per page, manual review of page and HTML Code Sniffer results.

**Discussion:** For the purpose of this initial evaluation, testing was performed through manual execution of HTML Code Sniffer on individual pages. Results for both the Section 508 and WCAG AA errors and warnings were combined in the page counts.

For the purposes of this assessment, we examined a representative sample of pages, as a number of pages share the same templating. However, we would expect the overall trend to be similar in scale and actual findings if every page were to be examined.

There were some common errors and warnings found in the header and footer of the application. As a result, these errors and warnings are duplicated on every page in the system. For readability, we have combined these findings in the “Error Categories” column. These errors are included in the error counts.

The four error categories that comprise “Shared header/footer errors” are:

* Image element missing alt text
* Link using anchor that does not exist
* Heading tag found with no content
* Heading structure not logically nested

Shared header/footer warnings:

* Element has position fixed, which may require scrolling in two dimensions
* Anchor element found with link content but no href, ID, or name
* Heading markup should be used if content is a heading
* Element is absolutely positioned and the background color cannot be determined. \*

Items marked with an asterisk(\*) are likely false positives in the case of this web application.

#### Accessibility Errors and Warnings

| Page | Total Errors | Total Warnings | Error Categories |
| --- | --- | --- | --- |
| Login Page | 3 | 8 | Errors:   * Shared header/footer errors * Link using anchor that does not exist * Insufficient text contrast   Warnings:   * Shared header/footer warnings * Element text placed over background image * If element is navigation, recommend list markup |
| Dashboard | 17 | 19 | Errors:   * Shared header/footer errors * Insufficient text contrast   Warnings:   * Shared header/footer warnings |
| Reports | 15 | 22 | Errors:   * Shared header/footer errors   Warnings:   * Shared header/footer warnings * Consider using a caption element to the table |
| Contacts | 18 | 22 | Errors:   * Shared header/footer errors * checkbox element without name/label * Form field without label * button without name/label   Warnings:   * Shared header/footer warnings * Consider using a caption element on the table |
| Add New Contact | 46 | 36 | Errors:   * Shared header/footer errors * Form does not contain a submit button * Text input element does not have a name * Form field should be labeled * Button with insufficient contrast   Warnings:   * Shared header/footer warnings * Label 'for' attribute contains an ID that does not exist * Element contains a potentially faulty value in its autocomplete |
| Tools - All Transactions | 26 | 18 | Errors:   * Shared header/footer errors * Duplicate id attribute * Insufficient contrast   Warnings:   * Shared header/footer warnings * Label 'for' attribute contains an ID that does not exist * Accessible name for element does not contain visible label text |
| Tools - Import/Export | 15 | 20 | Errors:   * Shared header/footer errors   Warnings:   * Shared header/footer warnings |
| Tools - First Time COH | 7 | 11 | Errors:   * Shared header/footer errors * Form does not contain a submit button * Text input element does not have a name * Form field missing label * Duplicate ID   Warnings:   * Shared header/footer warnings * Check that element has an inherited background color or image * Label 'for' attribute contains an ID that does not exist |
| Notifications | 16 | 28 | Errors:   * Shared header/footer errors * Heading tag with no content   Warnings:   * Shared header/footer warnings * Consider using a caption element to identify table data |
| Profile - Account | 16 | 53 | Errors:   * Shared header/footer errors * Button with insufficient contrast   Warnings:   * Shared header/footer warnings |
| Profile - Add User | 24 | 59 | Errors:   * Shared header/footer errors * Heading tag with no content * Text input element does not have a name * Form field missing label   Warnings:   * Shared header/footer warnings * Consider using a caption element to identify table data \* |

## AQL: Color and Contrast

**AQL***: Easily distinguishable/high contrast colors, type and interactive elements*

**Assessment:** Not Met (Marginal)

**Assessment Method:** HTML Code Sniffer execution of both Section 508 and WCAG AA criteria per page, manual review of page and HTML Code Sniffer results

**Discussion:** Only a small number of elements were identified as not being easily distinguishable or having sufficient contrast, based on HTML Code Sniffer’s checks and manual review. Elements identified included:

* The 'Report Issue' button in the site footer
* Red text present on the Login page
* Red error/warning text used across the site
* The aqua colored buttons

Given the small number and relatively low effort for fixing these items (i.e. by modifying colors and/or increasing font sizes), we recommend treating this criteria as marginally not met.

## AQL: Keyboard Navigation

**AQL:** *Functional keyboard/tab navigation*

**Assessment:** Not Met (Marginal)

**Assessment Method:** Manual testing and review

**Discussion:** Most pages have sufficient keyboard navigation with a few exceptions:

* Login page: The Show/Hide password icon cannot be activated by keyboard
* Site header menu: The Search Contacts and Search Transactions menu items cannot be focused using the keyboard
* Search Transactions page: The Filter Options left hand menu does not provide a way to expand sections with keyboard controls

Given the small number and relatively low effort for fixing these items (i.e. by modifying the specific instances identified), we recommend treating this criteria as marginally not met.

## AQL: Text Labels

**AQL***: All non-text content has a text alternative that serves the equivalent purpose.*

**Assessment:** Not Met (Marginal)

**Assessment Method:** Manual testing and review

**Discussion:** There is very little non-text content, but some of the images present are missing alt text. Specifically, the FEC logo in the upper left corner and the show/hide password icon.

Given the small number and relatively low effort for fixing these items (i.e. by modifying the specific instances identified), we recommend treating this criteria as marginally not met.

## AQL: Page Titles

**AQL***: Clear webpage title that describes their topic and function/purpose*

**Assessment:** Pass

**Assessment Method:** Manual testing and review

**Discussion:** The web page title for FECFile Online is "FECFile Online," but the title is not dynamically changed or updated during navigation. This title is descriptive of the application and technically meets the letter of the specification, given that FECFile Online is a single page application (SPA) that technically never changes to another page.

Best practices for accessibility with SPAs recommends that the page title be dynamically updated as a user navigates to other parts of the site (i.e. when they would experience a “page change”). For adherence to the spirit as much as the letter of this specification, adjusting the title text as part of navigation would align with best practices for SPAs.

# Deliverable: Deployment

**Overall Performance Standard:**

* *Code must successfully build and deploy into staging environment*

**Overall Assessment:** Not Met (Significant)

## AQL: Single Command Build

**AQL:** *Successful build with a single command*

**Assessment:** Pass

**Assessment Method:** Manual review

**Discussion:** Code deployment is being managed using Jenkins Pipelines, and there is a Jenkins Pipeline file for each of the 3 fecfile repositories being examined. These pipeline files, named Jenkinsfile, are JSON files that define the tasks necessary to run a continuous delivery pipeline using a suite of plugins. The current Jenkinsfile files build the deployment Docker images and then deploy them to a Kubernetes controlled cluster located in the Amazon Web Services (AWS) cloud. There are three target environments defined in the deployment pipeline: DEV, QA, and UAT.

Without direct access to the related AWS clusters, the team is not able to test these deployments directly. However, based on review of the scripts, these files appear to support FEC’s goal of building with a single command using Jenkins CLI.

## AQL: Cloud.gov Deployment

**AQL:** *Deployable to cloud.gov in a single command*

**Assessment:** Not Met (Marginal)

**Assessment Method:** Manual review

**Discussion:** Technically, the existing deployment scripts do not deploy to cloud.gov. However, based on the team’s review of the existing Jenkinsfile JSON files, it will be possible to modify them to deploy the Docker image to [cloud.go](http://cloud.gov/)v rather than the AWS cloud. To do this, the Dockerfile build file will need to be modified so that it meets the requirements of cloud.gov. The switch kubectl command in the Jenkinsfile that pushes out the deployment to the target environment will need to be replaced with the appropriate "cf push" command.

## AQL: Public CI/CD Scripts

**AQL**: *CI/CD scripts are public*

**Assessment:** Pass

**Assessment Method:** Manual review

**Discussion:** The CI/DC scripts are public and maintained in the Jenkinsfile present at the root of each of the three repositories.

## AQL: Branching Strategy

**AQL**: *Branching strategy is followed*

**Assessment:** Not Pass (Significant)

**Assessment Method:** Manual review

**Discussion:** The repositories are not currently using the Git Issues feature, which is a key aspect of the FEC Contributing Guidelines[[1]](#footnote-1) and the main driver for this finding. In terms of branching strategy, no SOPs or guidelines are provided, but review of the network diagram does indicate patterns of behavior: the development team generally branches off develop for sprint work, committing that work back to develop. This work is then pushed to release. As mentioned previously, there are comments referencing a vendor-maintained JIRA instance and pull requests, so this appears to have been managed through the prior development team’s Agile sprint process and JIRA. Notably, the usage of pull requests (PRs) through FEC’s GitHub seems inconsistent. Only the fecfile-ImageGenerator repository uses them consistently, and, in most cases, these PRs are used for full sprint commits.

