Team 6
System Documentation Questions
Project 3

Question: Identify the design paradigm the team chose for your prototype and explain why you chose that paradigm.

The design paradigm that we chose for the prototype is object-oriented programming. OOP is preferred to be used with languages such as C++, Java, JavaScript, etc. We chose JavaScript, so we decided to go with OOP design paradigm. Our project is a movie recommendation application; we are using objects and classes from OOP design paradigm. We create divisions with separate classes such as “login”, “register”, “watched movies”, “need to watch”, “liked movies”, “disliked movies”, and “recommendation”. Within each class, we can go ahead and create separate objects. For example, for the “register” class, we create object inputs for “full name”, “age”, “email address”, “username”, “password”, “verify password”, and selecting “favorite genre”. Another example is in the “watched movies” class, creating an object “add” to add movies into the watched movies section. We benefit more by using OOP for our project because it makes our complex movie recommendations divided into simpler structures. We can also reuse some objects between different classes, for example “add” object can be used for “liked movies” and “disliked movies”. It is also easy to debug any issues because of the simpler structure under OOP. Classes help us keep track of the main structures in our large collection of classes. We use objects within classes to go further in defining what happens in that class. We use attributes in class to contain data. The attributes allow us to handle different data separately between objects. We also have methods to perform actions. For example, based on the data we receive from the user then we recommend a specific movie. OOP requires thinking about the structure of the programming and planning at the beginning of coding. Our team exactly planned the structure and then started coding.

Question: Identify the software architecture the team chose for your prototype and explain why you chose that architecture.

Our team is using Peer-to-Peer (P2P) Architecture for project 3. We are using P2P because we decided to have equal control over the project by sharing it under one repository and committing changes to it. The benefits of this architecture help us not lose the project if something happens to a device of one team member. Recovery is quicker, to avoid losing any files or data because it is equally shared between all peers. Since we are working as a team, we distributed tasks and workloads between each other. P2P also allows us to be suppliers and consumers at the same time because we can run our project ourselves to perform user-based tasks. P2P allows easier access to files shared between each other. We make P2P software

architecture possible by using Github to host the repository. Adaptability is easy with P2P, for example, we can easily share our repository with professors and GTAs for review or demo. P2P also doesn't fail because the files are not based on central server type architecture. P2P also offers us high efficiency because collaboration between devices can offer diverse resources and flexibility. Implementation of peer-2-peer architecture is easy; one team member creates the repo and then other team members join and start collaboration. Client-Server Architecture doesn't work for project 3 because we do not have a client-server to work with. 3-tier or n-tier architecture is preferable for project 4 because we only implement the presentation tier in project 3. Pipes-and-Filters and Event-Driven architectures fail to work with front-end and collaboration prospects of project 3. Therefore, P2P is the best architecture to be used with Project 3 Movie Recommendation.

Question: Identify the design patterns you used in your design and explain how you applied them.

In the choice of Design Patterns, Structural is the best choice for Project 3 and for going further into project 4. Structural design pattern helps in assembling objects and classes into larger structures. Structural helps in keeping the design pattern of project 3 flexible and efficient. The structure is complex for movie recommendations based on the number of classes and objects that are needed, however, structural design can help in simplifying it. For example, if we use IMBD for movie rating API, we use the "Adapter" method to make multiple classes work together to display the rating and recommend movies based on the rating. The "Bridge" method helps in separating implementation and abstraction which is the initial goal. With the "Bridge" method in structural design, we can easily implement the functional abstraction for project 3 and leave back-end implementation for project 4. Using the "Composite" method we can create more objects within objects. For example, movies that we like, or dislike can be added with the "add" object, but within that object, we need a newer object to perform the action of adding the movie. We can use the "Decorator" method to perform actions between different classes. For example, "need to watch" movies can be added to "Recommendation" based on the preferred genre. The "Facade" method can be useful in adding libraries or frameworks into a set of classes. Flyweight is very essential in project 3 and further, for project 4, it can help with the efficiency and speed of the movie recommendation. By using structural design patterns, we can face and resolve multiple concerns and issues. Creational design pattern doesn't have enough object scope, and Behavioral has unnecessary object scopes. Therefore, a Structural design pattern is the best fit.