**Design decisions for Restylinchpin Project**

**Python Flask**

The motivation behind using Flask instead of Django was to stay focused on the experience and flexibility by removing the overhead and the unnecessary directory structure. This also gave us more control on which components to use (eg: using TinyDB and how to interact with them).

**TinyDB**

TinyDB is a non-relational, compact and lightweight database. It has no external dependencies and the applications which are best suited to TinyDB are small applications, one like the Restylinchpin project where a traditional SQL-DB server based approach would be an overload.

**UUID**

UUID is used for assigning a unique ID to each workspace and a unique API key to each user for authentication and user access. This ID is unique and not the same as the auto-increment ID which is mostly used in relational databases. UUID is safer than the auto-increment ID as it is difficult to predict and the probability of a UUID collision is exceptionally low.

**Authentication methodology**:

API key based authentication system for authenticating user operations with workspaces and user accounts.

**Advantages**:

More convenient to use as we can easily delete, reset keys without affecting the user's account password, vulnerability is limited as well.

User has to send a basic authentication header with username and password using the **login endpoint**: /login and an autogenerated (hashed) API key is returned to the user in the response.

This API key is verified by selecting API key authentication header and passing api\_key as key, value(token received during login) for each of the API calls involving user and workspace actions to grant resource access.

**Credential management using user account**

Credentials are stored in a credentials folder linked with user account in the database. We decided to store user specific credentials instead of workspace specific credentials to facilitate easy storage of credential folder anywhere on the server and flexibility to use the same set of credentials across all the workspaces created by user. User can upload a file/ pass string to upload credentials corresponding to user account. They maybe already encrypted or unencrypted which will be encrypted using ansible vault for secure storage.

**Testing**

Used python unittest framework for unit testing functions in utility module in order to verify all utility functions are working as expected before using them in the main module.