#### Branching Overview by Repository

| Repository | Branches | Open Issues | Closed Issues | Open PRs | Closed PRs |
| --- | --- | --- | --- | --- | --- |
| fecfile-online | 3 | 0 | 0 | 1 | 1 |
| fecfile-Validate | 13 | 0 | 0 | 0 | 2 |
| fecfile-ImageGenerator | 21 | 0 | 0 | 8 | 15 |

# Deliverable: Documentation

**Overall Performance Standard:**

* *Documentation to run the application should be public*
* *Architecture and database documentation is public*
* *Tickets contain completion criteria and business logic*

**Overall Assessment:** Not Met (Significant)

## AQL: Major Functionality

**AQL***: Major functionality in the software/source code is documented. All repeated processes have a written process.*

**Assessment:** Not Met (Significant)

**Assessment Method:** Manual testing and review

**Discussion:** Documentation for high-level functionality (e.g. builds, code review, database operations/ETL, etc.) is absent in the publicly available repositories for both fecfile-Validate and fecfile-ImageGenerator. The README.md file located in the fecfile-online repository does describe the app directory structure, how to start the application, and how to build the Docker images. However, further information would be required to understand how major functionality works, how the application is architected, and how to maintain operations for the application.

## AQL: Dependencies

**AQL:** *All dependencies are listed and the licenses are documented.*

**Assessment:** Not Met (Significant)

**Assessment Method:** Manual testing and review

**Discussion:** The software package dependencies are documented for the Python APIs in the requirements.txt file and for the Angular front-end in the package.json file. This is appropriate and located in the repositories in the file structure where developers would expect to find the files. There is no license declared for any of the three repositories under review. A software license is normally declared in the README.md file of a repository.

## AQL: Inline Code Documentation

**AQL***:* *Individual methods are documented inline using comments that permit the use of documentation generation tools such as JSDoc.*

**Assessment:** Not Met (Significant)

**Assessment Method:** Manual testing and review

**Discussion:** A manual review of both the Python API and Angular front-end code base revealed only a small percentage of the functions and methods have comments that describe the use of the function, the type and description of the function parameters, and the type and description of the return variable. Functions that did have a description comment sometimes did not have the accompanying parameter descriptions and type definitions. Also, no automated documentation tools (e.g. Sphinx for Python, Compodoc for Angular/typescript) could be found in the package dependency lists for any of the reviewed repositories.

## AQL: System Diagram

**AQL***: A system diagram is provided.*

**Assessment:** Not Met (Significant)

**Assessment Method:** Manual testing and review

**Discussion:** No systems diagrams were found in any of the three repositories. The fecfile-online repository README.md file does contain a description of the repository directories and a diagram showing the connections between the front-end, back-end, and database Docker containers but with minimal information presented. No system diagram was found that describes the FEC AWS systems and how they relate to each other.

## AQL: Database

**AQL**: *Database structure and team practices are documented.*

**Assessment:** Not Met (Significant)

**Assessment Method:** Manual testing and review

**Discussion:** No entity relationship diagram describing the Postgres database schema could be found in any of the three repositories reviewed. Team practices and standards for database design/usage were not documented in the public repositories.

AQL: Product-level decisions

**AQL***:* *Product-level decisions should be documented in a public repository.*

**Assessment:** Not Met (Significant)

**Assessment Method:** Manual testing and review

**Discussion:** Product-level decisions (e.g. decisions about functionality, architecture) were not present in any of the three public repositories. Epics, user stories, and issue reports were tracked in the JIRA project used by the prior development team. This backlog does contain appropriate documentation for product-level decisions (including review and approvals by FEC SMEs), but this is not sufficient to meet the requirements outlined in the QASP.

# Deliverable: Security

**Overall Performance Standard:**

* *OWASP Application Security Verification Standard 4.0, Level 2*

**Overall Assessment:** Not Met (Significant)

## AQL: Security Vulnerabilities

**AQL**: *Code submitted must be free of medium and high-level static and dynamic security vulnerabilities.*

**Assessment:** Not Met (Significant)

**Assessment Method:** Automated scanning and manual review

**Discussion:** Given that this AQL covers both static and dynamic security testing, we have treated it as a composite of the other static and dynamic tests executed below. As shown in the following sections, significant medium and high level vulnerabilities were identified in both static and dynamic tests.

## AQL: Dependency Scan: Overall

**AQL**: *Clean tests when dependencies are scanned with an automated tool such as Snyk and npm audit for vulnerabilities.*

**Assessment:** Not Met (Marginal)

**Assessment Method:** Automated scanning with npm audit and Python Safety

**Discussion:** Using npm audit or Python Safety dependency analysis tools, we scanned the four existing code bases within the three repositories. Every repository was found to have between one and five out- of-date packages with security issues. Overall, this is within normal bounds, especially given that the applications have not been modified in a number of months. For the most part, the packages can be upgraded to later versions with little difficulty. However, there are four instances below that would require a major version upgrade, which should be reviewed and tested to avoid introducing regression issues. There is also one instance for the Javascript package Quill in which npm audit reports no available patch or update.

Each of the four code bases is discussed in detail below. Overall, we evaluate this AQL as marginally not met.

### Dependency Scan: fecfile-online/django-backend

**Assessment:** Not Met (Marginal)

The Python code in this repository was scanned with the Python tool Safety. Five packages were found to have issues: Django, Django Rest framework, Gunicorn, Jinja2, and urllbi3. All of these packages were found to have upgrades available. Notably, Gunicorn and Jinja2 would require major release upgrades, which should be reviewed before upgrading to avoid introducing regression issues.

### Dependency Scan: fecfile-online/front-end angular code

**Assessment:** Not Met (Marginal)

NPM Audit of the fecfile/front-end angular code generated sixty-five total issues. Fifty-one of these are “development dependencies” used in the development process (i.e. but are likely not to have any impact on production system security). In the table below, findings classified as “development dependencies” are broken out separately.

#### NPA Audit Results for fecfile-online/front-end

| Severity | Finding Count | Development Dependency |
| --- | --- | --- |
| High | 0 | No |
| Moderate | 10 | No |
| Low | 0 | No |
| Critical | 2 | Yes |
| High | 34 | Yes |
| Moderate | 8 | Yes |
| Low | 11 | Yes |

The 10 moderate, non-development dependency findings were generated by five distinct packages: ansi-regex, yargs-parser, xlsx, postcss, and quill. The NPM Audit tool suggested updates for all of these except for quill, where no patch is available. Further review and analysis will be needed to identify next steps for this package. This level of findings is to be expected for a code base that has not been updated in a number of months.

The two Critical severity-development-dependencies findings were introduced by the Karma NPM package, which is used as part of testing frameworks. These findings are concerning based on severity, but security issues with this package would not be expected to cause any vulnerabilities in a deployed production system.

### Dependency Scan: fecfile-Validate

**Assessment**: Not Met (Marginal)

The Python code in this repository was scanned with the Python tool Safety. Only the Gunicorn package was found to have issues. This upgrade would be a major release upgrade, and should be reviewed before upgrading to avoid introducing regression issues.

### Dependency Scan: fecfile-ImageGenerator

**Assessment**: Not Met (Marginal)

The Python code in this repository was scanned with the Python tool Safety. Five packages were found to have issues: Flask-Cors, Jinja2, urllib3, Werkzeug, and Gunicorn. All of these packages were found to have upgrades available. Notably, Gunicorn and Jinja2 would require major release upgrades, which should be reviewed before upgrading to avoid introducing regression issues.

## AQL: Static Code Analysis and Passive Scanning

**AQL**: *Clean tests when static code analysis using Bandit and passive scanning with a scanner such as OWASP ZAP along with documentation explaining any false positives*

**Assessment:** Not Met (Significant)

**Assessment Method:** Automated testing with Bandit, SonarQube, and OWASP ZAP

**Discussion:** For Python static code analysis, we utilized Bandit. Bandit identified a large number of issues in the Django-backend codebase. By occurrence, the bulk of the findings are related to how SQL commands were created, but there are a number of other categories as well. For instance, the findings related to “possible hard coded password” are verified issues. It is noteworthy that the Django ‘SECRET\_KEY’ used to secure access to the API has a hard coded default value in the settings.py file. As noted by FEC staff, secret keys are flagged as “critical setting” in Django’s deployment checklist and could lead to security issues if a production deployment was made without updating this value.

SonarQube was used to perform security focused static code analysis on the Typescript code used in the fecfile-online/front-end component. On initial inspection, the majority of the findings appear to be false positives. For example, most of the issues with hard-coded credentials are false positives because they are found in test code. Other items, such as many of the "target=\_blank links" can be determined to be false positive because the link targets are FEC controlled web pages. There are a few instances that warrant additional examination to decide what code changes, if any, are needed. While the overall score does not meet the acceptable quality level, the findings need some further review to separate out what is likely a high number of false positives, particularly in the higher severity categories.

The OWASP ZAP passive scan found five unique issues, two of which appear to be false positives. The remaining three issues are related to improperly set HTTP response headers. These issues are relatively low level-of-effort fixes.

Overall, this section has been marked as Not Met (Significant). The large number of legitimate Medium findings in the Django-backend codebase will require some significant refactoring to resolve and bring it into compliance with the acceptable quality levels. Details and counts for each repository are included below.

