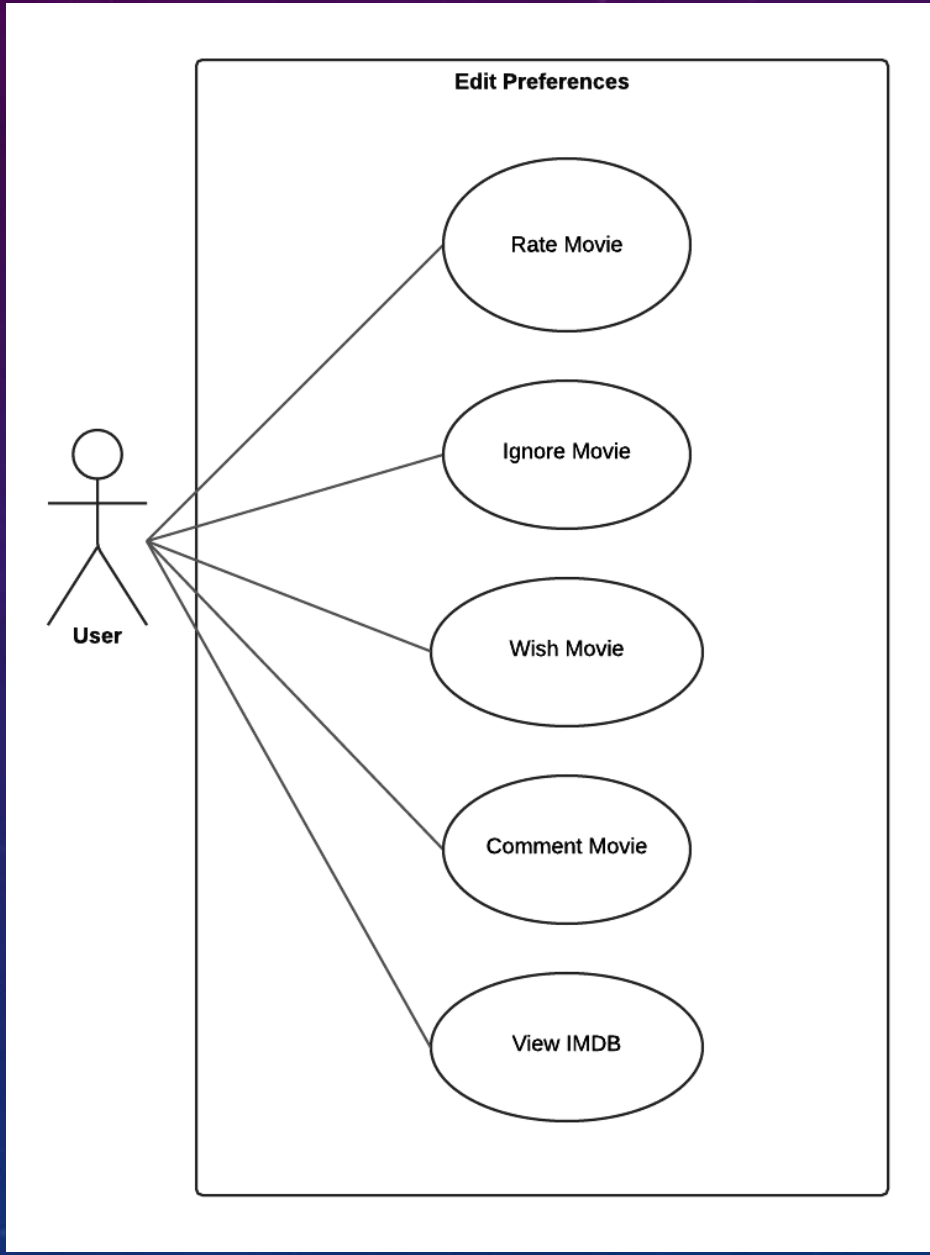
CSCI 5448 FINAL PRESENTATION

MEDIA PREFERENCE TRACKING

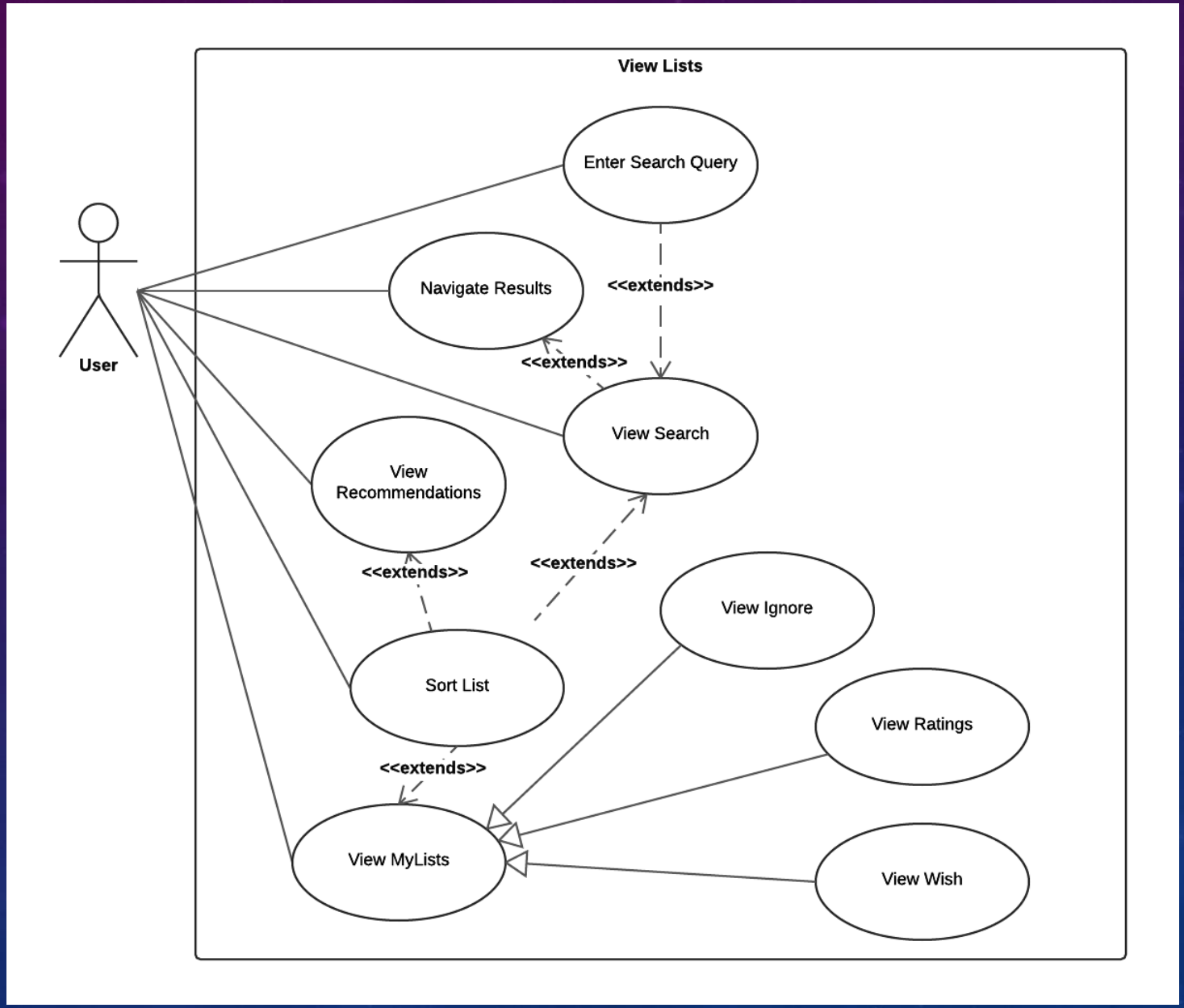
BY MICHAEL PAULY

2016.04.27

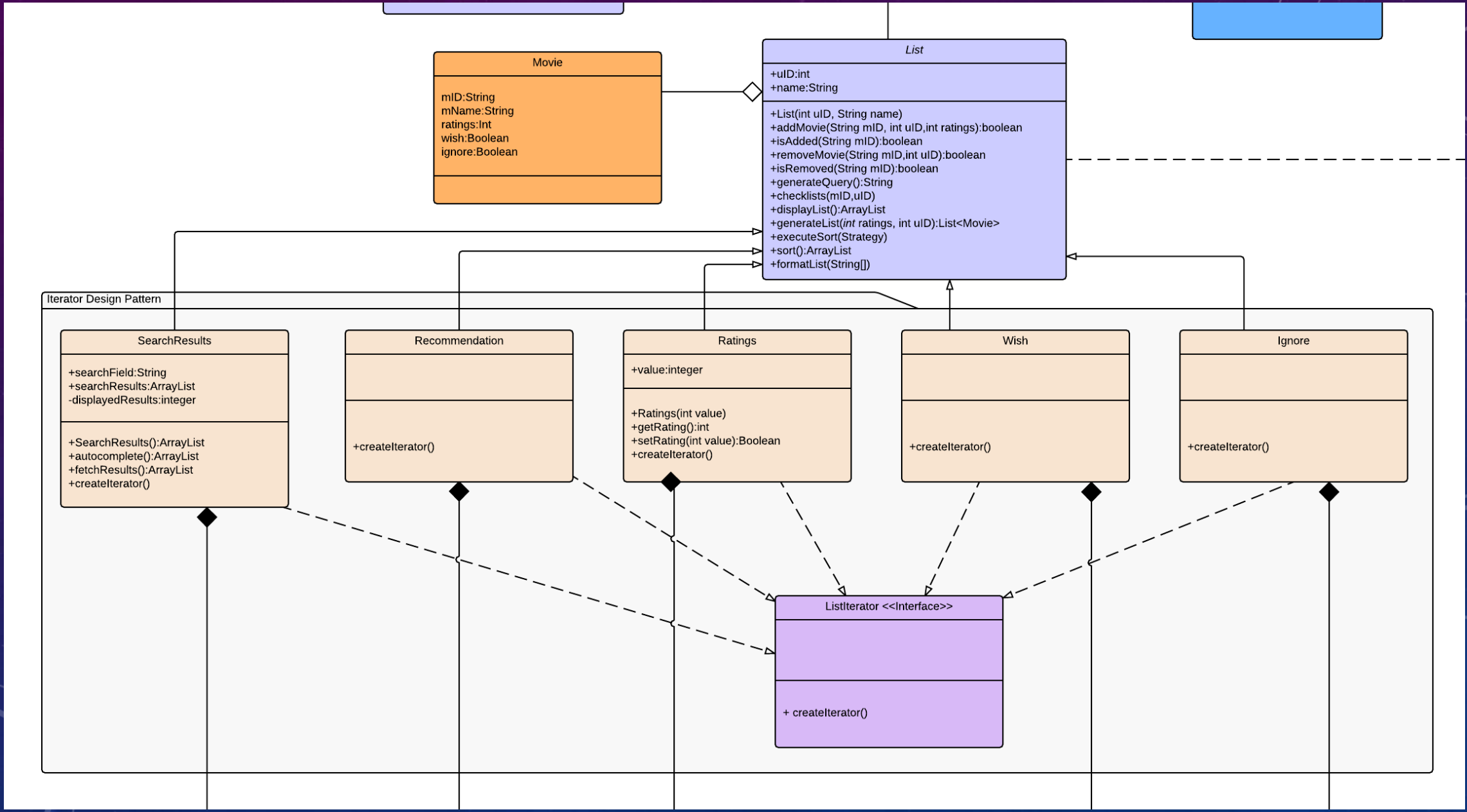
USE CASE – EDIT PREFERENCES



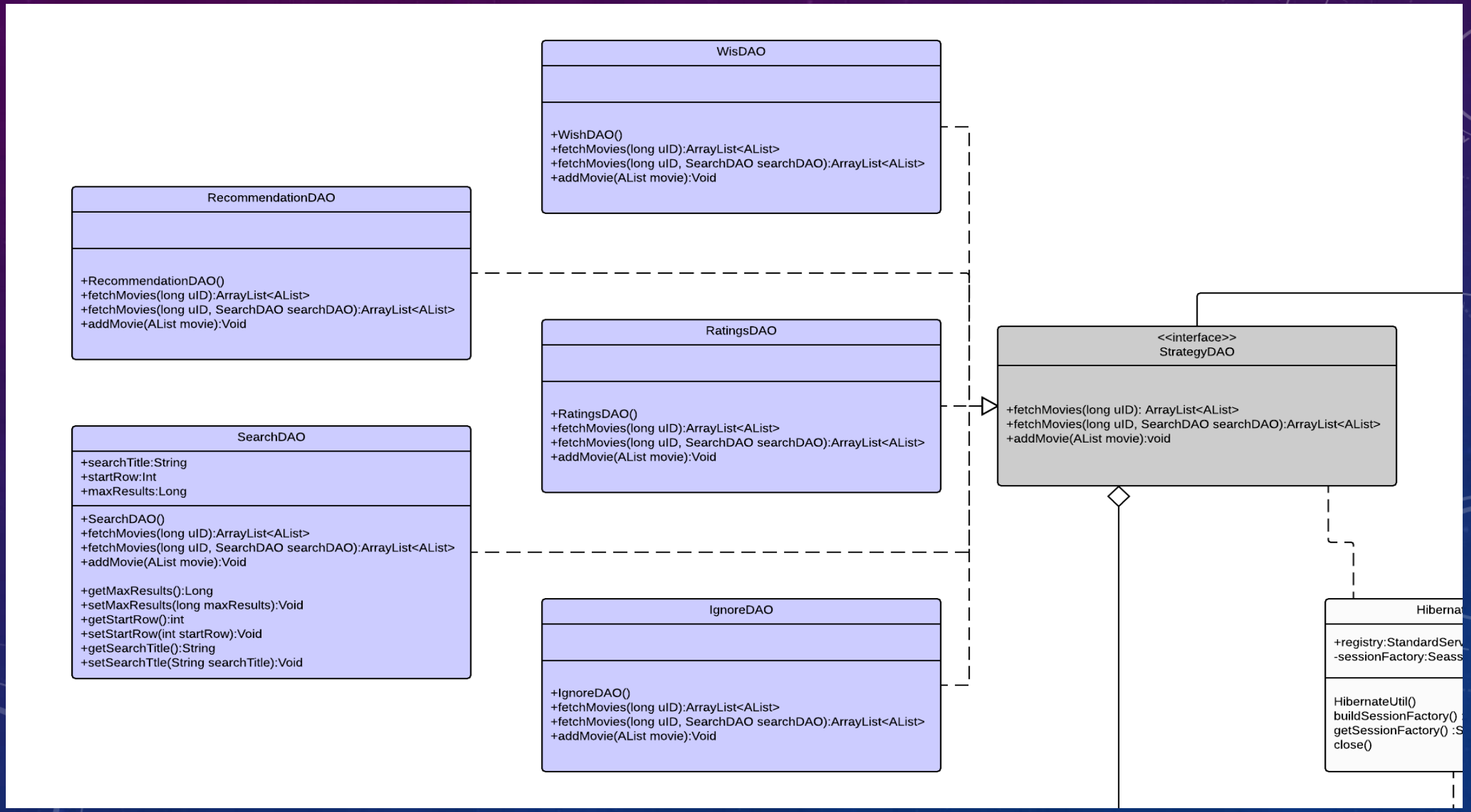
USE CASE – VIEW LISTS



DESIGN PATTERNS – STRATEGY



DESIGN PATTERNS – STRATEGY



DESIGN PATTERNS – POTENTIAL CANDIDATES

- There are a plethora of design patterns in various combinations that could have been applied to this project:
  - State – Could have handled the different views of each list
    - Filter/Sorting options will vary based on media and list
  - Prototype combined with Factory – creation of each result list
