**Application Overview**

This application is a tool which will help users build their ideal optical solution, when the user first starts the application they will be prompted to enter which type of optics they would like to avail of. This will range from generic framed optics to contact lenses and contactless, ensuring a wide range of choices is vital as again to reiterate; the objective is to create the ideal optical solution which is unique to each user experience.

**Functional Requirements**

The application must implement at least four different design patterns and they all must work together in order to accomplish the given task, what this means is instead of each pattern “doing their own thing” they all must logically work together.

**User Requirements/User Interface (I’ll talk about the UI)**

**Design Pattern Explanation (Add any other patterns missing :])**

There are four design patterns which have been implemented to ensure efficiency and decrease overall workload in terms of programming in this application. Each design pattern will be looked at in detail and then the use of said design pattern will be justified.

**Factory Pattern**

The factory pattern is a simple design pattern which allows for a base template to be created (in our case optics) and can then be extended out into other classes which will inherit the base templates methods and attributes. All attributes common to each optic variant have been added to the base “Ocular” class, this is along with any methods which may be needed across all classes. The only attribute which is unique to each class is their type i.e. framed, contact & contact lenses. This not only helps to identify what type of optics have been chosen but also cuts down on the overall workload, if the factory pattern was not implemented each individual class for optics would need all attributes and methods present instead of inheriting them from the Ocular superclass.

**Abstract Pattern (you talk about this)**

**Command Pattern (button commands and that, still implementing)**

**UML Diagrams (I’ll do this)**