#### Bandit Scan Summary: Overall

| Repository | High | Medium | Low |
| --- | --- | --- | --- |
| django-backend | 0 | 430 | 13 |
| ImageGenerator | 0 | 4 | 0 |
| Validate | 0 | 0 | 0 |

#### Bandit Scan Details: django-backend

| Issue Name | Severity | Occurrences |
| --- | --- | --- |
| URL open for permitted schemes. Allowing use of custom schemes is often unexpected | Medium | 5 |
| String-based SQL construction | Medium | 375 |
| Probable insecure usage of temp file/directory | Medium | 50 |
| Using ElementTree to parse untrusted XML data is known to be vulnerable | Low | 1 |
| Possible hard coded password | Low | 9 |
| Try, Except, Pass detected. Possible unhandled exception. | Low | 3 |

#### Bandit Scan Details: fecfile-ImageGenerator

| Issue Name | Severity | Occurrences |
| --- | --- | --- |
| URL open for permitted schemes. Allowing use of custom schemes is often unexpected | Medium | 2 |
| Use of insecure MD2, MD4, MD5, or SHA1 hash function | Medium | 2 |

#### SonarQube Scan Details: fecfile-online/frontend

| Name | Risk Level | Number of Instances | Likely False Positive? |
| --- | --- | --- | --- |
| Disabling Angular built-in sanitization is security-sensitive | High | 2 | Requires more analysis |
| Hard-coded credentials are security-sensitive | High | 7 | Yes |
| Using slow regular expressions is security-sensitive | Medium | 5 | Requires more analysis |
| Links with "target=\_blank" are security sensitive | Low | 14 | Mixed |
| Using clear-text protocols is security-sensitive | Low | 1 | Yes |

#### OWASP ZAP Summary

| Name | Risk Level | Number of Instances | Likely False Positive? |
| --- | --- | --- | --- |
| X-Frame-Options Header Not Set | Medium | 1 | No |
| Absence of Anti-CSRF Tokens | Low | 4 | Yes |
| Incomplete or No Cache-control Header Set | Low | 1 | No |
| Timestamp Disclosure - Unix | Low | 1185 | Yes |
| X-Content-Type-Options Header Missing | Low | 6 | No |

**False Positive evaluations:** The “Absence of Anti-CSRF Tokens” finding appears to be a false positive because the application uses JSON Web Tokens (JWT) for authentication. Applications using JWT generally do not need Anti-CSRF tokens. The “Timestamp Disclosure - Unix” finding also seems likely to be a false positive. Using a representative sampling of the 1185 instances, the timestamps appear to be present in a list of disallowed passwords.

## AQL: Technology Up-to-Date

**AQL**: *No more than one major version behind on essential underlying technology*

**Assessment:** Not Met (Significant)

**Assessment Method:** Manual Inspection

**Discussion:** Based on code review, the team identified five major technologies as essential to the proper operation of the three applications. As seen in the table below, the Typescript and Angular versions do not meet the acceptable quality level. Of these two, the outdated Angular version is more concerning, as this is causing a number of vulnerabilities in the package dependency scan. Both the Typescript and Angular packages have some breaking changes in the newer versions. Regression testing will be needed after performing the upgrade.

#### Essential Technology Versions

| Technology | Version | Latest Version | Meets AQL |
| --- | --- | --- | --- |
| Python | 3.7 | 3.9 | Yes |
| Typescript | 2.7 | 4.4 | No |
| Angular | 6.0 | 12,2 | No |
| Django | 2.2 | 3.2 | Yes |
| Flask | 1.0 | 2.0 | Yes |

# Deliverable: User Research

*Not applicable for this assessment, per FEC guidance.*

# Deliverable: Open Source

**Overall Performance Standard:**

* *Software is open source*
* *A version-controlled, public repository of code comprising the product is the working codebase, which will remain in the government domain*
* *Default to free or open source tooling.*

**Overall Assessment:** Not Met (Significant)

## AQL: Public Repository: Working Code

**AQL:** *The public FEC repositories are the working code bases that should be updated every workday.*

**Assessment:** Not Met (Marginal)

**Assessment Method**: Manual review

**Discussion:** The commit network graph for fecfile-online shows that code commits were being bundled into branches that were assigned to a particular task. An example commit tag found in the network graph is “FNE\_2797\_shawn\_sprint42”. These tasks are branched and appear to be updated daily within a sprint before being merged back in.

However, the code in the current GitHub housed public repositories do not match branches, commits or pull requests in the GitHub UI that appear in the “Insights” network diagram. This can happen when a repository is imported into GitHub with existing commits (i.e. the pre-existing commits are only found in the network graph). This suggests that the code was being developed outside of the public GitHub repository and later imported. It appears the developers were managing code in a somewhat standard open source workflow, with working code committed daily, but utilizing separate, private repositories. As such, we evaluate this AQL as marginally not met.

## AQL: Public Repository: Issue Guidance

**AQL**: *Issues/tickets are in a public repository and follow issue guidance.*

**Assessment**: Not Met (Marginal)

**Assessment Method**: Manual review

**Discussion:** As discussed in the above section, it appears that the prior development team was managing task tickets and repository commits using open source best practices (such as linking an issue to a branch PR), but in their own private repository. All three public repositories have zero issues/tickets stored publicly.

A review of representative samples of epics, user stories, and issues tracked in the prior team’s JIRA instance (<https://team-1615471223936.atlassian.net/browse/FNE>) showed that ticket/issue structure and content were largely passable based on FEC’s issue guidance.

Given the above factors, we assess this AQL as marginally not met.

## AQL: Pull Request Documentation

**AQL**:  *Pull request documentation is followed.*

**Assessment**: Not Met (No evidence)

**Assessment Method**: Manual review

**Discussion**: As elaborated in the prior “Branching Strategy” section, pull requests were rarely used with respect to the public repository. The network diagrams appear to show that pull requests were regular parts of the team’s workflow. However, with no evidence available in the public that the team was adhering to FEC’s Contribution Guidelines, and a lack of evidence/documentation for the prior team’s internal procedures for pull requests, this factor is evaluated as not met based on a lack of evidence.

# Deliverable: Design and Usability

**Overall Performance Standard:**

* *This site should look like it’s part of a suite of FEC-branded products. The pattern library should be used as a jumping off point and the family of icons, colors, interactions, and typography should be employed as appropriate to ensure consistency across our suite of products and fit the new system.*
* *Components should be reusable.*
* *Clear user flow within the application*

**Overall Assessment:** Not Met (Significant)

## AQL: UI Design Best Practices and Alignment with FEC Pattern Library

**AQL**: Industry best practices for UI design for modern websites. UI considerations include: proper alignment of elements, proper spacing, consistency in sizing, typography, color(s) throughout the product. (Note: Based on discussion, this item was extended to include evaluation of alignment with the FEC Pattern Library.)

**Assessment**: Not Met (Significant)

**Assessment Method**: Manual review

**Discussion:** Major pages of fecfile-online were evaluated for their conformance with FEC’s pattern library. A summary of the major findings are below, with a full detailed evaluation in the appendix.

*Typography: Not Met (Significant).*  There is a general usage of Lato fonts inside the application. There are some exceptions: Arial is used on the login page, and the Report Issue button in the site footer uses Roboto. Gandhi Serif and Karla are FEC defaults.

*Colors: Not Met (Marginal).* The site does use colors consistently and only partially in alignment with FEC Pattern Library guidance. For example, some notable color exceptions are the black header, red colors for alerts/buttons, and blue link text. There are more subtle differences that are difficult to detect visually, such as the gray colors and occasionally even the black text do not match the pattern library. Of the UX pages/elements reviewed, seven were fully compliant, six were partially compliant, and five were non-compliant.

*Icons: Not Met (Marginal).* In general, there is some reuse of icons from the pattern library and/or FEC.gov. Icons that are not directly reused from the pattern library are generally stylistically consistent (with one noted exception) and an average user is unlikely to be distracted by them.

*Grid: Not Met (Significant).* The fecfile-online application uses the Bootstrap grid system in most, but not all, places where pages and components are divided, which is different from the 12 column paradigm specified in the pattern library. Further, it has not been configured to provide responsive layout for alternate screen sizes.

*Page Layouts: N/A.* The pattern library page templates are Wagtail templates used for the FEC.gov content management system, and are generated oriented for content delivery (i.e. as opposed to data entry). For purposes of pattern library conformance, this factor is not applicable. However, applicable improvements are covered in the user flow and information architecture section.

*Components: Not Met (Significant).* As outlined in the details provided in the appendix, there are a relatively substantial number of non-conformances with the pattern library on all evaluated pages and UX elements. Examples include non-conforming buttons, tables, and pagination.

## AQL: Usability Evaluation:

**AQL:**

* *Clear user flow and information architecture*
* *Clear integration of user research and usability best practices*
* *Friendly, easily understood and actionable content.*

**Assessment:** Not Met (Significant)

* Pass: 3
* Not Met (Partial): 2
* Not Met (Significant): 4
* N/A: 1

**Assessment Method:** Manual evaluation using Jakob Nielsen’s Ten Usability Heuristics.[[2]](#footnote-2)

### Usability Evaluation: Visibility of System Status

**Heuristic***: The design should always keep users informed about what is going on, through appropriate feedback within a reasonable amount of time.*

**Heuristic Assessment:** Not Met (Significant)

Sample Successes:

* Most pages have a heading which describes in a consistent manner the current page a user is viewing. There are some exceptions such as the Dashboard page, which can be remediated relatively easily.
* Users are notified of overdue forms when in the edit process.
* Notifications menu item includes a highlight of the number of unread items.

Sample Issues:

* The “train stop” component does not accurately reflect a user's progress through the F3X form submission process. There are states that are not reflected in the train stop, and within what appears to be forward action, the marker moves both forward and backward on the track.
  + In addition, the train stop component is not displayed at all times, even when the page is a step noted in the component. For example, it is not visible on the “Sign” page, even though “Sign” is the second to last marker on the component.
* Export Contacts screen does not provide any text content explaining the download in progress.
* Updating a contact does not provide any kind of success indication on save.