  - Template – all unique information for each list passed into basic class

DESIGN PATTERNS – POTENTIAL CANDIDATES

- Command – Undo/Redo a user editing specific movies (changing from ignore/wish/rated)
  - This could eliminate save button for each page, but also not commit changes to the database until a user navigates away from page. Proxy could be coupled with this to allow more deliberate commits.
- Decorator –
  - Page creates: List, page navigation buttons
  - List creates: Movie, sort options
  - Movie creates: Rate/Wish/Ignore buttons

DEPENDENCY INJECTION - CONSTRUCTOR INJECTION - MOVIES

- Not positive which direction would have more strength in projects current format.
  - The following example would allow for modifications to the structure of the movie object itself

```
@Service
@Entity
@Table(name="MovieName")
public class Movie
{
    @Id
    private long   mID;

    public String movieTitle;
```

```
@Service
public class MovieDAO
{
    public String movieTitle;
    public long mID;
    private Movie movie;

    @Autowired
    public MovieDAO(Movie movie)
    {
        this.movie = movie;
    }
```

DEPENDENCY INJECTION - CONSTRUCTOR INJECTION -LISTS

- This project could heavily rely upon constructor injection style Dependency Injection within the Spring framework.

```
@Service
public interface StrategyLists
{

    public ArrayList<Moviedisplayformat> fetchMovies(long uID);
    public ArrayList<Moviedisplayformat> fetchMovies(long uID, SearchDAO searchDAO);
    public void addMovie(AList movie);

}
```

```
@Service
public class WishList implements StrategyLists
{

    public long mID,uID;
    public String movieTitle;
    public boolean wish,ignore;
    public int ratings;

    MovieDAO moviedao= new MovieDAO();

    @Autowired
    public WishList()
    {   }
```

A VIDEO OF THIS PRESENTATION AS WELL AS A DEMONSTRATION OF THE SOFTWARE CAN BE FOUND HERE:

https://github.com/micscopau/CSCI5448_MediaPreferenceTracking/blob/master/MediaPreferenceTracking_Part3_MPauly.swf