We apply MVC pattern in our software, to enhance maintainability and extensibility. I will provide a simple scenario to explain how it works. The user wants to search cleaning agents and he chooses tags as he wants in the Main interface as view. The class CleaningAgentFetcher as controller receives the user input, fetches cleaning agents related to the tag chosen and returns a collection of cleaning agents back to main. Finally, the result of the searched cleaning agents is shown on the main interface. It should be mentioned that all of the classes designed as controller are static which means they only contains static methods and cannot be instantiated. Our motivation is that we regard these classes as tools so we want to use them freely without considering whether those objects have been constructed.

We have three highlights in our software structure. The first is how we support different languages. To realize this, we design an enumeration called LanguageType, which contains values representing different languages. In class LanguagePreference, user’s selections of interface and content language are saved in the form of LanguageType. Besides, the class InternationalString maintains a LanguageType to string map which contains content of different languages. It is directly used in class CleaningAgent and Tag, realizing the support of different languages. Finally, LanguageType is also used in getting resource bundles since those values also contain an instance of class Locale. The interface language property in class LanguagePreference is used to obtain content of different languages dynamically. It is extremely easy to support another language since the only thing needed to do is just adding another value in LanguageType, and then the new language is supported in the whole business logic layer.

The second highlight is how we save user’s preferences. Beside language preference, we also save user name and date of registration in user preference and colors for different tag types in tag preference. The class User saves all these three preferences, being responsible for serializing them and writing them into a file on disk and reverse. Under this architecture, user settings are saved permanently on disk and are applied the second time user uses the software.

The third highlight of our software is that we use JavaFX as our GUI framework. JavaFX provides many advanced and new features. The first is that it has a clear structure. Every single user interface has a main class as entry, an FXML file expressing interface, a separate controller defines how controls behave and a CSS file controlling style. Besides, JavaFX also features better look. We apply flat design in our software and assign still blue as our theme color. We also have different colors for different tag types. Moreover, JavaFX features more flexible layout, allowing us to arrange controls more freely. The sizes of many our user interfaces are adjustable thanks to this characteristic. Finally, we maintain consistency among all the user interfaces. They are designed and implemented with the same look and similar usability.