### Usability Evaluation: Match Between System and the Real World

**Heuristic:** *The design should speak the users' language. Use words, phrases, and concepts familiar to the user, rather than internal jargon. Follow real-world conventions, making information appear in a natural and logical order.*

**Heuristic Assessment:** N/A

A key component of this evaluation factor is that the system uses language and design familiar to the users of the system. The abundance of terminology of art (e.g. “Conduit Earmarks”) utilized in the data entry forms poses a challenge in evaluating this factor. In many cases, this terminology is well supported by help text and pop-up boxes. However, in several discussions, FEC staff have expressed some concerns about nomenclature (e.g. whether “Contacts” is an accurate name for the organizations, individuals, and other entities tracked by the application) and layout (e.g. whether the layout of the data entry fields realistically matches how users would process the data). However, without further user research, documentation, or stakeholder reviews, this factor is difficult to evaluate fairly and objectively.

### Usability Evaluation: User Control and Freedom

**Heuristic:** *Users often perform actions by mistake. They need a clearly marked "emergency exit" to leave the unwanted action without having to go through an extended process.*

**Heuristic Assessment:** Pass

Sample Successes:

* Forms typically have Previous or Cancel buttons to go back to the previous page or step.
* After adding a contact, there is an option in the pop-up menu to 'trash' that contact.
* Unsubmitted forms may be sent to, or taken out of, the recycle bin before submission.
* Web browsers do support some limited undo and redo functionality for users who are familiar with that feature, and the application does not prohibit or impair that usage.

Sample Issues:

* The delete option is not always available for business rule reasons, for example if a contact has been used in a form. In these cases the delete menu item is not available.

### Usability Evaluation: Consistency and Standards

**Heuristic:** *Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform and industry conventions.*

**Heuristic Assessment:** *Not Met (Significant)*

Sample Successes:

* Overall internal consistency is high.
* Reasonably consistent terminology usage.
* Patterns utilized by the data entry forms are internally consistent across all data entry forms.
* Patterns utilized by tabular pages (e.g. All Transactions, Reports) are relatively consistent internally.

Sample Issues:

* Significant deviations (as previously noted) with FEC brand guidance and pattern library.
* Prominent UX elements diverge significantly from what users might expect based on other websites and applications (e.g. the previously noted “train stop” component’s inconsistencies; the doubled navigation bars for My Forms and the navigation within the forms).
* The activity flow within the Forms alternates between horizontal (e.g. entering data and saving) and vertical (e.g. viewing all transactions, submitting).
* The Dashboard implies action, but none of its elements are actionable directly from the page.

Without further user research to reconcile these findings, we assess this overall heuristic as significantly not met.

### Usability Evaluation: Error Prevention

**Heuristic:** *Good error messages are important, but the best designs carefully prevent problems from occurring in the first place. Either eliminate error-prone conditions, or check for them and present users with a confirmation option before they commit to the action.*

**Heuristic Assessment:** Pass

Sample Successes:

* Significant actions such as recycling a contact require confirmation rather than a single menu click.
* Form submission generates a Certification dialog where users must confirm the action.
* When users filling form fields enter errors in the format or content of the input, a message is displayed immediately to the user immediately adjacent to the field.

Sample Issues:

* Pages often lack explanatory or introductory information.

### Usability Evaluation: Recognition Rather Than Recall

**Heuristic:** *Minimize the user's memory load by making elements, actions, and options visible. The user should not have to remember information from one part of the interface to another. Information required to use the design should be visible or easily retrievable when needed.*

**Heuristic Assessment:** Not Met (Partial)

Sample Successes:

* The “Recently Saved Reports” on the Dashboard page provides a reminder of what a user was working on in previous sessions.
* Both the “My Forms” navigation (i.e. far left hand menu) and the Form 3X make generous use of tooltips to assist users with contextual information and definitions of potentially unfamiliar terms.
* The application maintains a user’s Contacts, which allows for a user-specific auto-suggest to help with data reconciliation.
* Centralized Notifications screen provides a way for users to receive all their reminders in one place.
* In general, icons are labelled with equivalent text.
* Search results are presented with their filters.

Sample Issues:

* Information icons are not used consistently on every form field.
* The content for many of the information icons is a "request language from RAD" placeholder.
* In some instances, icons are used without context and without hover text (e.g. collapsed left nav, the “carrot” to expand transactions in the Contacts screen).

### Usability Evaluation: Flexibility and Efficiency of Use

**Heuristic:** *Short cuts—hidden from novice users—may speed up the interaction for the expert user such that the design can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.*

**Heuristic Assessment:** Not Met (Partial)

Sample Successes:

* The “Reports” page allows for a number of actions directly from the page.
* The application supports keyboard navigation when completing forms.
* The left hand common forms sidebar still shows form names when collapsed to allow quick access to move to those forms for experienced users.
  + *Note: After discussion, FEC staff have noted that this feature may not actually improve usability, and would greatly benefit from usability testing, which we concur with.*
* The “All Transactions” page (e.g. on a Form 3X) allows power users to efficiently review, edit, clone, and delete transactions.

Sample Issues:

* We were unable to find shortcuts or quick actions for what would appear to be common actions such as opening a Form 3X for the current month or adding a new contact.
* The horizontal “kebab” menu (three dots) shown on many summary tables, such as on the “Contacts” or “Reports” pages, often hides only 2 or 3 items. There is often sufficient space to show those icons directly, which would both be more informative to users and require fewer clicks to execute.
* Dashboard items are static, not clickable/drillable, and require navigation to the “Reports” screen or to the left hand navigation.

### Usability Evaluation: Aesthetic and Minimalist Design

**Heuristic:** *Interfaces should not contain information which is irrelevant or rarely needed. Every extra unit of information in an interface competes with the relevant units of information and diminishes their relative visibility.*

**Heuristic Assessment:** Not Met (Significant)

Sample Successes:

* “My Forms” left hand sidebar is minimized when actively working on a form or contacts.
* On the “Contacts” page, the filter sidebar is hidden until the filter button is pressed.
* In several instances (e.g. the “Reports” page), different results are “bucketed” and can be flipped between (e.g. Current Year reports and Archives are separated).

Sample Issues:

* On the “Search Transactions” page, the filter bar is shown by default taking up a significant space.
* On the “Search Transactions” page, the Receipts/Disbursements/Loans/Other buttons take up significant space. That functionality could either be included in the filter sidebar, added as a table column, or simply reduced to smaller radio buttons to allow more space and focus on the data.
* “Transactions Categories” sidebar remains expanded when editing individual receipts.
* Many of the “Import…” pages are long text pages with buttons and hyperlinks embedded throughout the text.
* The application does not use space well when displaying in alternate screen sizes and orientations.

### Usability Evaluation: Recognize, Diagnose, and Recover from Errors

**Heuristic:** *Error messages should be expressed in plain language (no error codes), precisely indicate the problem, and constructively suggest a solution.*

**Heuristic Assessment:** Pass

Sample Successes:

* Errors, warnings, and success messages we have encountered are in clear language.

Sample Issues:

* When deleting or undeleting a contact, the ID number is shown in the confirmation box rather than the contact name.

### Usability Evaluation: Help and Documentation

**Heuristic:** *It is best if the design does not need any additional explanation. However, it may be necessary to provide documentation to help users understand how to complete their tasks.*

**Heuristic Assessment:** Not Met (Significant)

Sample Successes:

* Information icons are present in a relatively large number of places.

Sample Issues:

* Information icons are present in some places, but not consistently. In some cases, they show placeholder text rather than actual content.
* The Help menu item is generally not functional. Selecting the User Guide, Glossary, or NextGen System Help does not show any information.
* Instruction text is not present on forms and other pages where users are required to make decisions or provide information.
* Context sensitive Help text or links are not present. For example, at a minimum, a link to existing instructions on the Form 3x would be extremely helpful for users when they are in the process of completing F3X online.

# Deliverable: Cloud-native application

**Overall Performance Standard:**

* *Best practices for modern cloud native applications are followed.*

**Overall Assessment:** Not Met (Significant)

## AQL: Twelve Factor Application Framework

**AQL:** *12-factor application framework is followed: https://12factor.net/*

**Assessment:** Not Met (Marginal)

**Assessment Method:** Manual review

**Discussion:** Overall, the application generally conforms to the majority of criteria set out by the 12 Factor framework. Six of the factors pass, with only two clear non-compliances, one partial, and three items requiring further investigation.

#### 12 Factor Assessment

| 12 Factor | Findings | Compliant |
| --- | --- | --- |
| Codebase | The code is currently split up into 3 repositories: fecfile-online, fecfile-Validate, and fecfile-ImageGenerator. This code grouping is not optimal as the independent Python backend and Angular frontend should be split into separate repos. Also, it would be advantageous to merge the fecfile-online backend and the fecfile-Validator repos since both can share the same code base that defines the validation rules for the forms and individual form fields. | No |
| Dependencies | The repos correctly capture the programming language supporting libraries in the repos by using the Python requirements.txt file and the Node package.json file, | Yes |
| Config | The 12-factor recommendation is to have fully independent environment variables defined in each environment that are then referenced by the app. This includes not defining environments such as “dev” or “prod” in configuration files. Examples breaking this policy can be found in the code base. One example is in fecfile-ImageGenerator/config.py where app variables are set with “dev” references. Also, the hardcoding of values in the environments file in fecfile-online/front-end/src/environments breaks this principle. | No |
| Backing services | Resources consumed by the app need to be removed or replaced without code changes. To meet this requirement, references and needed configuration settings should be placed in system environment variables. This best practice is related to #3 Config, and examples in the code base that do not conform can be found in the files referenced there. | No |
| Build, release, run | The current build process as managed by the Jenkinsfile in each repo creates a unique release ID and then merges in configuration values when creating the Docker image, which is then used as the release. This process conforms to the factor best practices. | Yes |
| Processes | The backend applications meet this criteria as they do not use any local memory or filesystem resources to maintain state between user requests. Session information is handled by the framework session manager and backend database. | Yes |
| Port binding | Port binding is handled in the fashion recommended by 12 Factors via the Docker runtime environments and their port binding. | Yes |
| Concurrency | A review of the code has not resulted in the discovery of code that creates, manages, or removes local processes such as daemons or local permanent files that would prohibit horizontal scaling of the runtime Docker images. However, no scaling mechanisms (e.g. load balancing) are explicitly called for or set up. | Partial |
| Disposability | Long running data validation and PDF image generating processes do not handle processing interruption gracefully when interrupted and not completed if a sudden system shutdown occurs. | No |
| Dev/prod parity | Use of the Docker images built with the same process for all hosting environments (i.e. dev, qa, stage, prod) that use the same backing services and same versions complies to this Factor best practice. | Yes |
| Logs | The Python-based backend applications all use the Python logging package which, by default, logs messages as a stream to the console (STDERR) and can be rerouted by the system if necessary. This conforms to the 12 Factor best practices. | Yes |
| Admin process | All code, both admin and application, are housed within the same repo. No additional code sources have been reference or located. | Yes |

## AQL: Application Code Base

**AQL:** *Applications each have their own codebase*

**Assessment:** Not Met (Significant)

**Assessment Method:** Manual review

**Discussion:** As specified in the 12 Factor review, fecfile-online/front-end and fecfile-online/django-backend could legitimately be considered separate applications. These are currently kept together in a single repository.

## AQL: Environment Configuration

**AQL:** *Config is stored in the environment.*

**Assessment:** Not Met (Significant)

**Assessment Method:** Manual review

**Discussion:** As specified in the 12 Factor review, the cloud native recommendation is to have fully independent environment variables defined in each environment that are then referenced by the application. However, this pattern is not followed.

## AQL: Database Design and Best Practices

**AQL:** Database design/structure best practices are documented and followed.

**Assessment:** Not Met (Significant)

**Assessment Method:** Manual review

**Discussion:** As specified in the “Documentation” review, database design, architecture, and patterns are not documented in the repository.

## AQL: Model-View-Controller and Application ORM

**AQL:** *Model-view-controller (MVC) architectural model and application ORM are used to interact with database*

**Assessment:** Not Met (Significant)

**Assessment Method:** Manual review

**Discussion:**

The fecfile-online/front-end is developed with Angular and closely adheres to a MVC architectural model--which is expected for an Angular-based application, as Angular enforces this architectural pattern.

(Note: We have chosen to interpret “MVC” as a general architecture model. The developers of Angular characterize it as a Model-View-ViewModel (MVVM) pattern, but it adheres to the same principles of separation between the presentation, logical, and data layers that characterize MVC, which we believe to be the intent of this AQL.)

We did not evaluate fecfile-online/django-backend, ImageGenerator, and Validate in terms of MVC implementation. Given that these three components are largely REST APIs, we considered a full MVC evaluation not applicable because there is no presentation layer to separate out.

In terms of Object Relational Mapping (ORM), the fecfile-online/django-backend does not make proper use of an ORM. While the Django framework is used to build models, there is much more interaction with the database through the use of SQL strings than using the Django ORM. As such, we would assess ORM usage as significantly not met.

The fecfile-online/front-end, ImageGenerator, and Validate code bases do not require a database connection and, therefore, ORM does not seem applicable.

In summary, while the fecfile-online/front-end utilizes MVC appropriately, the fecfile-online/django-backend does not implement ORM. As a result, this AQL is assessed as significantly not met.

## AQL: Object-Oriented Programming

**AQL**: Software is built using object-oriented programming.

**Assessment:** Not Met (Significant)

**Assessment Method:** Manual review

**Discussion:** The four separate code bases (fecfile-online/django-backend, fecfile-online/front-end, ImageGenerator, and Validate) utilize object-oriented programming (OOP) to substantially different degrees. Two of the applications (ImageGenerator and Validate) do not use OOP at all. And while OOP is utilized in both fecfile-online/front-end and fecfile-online/django-backend, extending its usage would likely lead to benefits in terms of maintainability. As such, the overall assessment for this AQL is significantly not met. A more detailed assessment of each repository is provided below.

The fecfile-online/django-backend Python code does make use of some classes, but a significant amount of code is not contained in a class. This by itself may not necessarily cause quality issues, but there are instances of code duplication that could be removed with better use of class inheritance (or proper functional architecture).

The fecfile-online/front-end web application is written using an OOP paradigm. This is expected given that the Angular framework is itself object-oriented. While almost all code in this application is part of an object, static code analysis identified 22% code duplication. With Angular, it would not be uncommon for static code analysis to pick up some unavoidable duplication of boilerplate code. Unfortunately, review of some of this duplicated code does show duplication of actual logic (i.e. non-boilerplate code). We would expect proper object modeling to reduce the 22% of duplicated code to a more manageable number.

The fecfile-ImageGenerator Python code does not use OOP. Code review of the application does highlight several instances in which code appears to be cut and pasted from one file to another. A more object-oriented approach or a better architected functional approach would provide significant improvements in maintainability.

The fecfile-Validate Python code does not use an object-oriented structure. This application is quite simple and likely would not gain significant benefits from rework as an object-oriented system. Nonetheless, it does not meet the AQL as currently implemented.

# Deliverable: Performance

*Not applicable for this assessment, per FEC guidance.*

# Deliverable: Mobile Accessible

*Not applicable for this assessment, per FEC guidance.*

# Deliverable: Accepted Features

*Not applicable for this assessment, per FEC guidance.*

# Assessment, Part 2: Feasibility of Integrating Previously Developed Code

At the application level, the overall architecture is workable and consists of production-level, industry-standard frameworks. Both the front-end Angular framework and the back-end Django-Postgres combination are widely supported, extensible through a variety of add-on packages, and generally provide the features needed to build maintainable applications quickly.

Both Angular and Django are “batteries included” frameworks, meaning they come with a host of features and services (e.g. HTML templating, session management, authentication, etc.) that are integrated parts of the framework core (i.e. as opposed to third party, add-on modules). As such, our overall recommendation for moving forward with the three applications is to:

* refactor the code where necessary, with an emphasis on consolidation to reduce duplicate code;
* leverage the native capabilities of the frameworks to reduce complexity;
* simplify code through object-oriented design;
* and make overall improvements for performance and stability.

By focusing our efforts on refactoring, we can address technical deficiencies in the existing applications, while preserving the business logic that has been captured for the existing forms, fields, and their associated business rules and processes.

While further work needs to be done in several areas—more granular specification and prioritization of the refactoring work, further user research and design on the user experience, validation of existing features and user stories—we believe that a significant portion of the existing application can be reused and repurposed.

## Notes and Assumptions:

* Feasibility assessment is based on the assumption that functional/business logic and specifications are implemented as intended.
* The findings below are grouped for readability and do not represent prioritization or a roadmap.
* Examples for each finding are illustrative and not intended to be exhaustive.
* Further analysis and specification need to be done to make these findings actionable for development.

## 

## Software Frameworks

| Application | Framework | App Version | Current Version |
| --- | --- | --- | --- |
| fecfile-Validate | Python Flask | 1.0.0 | 2.0.2 |
| fecfile-ImageGenerator | Python Flask | 1.0.2 | 2.0.2 |
| fecfile-online | Python Django | 2.2.13 | 3.2.8 |
| frontend | Angular | 6.0.3 | 12.2.12 |

## Recommendations by Category:

| Reduce Code Duplication |
| --- |
| fecfile-Validator: Migrate from Flask to Django and fecfile-online to leverage validation code used on ORM models. The validation code on the fecfile-online models can be the canonical source of validation before values are saved to the database. [See: [fecfile-Validate/api/rules/\*.json](https://github.com/fecgov/fecfile-Validate/tree/master/api/rules)] Alternatively, place fecfile-online model ORM models and validation code into their own repo to be utilized by both the fecfile-Validator and fecfile-online projects. |
| fecfile-ImageGenerator: Refactor schedules code to pull redundant code snippets into a library or base class. [See: [https://GitHub.com/fecgov/fecfile-ImageGenerator/blob/master/routes/src/schedules/sh1\_schedule.py#L88:L110](https://github.com/fecgov/fecfile-ImageGenerator/blob/master/routes/src/schedules/sh1_schedule.py#L88:L110) and [https://GitHub.com/fecgov/fecfile-ImageGenerator/blob/master/routes/src/schedules/sh2\_schedule.py#L55:L77](https://github.com/fecgov/fecfile-ImageGenerator/blob/master/routes/src/schedules/sh2_schedule.py#L55:L77)] |
| fecfile-online/front-end: Refactor out code duplication by creating parent classes or reusable functions containing the common code. [See: [fecfile-onine/font-end/src/app/contacts/service/contacts.service.ts lines 70-79](https://github.com/fecgov/fecfile-online/blob/main/front-end/src/app/contacts/service/contacts.service.ts#L70-L79)  [fecfile-onine/font-end/src/app/sched-c/loan-summary/service/loan-summary.service.ts lines 59-68](https://github.com/fecgov/fecfile-online/blob/main/front-end/src/app/forms/sched-c/loan-summary/service/loan-summary.service.ts#L59-L68)] |

| Reduce Code Complexity by Leveraging Framework Features or 3rd Party Packages |
| --- |
| fecfile-ImageGenerator: Migrate from Flask to Django to maintain consistency in frameworks used and leverage the Django PDF outputting module: <https://docs.djangoproject.com/en/3.2/howto/outputting-pdf/> |
| fecfile-online/django-backend: Create Django ORM models and utilize the Django migrations feature to create and manage all database entities. [See: [fecfile-online/django-backend/fecfiler/sched\_A/models.py](https://github.com/fecgov/fecfile-online/blob/main/django-backend/fecfiler/sched_A/models.py)] |
| fecfile-online/django-backend: Eliminate Postgres functions and move that functionality into the Django model ORM framework. |
| fecfile-online/django-backend: Migrate database-calling functions located in schedule views that can be handled by the schedule ORM model. [See: [fecfile-online/django-backend/fecfile/sched\_A/views.py lines 245-256](https://github.com/fecgov/fecfile-online/blob/main/django-backend/fecfiler/sched_A/views.py#L245-L256) and [fecfile-online/django-backend/fecfile/sched\_A/views.py lines 432-470](https://github.com/fecgov/fecfile-online/blob/main/django-backend/fecfiler/sched_A/views.py#L432-L470)] |
| fecfile-online/front-end: Change complicated forms whose inputs are managed in HTML to Angular Dynamic Forms: <https://angular.io/guide/dynamic-form> [See: [fecfile-online/front-end/src/app/forms/form-3l/f3l/form-entry/form-entry.component.html](https://github.com/fecgov/fecfile-online/blob/main/front-end/src/app/forms/form-3l/f3l/form-entry/form-entry.component.html#L86-L201)] |
| fecfile-online/front-end: Use modules to break code up into feature units that can be downloaded to the browser with “lazy-loading” so that the app is more responsive: <https://angular.io/guide/lazy-loading-ngmodules> |
| fecfile-online/front-end: Create Angular HTTP interceptor to centralize credentials into one place. <https://angular.io/api/common/http/HttpInterceptor> [See: [https://GitHub.com/fecgov/fecfile-online/blob/main/front-end/src/app/admin/manage-user/service/manage-user-service/manage-user.service.ts#L23:L28](https://github.com/fecgov/fecfile-online/blob/main/front-end/src/app/admin/manage-user/service/manage-user-service/manage-user.service.ts#L23:L28)] |
| fecfile-online/front-end: Replace custom table components with sorting functionality with table components from the Angular Material library. <https://material.angular.io/components/table/overview> [See: [https://GitHub.com/fecgov/fecfile-online/blob/main/front-end/src/app/admin/manage-user/manage-user.component.html#L208-L233](https://github.com/fecgov/fecfile-online/blob/main/front-end/src/app/admin/manage-user/manage-user.component.html#L208-L233)] |
| fecfile-online/front-end: Leverage use of Angular Material components to simplify and enhance UX. <https://material.angular.io/components/categories> |

| Simplify Code by Capturing Relationships Through Object-Oriented Design Patterns |
| --- |
| fecfile-Validator: Replace lists of forms with similarities with object-oriented based classes that manage common features across form types. [See: [fecfile-Validate/api/validate.py lines 15-79](https://github.com/fecgov/fecfile-Validate/blob/master/api/validate.py#L15-L79)] |
| fecfile-online/django-backend: Create Django View classes for each schedule and generalize properties into a base class that are currently hard coded as custom constants. [See: [fecfile-online/django-backend/fecfile/sched\_A/views.py lines 198-204](https://github.com/fecgov/fecfile-online/blob/main/django-backend/fecfiler/sched_A/views.py#L198-L204) and [fecfile-online/django-backend/fecfile/sched\_B/views.py lines 105-110](https://github.com/fecgov/fecfile-online/blob/main/django-backend/fecfiler/sched_B/views.py#L105-L110)] |
| fecfile-online/django-backend: Eliminate code duplication among schedule views by refactoring common code into either a package library or base object-oriented class. [See: [fecfile-online/django-backend/fecfile/sched\_A/views.py lines 283-300](https://github.com/fecgov/fecfile-online/blob/main/django-backend/fecfiler/sched_A/views.py#L283-L300) and [fecfile-online/django-backend/fecfile/sched\_C/views.py lines 106-122](https://github.com/fecgov/fecfile-online/blob/main/django-backend/fecfiler/sched_C/views.py#L106-L122)] |
| fecfile-online/front-end: Create or enhance data models that validate and manage the form data collected and sent to the back end. [See: [fecfile-online/front-end/src/app/forms/sched-c/model/loan.models.ts](https://github.com/fecgov/fecfile-online/blob/main/front-end/src/app/forms/sched-c/model/loan.model.ts)] |

| Refactor to Improve Performance and Stability |
| --- |
| fecfile-ImageGenerator: Replace active shell management of temporary files with the more robust and fault-tolerant tempfile package. [See: [https://GitHub.com/fecgov/fecfile-ImageGenerator/blob/master/routes/src/form3x.py#L173:L174](https://github.com/fecgov/fecfile-ImageGenerator/blob/master/routes/src/form3x.py#L173:L174)] |
| fecfile-online/django-backend: Increase logging statements throughout views files that can be turned on in debugging mode and facilitate troubleshooting. |

## User Experience (UX) Reusability and Accessibility/508 Remediation

As illustrated by our assessment findings in the “Design and Usability” section, the existing UX patterns have a mix of successes and areas for improvement. However, a lack of supporting user research presents a challenge in assessing reusability of patterns and front-end code. Without research to validate or refute our initial assessment, it is difficult to make recommendations as to what UX patterns and paradigms to retain for reuse—which also affects our confidence in recommendations for accessibility/508 remediation. In lieu of recommendations for reuse, then, we have provided an overview of a UX research approach for discussion and refinement with FEC below.

Our team is committed to rigorous UX research and user-driven design. This commitment leads to a more in-depth understanding of the design’s validity and effectiveness. While conducting our UX Research, we will focus on these four common stages of product development processes: Discover, Explore, Test, and Listen.

The goal of the **Discovery** stage is to surface unknowns and uncover what users may really need. We want to validate and discard assumptions by performing user interviews, stakeholder interviews, and requirement or constraint gathering tasks.

In the **Explore** stage, we hope to gain a full understanding of the problem space and how to best satisfy the user’s needs. To accomplish this, we will utilize a variety of techniques, including some (if not all) of the following:

* review the current legacy FECFile,
* build personas,
* perform journey mapping,
* conduct design reviews,
* analyze tasks, and
* perform card sorting.

These tasks will help us understand inconsistencies, consider diverse contexts, and outline the long-term implications of potential new designs.

In the **Testing** stage, we will use validation methods to help us ensure the systems we design work well for the people using them. We will use such methods as qualitative usability testing, benchmark testing, and accessibility evaluation. We live the mantra test early and often.

In the last stage, **Listen**, we will utilize testing methods like surveys, analytics reviews, search log analysis, usability bug reviews, and FAQ reviews. Listening throughout the research stage will help us understand new problems that arise and direct future efforts.

The research approach outlined above will help us define aspects of the current user experience that can be reused and help us identify areas for redesign. This research, while potentially time consuming, is critical, as we do not believe the benefits of speed-to-market outweigh the benefits of UX discovery and research.

# Appendix: Linting Errors

**Recommended Priority Key**:

* High: Likely a bug or logic error with actual impact; needs developer intervention
* Moderate: Could be related to a bug or logic error with actual impact; likely needs developer intervention
* Low: Change has semantic value but likely not an error; may need developer intervention
* Trivial: Style or readability issue but not an error; unlikely to need direct developer intervention (i.e. can be scripted or automated through tooling)

#### fecfile-online django-backend Python Code flake8 Results

| Count | flake8 Code | Description | Rec. Priority | Auto Fixable |
| --- | --- | --- | --- | --- |
| 183 | E722 | do not use bare 'except' | Moderate | No |
| 178 | F841 | local variable is assigned to but never used | Moderate | No |
| 28 | W605 | invalid escape sequence '\w' | Moderate | No |
| 11 | F811 | redefinition of unused 'datetime' from line 11 | Moderate | No |
| 5 | F523 | '...'.format(...) has unused arguments at position(s): 0 | Moderate | No |
| 2 | E402 | module level import not at top of file | Moderate | No |
| 2 | F403 | 'from .views import \*' used; unable to detect undefined names | Moderate | No |
| 1 | F524 | '...'.format(...) is missing argument(s) for placeholder(s): 1 | Moderate | No |
| 1 | F632 | use ==/!= to compare constant literals (str, bytes, int, float, tuple) | Moderate | No |
| 264 | E265 | block comment should start with '# ' | Low | Yes |
| 247 | F401 | module imported but unused | Low | No |
| 28 | E711 | comparison to None should be 'if cond is None:' | Low | No |
| 13 | E701 | multiple statements on one line (colon) | Low | Yes |
| 7 | E703 | statement ends with a semicolon | Low | No |
| 6 | E713 | test for membership should be 'not in' | Low | No |
| 3 | E712 | comparison to False should be 'if cond is False:' or 'if not cond:' | Low | No |
| 5336 | E501 | line too long (95 > 79 characters) | Trivial | Yes |
| 2811 | W291 | trailing whitespace | Trivial | Yes |
| 454 | E231 | missing whitespace after ':' | Trivial | Yes |
| 397 | E117 | over-indented | Trivial | Yes |
| 149 | E225 | missing whitespace around operator | Trivial | Yes |
| 149 | E302 | expected 2 blank lines, found 1 | Trivial | Yes |
| 112 | W293 | blank line contains whitespace | Trivial | Yes |
| 107 | E203 | whitespace before ':' | Trivial | Yes |
| 81 | E128 | continuation line under-indented for visual indent | Trivial | Yes |
| 61 | E303 | too many blank lines (2) | Trivial | Yes |
| 50 | E101 | indentation contains mixed spaces and tabs | Trivial | Yes |
| 50 | W191 | indentation contains tabs | Trivial | Yes |
| 47 | E251 | unexpected spaces around keyword / parameter equals | Trivial | Yes |
| 45 | F821 | undefined name 'get\_entities' | Trivial | No |
| 43 | E111 | indentation is not a multiple of 4 | Trivial | Yes |
| 43 | E122 | continuation line missing indentation or outdented | Trivial | Yes |
| 39 | E222 | multiple spaces after operator | Trivial | Yes |
| 33 | E261 | at least two spaces before inline comment | Trivial | Yes |
| 32 | E127 | continuation line over-indented for visual indent | Trivial | Yes |
| 23 | E262 | inline comment should start with '# ' | Trivial | Yes |
| 22 | E221 | multiple spaces before operator | Trivial | Yes |
| 20 | E201 | whitespace after '{' | Trivial | Yes |
| 17 | E202 | whitespace before ')' | Trivial | Yes |
| 17 | W292 | no newline at end of file | Trivial | Yes |
| 14 | E305 | expected 2 blank lines after class or function definition, found 0 | Trivial | Yes |
| 13 | E115 | expected an indented block (comment) | Trivial | Yes |
| 13 | E211 | whitespace before '(' | Trivial | Yes |
| 9 | E116 | unexpected indentation (comment) | Trivial | Yes |
| 9 | E124 | closing bracket does not match visual indentation | Trivial | Yes |
| 4 | E114 | indentation is not a multiple of 4 (comment) | Trivial | Yes |
| 4 | E271 | multiple spaces after keyword | Trivial | Yes |
| 4 | W391 | blank line at end of file | Trivial | Yes |
| 2 | E131 | continuation line unaligned for hanging indent | Trivial | Yes |
| 1 | E125 | continuation line with same indent as next logical line | Trivial | Yes |
| 1 | E228 | missing whitespace around modulo operator | Trivial | Yes |
| 1 | E266 | too many leading '#' for block comment | Trivial | Yes |
| 1 | E272 | multiple spaces before keyword | Trivial | Yes |
| 1 | E301 | expected 1 blank line, found 0 | Trivial | Yes |
| 1 | E304 | blank lines found after function decorator | Trivial | Yes |

#### fecfile-online frontend Typescript Code ESLint Results

| Count | Error | Rec. Priority | Auto Fixable |
| --- | --- | --- | --- |
| 3 | [Error/no-duplicate-case] | High | No |
| 768 | [Error/no-undef] | Moderate | No |
| 649 | [Error/no-unused-vars] | Moderate | No |
| 5 | [Error/no-use-before-define] | Moderate | No |
| 5 | [Error/array-callback-return] | Moderate | No |
| 3 | [Error/no-case-declarations] | Moderate | No |
| 2 | [Error/no-constant-condition] | Moderate | No |
| 1 | [Error/valid-typeof] | Moderate | No |
| 1 | [Error/no-fallthrough] - switch case fall through. | Moderate | No |
| 1 | [Error/new-cap] | Moderate | No |
| 681 | [Error/camelcase] | Low | No |
| 306 | [Error/no-prototype-builtins] | Low | No |
| 304 | [Error/prefer-const] | Low | No |
| 125 | [Error/no-useless-constructor] | Low | No |
| 117 | [Error/no-unneeded-ternary] | Low | No |
| 70 | [Error/no-empty] | Low | No |
| 48 | [Error/no-unused-expressions] | Low | No |
| 18 | [Warning/no-var] | Low | No |
| 18 | [Error/no-mixed-operators] | Low | No |
| 17 | [Error/node/handle-callback-err] | Low | No |
| 15 | [Error/eqeqeq] | Low | No |
| 12 | [Error/no-useless-return] | Low | No |
| 10 | [Error/no-useless-escape] | Low | No |
| 10 | [Error/import/no-duplicates] | Low | No |
| 5 | [Error/no-return-assign] | Low | No |
| 4 | [Error/no-array-constructor] | Low | No |
| 3 | [Error/no-template-curly-in-string] | Low | No |
| 2 | [Error/no-sequences] | Low | No |
| 1 | [Error/default-case-last] | Low | No |
| 20730 | [Error/semi] | Trivial | Yes |
| 4875 | [Error/indent] | Trivial | Yes |
| 2805 | [Error/space-before-function-paren] | Trivial | Yes |
| 1712 | [Error/quotes] | Trivial | Yes |
| 886 | [Error/quote-props] | Trivial | Yes |
| 790 | [Error/spaced-comment] | Trivial | Yes |
| 776 | [Error/no-trailing-spaces] | Trivial | Yes |
| 716 | [Error/keyword-spacing] | Trivial | Yes |
| 692 | [Error/object-curly-spacing] | Trivial | Yes |
| 639 | [Error/space-before-blocks] | Trivial | Yes |
| 630 | [Error/padded-blocks] | Trivial | Yes |
| 580 | [Error/no-multiple-empty-lines] | Trivial | Yes |
| 419 | [Error/dot-notation] | Trivial | Yes |
| 399 | [Error/key-spacing] | Trivial | Yes |
| 267 | [Error/comma-spacing] | Trivial | Yes |
| 240 | [Error/brace-style] | Trivial | Yes |
| 213 | [Error/array-bracket-spacing] | Trivial | Yes |
| 166 | [Error/comma-dangle] | Trivial | Yes |
| 123 | [Error/no-tabs] | Trivial | Yes |
| 121 | [Error/no-multi-spaces] | Trivial | Yes |
| 106 | [Error/space-infix-ops] | Trivial | Yes |
| 99 | [Error/operator-linebreak] | Trivial | Yes |
| 95 | [Error/lines-between-class-members] | Trivial | Yes |
| 74 | [Error/space-in-parens] | Trivial | Yes |
| 38 | [Error/object-property-newline] | Trivial | Yes |
| 38 | [Error/arrow-spacing] | Trivial | Yes |
| 34 | [Error/object-curly-newline] | Trivial | Yes |
| 25 | [Error/eol-last] | Trivial | Yes |
| 16 | [Error/block-spacing] | Trivial | Yes |
| 14 | [Error/multiline-ternary] | Trivial | Yes |
| 8 | [Error/prefer-regex-literals] | Trivial | Yes |
| 7 | [Error/yoda] | Trivial | Yes |
| 5 | [Error/func-call-spacing] | Trivial | Yes |
| 4 | [Error/semi-spacing] | Trivial | Yes |
| 4 | [Error/no-mixed-spaces-and-tabs] | Trivial | Yes |
| 4 | [Error/dot-location] | Trivial | Yes |
| 2 | [Error/no-whitespace-before-property] | Trivial | Yes |
| 1 | [Error/space-unary-ops] | Trivial | Yes |
| 1 | [Error/no-unexpected-multiline] | Trivial | Yes |

#### fecfile-ImageGenerator Python Code flake8 Results

| Count | flake8 Code | Description | Rec. Priority | Auto Fixable |
| --- | --- | --- | --- | --- |
| 55 | C901 | function is too complex | Moderate | No |
| 10 | E722 | do not use bare 'except' | Moderate | No |
| 6 | F632 | use ==/!= to compare constant literals (str, bytes, int, float, tuple) | Moderate | No |
| 3 | F811 | redefinition of unused function | Moderate | No |
| 86 | F841 | local variable is assigned to but never used | Moderate | No |
| 41 | F401 | module imported but unused | Low | No |
| 1 | E117 | over-indented | Trivial | Yes |
| 1 | E121 | continuation line under-indented for hanging indent | Trivial | Yes |
| 3 | E123 | closing bracket does not match indentation of opening bracket's line | Trivial | Yes |
| 2 | E125 | continuation line with same indent as next logical line | Trivial | Yes |
| 3 | E126 | continuation line over-indented for hanging indent | Trivial | Yes |
| 13 | E127 | continuation line over-indented for visual indent | Trivial | Yes |
| 70 | E128 | continuation line under-indented for visual indent | Trivial | Yes |
| 6 | E201 | whitespace after '{' | Trivial | Yes |
| 6 | E202 | whitespace before ']' | Trivial | Yes |
| 3 | E203 | whitespace before ':' | Trivial | Yes |
| 1 | E211 | whitespace before '[' | Trivial | Yes |
| 4 | E221 | multiple spaces before operator | Trivial | Yes |
| 7 | E222 | multiple spaces after operator | Trivial | Yes |
| 29T | E225 | missing whitespace around operator | Trivial | Yes |
| 70 | E226 | missing whitespace around arithmetic operator | Trivial | Yes |
| 8 | E228 | missing whitespace around modulo operator | Trivial | Yes |
| 48 | E231 | missing whitespace after ',' | Trivial | Yes |
| 5 | E265 | block comment should start with '# ' | Trivial | Yes |
| 1 | E266 | too many leading '#' for block comment | Trivial | Yes |
| 1 | E272 | multiple spaces before keyword | Trivial | Yes |
| 12 | E302 | expected 2 blank lines, found 3 | Trivial | Yes |
| 51 | E303 | too many blank lines (3) | Trivial | Yes |
| 31 | W291 | trailing whitespace | Trivial | Yes |
| 3 | W292 | no newline at end of file | Trivial | Yes |
| 56 | W293 | blank line contains whitespace | Trivial | Yes |

#### fecfile-Validate Python Code flake8 Results

| Count | flake8 Code | Description | Rec. Priority | Auto Fixable |
| --- | --- | --- | --- | --- |
| 1 | F821 | undefined name | High | No |
| 4 | C901 | function is too complex | Moderate | No |
| 1 | E713 | test for membership should be 'not in' | Low | No |
| 5 | F401 | module imported but unused | Low | No |
| 3 | F841 | local variable is assigned to but never used | Low | No |
| 1 | E117 | over-indented | Trivial | Yes |
| 2 | E126 | continuation line over-indented for hanging indent | Trivial | Yes |
| 4 | E127 | continuation line over-indented for visual indent | Trivial | Yes |
| 13 | E128 | continuation line under-indented for visual indent | Trivial | Yes |
| 7 | E222 | multiple spaces after operator | Trivial | Yes |
| 5 | E225 | missing whitespace around operator | Trivial | Yes |
| 1 | E226 | missing whitespace around arithmetic operator | Trivial | Yes |
| 40 | E231 | missing whitespace after ',' | Trivial | Yes |
| 1 | E265 | block comment should start with '# ' | Trivial | Yes |
| 15 | E302 | expected 2 blank lines, found 3 | Trivial | Yes |
| 1 | E303 | too many blank lines (3) | Trivial | Yes |
| 7 | E305 | expected 2 blank lines after class or function definition, found 0 | Trivial | Yes |
| 47 | W291 | trailing whitespace | Trivial | Yes |
| 3 | W293 | blank line contains whitespace | Trivial | Yes |

# Appendix: Alignment with UX Pattern Library Details

**Compliance Evaluation Key:**

* No: Significantly non-compliant
* Partial: Mostly compliant, with non-compliances either minor or justifiable
* Yes: Compliant

| Page / Component | Typography | Typography Notes | Color | Color Notes | Icons | Icon Notes | Grid | Grid Notes | Layouts | Layout Notes | Components | Component Notes |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Login** | No |  | No |  | Partial | No exact match with library, but style is somewhat consistent. | No | See “Discussion” in main section | N/A | See “Discussion” in main section | No | Button style does not match the pattern library (PL) |
| **Header** | No |  | No | Black background and red notification color does not match pattern library. | Partial | No exact match with library, but style is somewhat consistent. | No | See “Discussion” in main section | N/A | See “Discussion” in main section | No | Header component does not match PL colors or size. "Official site" text not present |
| **Footer** | No |  | No | Grey background color is incorrect. Text does not use the base black color. Incorrect red for Report Issues button | N/A | No icons used | No | See “Discussion” in main section | N/A | See “Discussion” in main section | No | Footer color and layout does not match PL. Not all content from PL footer is present. |
| **Left Menu** | No |  | Yes |  | Partial | No exact match with library, but style is somewhat consistent. | No | See “Discussion” in main section | N/A | See “Discussion” in main section | No | Left menu does not match the colors or text styles for the "Side Navigation" PL component. |
| **Filter** | No |  | No | Grey colors do not match pattern library | Partial | No exact match with library, but style is somewhat consistent. | No | See “Discussion” in main section | N/A | See “Discussion” in main section | No | The following items do not match the PL component. Some examples of divergence are color scheme, expand/collapse icons, checkbox styles, filter application buttons |
| **Dashboard** | No |  | Partial | Grey colors do not match pattern library | N/A | No icons used | No | See “Discussion” in main section | N/A | See “Discussion” in main section | No | Tables do not match PL table component |
| **Contacts / Search Contacts** | No |  | Partial | Grey colors do not match pattern library | Partial | No exact match with library, but style is somewhat consistent. | No | See “Discussion” in main section | N/A | See “Discussion” in main section | No | Tables do not match PL Data Table component  Pagination is not an exact match for PL, but it is similar  Buttons are not an exact match for the PL in style or hover color.  Contacts table will expand to show additional info when clicked - this is not present in the PL but further refinement should be done to make this an acceptable extension. |
| **Search Transactions /**  **Tools - All Transactions** | No |  | Partial | Grey colors do not match pattern library | Partial | No exact match with library, but style is somewhat consistent. | No | See “Discussion” in main section | N/A | See “Discussion” in main section | No | The following items do not match the PL component suggestions:  Toggle buttons do not match PL horizontal toggle buttons.  Table does not match PL Data Table  Pagination style is not an exact match for the PL |
| **Reports** | No |  | Partial | Grey colors do not match pattern library | Partial | No exact match with library, but style is somewhat consistent. | No | See “Discussion” in main section | N/A | See “Discussion” in main section | No | The following items do not match the PL component suggestions:  Toggle buttons do not match PL horizontal toggle buttons.  Table does not match PL Data Table  Pagination style is not an exact match for the PL |
| **Edit Report Pages** | No |  | Partial | Grey colors do not match pattern library | Partial | No exact match with library, but style is somewhat consistent. | No | See “Discussion” in main section | N/A | See “Discussion” in main section | No | The following items do not match the PL component suggestions:  Toggle buttons do not match PL horizontal toggle buttons.  Table does not match PL Data Table  Pagination style is not an exact match for the PL |
| **Add/Edit Transaction** | No |  | Yes |  | N/A | No icons used | No | See “Discussion” in main section | N/A | See “Discussion” in main section | No | The following items do not match the PL component suggestions:  Oversized Memo Code checkbox  Left hand transaction categories sidebar does match the Side Navigation component |
| **Add/Edit Contact** | No |  | Mostly | Grey colors do not match pattern library | N/A | No icons used | No | See “Discussion” in main section | N/A | See “Discussion” in main section | No | Drop down "Type" selector does not match the Single-Select component from the PL. |
| **Import Contacts Page** | No |  | Yes |  | Partial | No exact match with library, but style is somewhat consistent. | No | See “Discussion” in main section | N/A | See “Discussion” in main section | No | Begin Upload clickable element should be either a button or link from the PL.  Other download links are not formatted as links from the PL |
| **Import FECFile** | No |  | Yes |  | Partial | No exact match with library, but style is somewhat consistent. | No | See “Discussion” in main section | N/A | See “Discussion” in main section | No | Begin Upload clickable element should be either a button or link from the PL.  Other download links are not formatted as links from the PL |
| **Import Transactions** | No |  | Yes |  | Partial | No exact match with library, but style is somewhat consistent. | No | See “Discussion” in main section | N/A | See “Discussion” in main section | No | Begin Upload clickable element should be either a button or link from the PL.  Other download links are not formatted as links from the PL |
| **Help - Feedback** | No |  | No |  | N/A | No icons used | No | See “Discussion” in main section | N/A | See “Discussion” in main section | No | Feedback system is not like anything suggested in the PL. This item should get further review with stakeholders to verify that it will be effective. |
| **Notifications** | No |  | Partial | Grey button hover color does not pattern library | Partial | No exact match with library, but style is somewhat consistent. | No | See “Discussion” in main section | N/A | See “Discussion” in main section | No | The following items do not match the PL component suggestions:  Toggle buttons do not match PL horizontal toggle buttons.  Table does not match PL Data Table  Pagination style is not an exact match for the PL |
| **Profile - Account** | No |  | Yes |  | N/A | No icons used | No | See “Discussion” in main section | N/A | See “Discussion” in main section | No | Edit buttons do not match buttons or icons from PL |
| **Profile - Add User** | No |  | Yes |  | No | Different icons than pattern library and FEC.gov - icons style not consistent | No | See “Discussion” in main section | N/A | See “Discussion” in main section | No | The following items do not match the PL component suggestions:  Buttons do not match PL button style  Radio buttons do not match PL radio button style  Text input boxes are not a good match for inputs in the PL  Table does not match table styles from the PL |

1. “Default Contributor License Agreement.” [GitHub.com/fecgov/FEC/blob/master/CONTRIBUTING.md#branches](https://github.com/fecgov/FEC/blob/master/CONTRIBUTING.md#branches). Federal Election Commission. Last accessed October 17, 2021. [↑](#footnote-ref-1)
2. Jakob Nielsen. “10 Usability Heuristics for User Interface Design.” <https://www.nngroup.com/articles/ten-usability-heuristics/>. Nielsen Norman Group. Last updated Nov 2020. [↑](#footnote-ref-2)