**Features of system in prioritized order:** (1 being top priority, 10 being the lowest):

1. A user-friendly interface.

This is a top priority as the target user will require an easy to use interface for the system. Without a GUI, the system would be very complicated and hard to use for the client(s).

1. The ability for the user to view the available spaces and the times they are open for booking. This is an important feature as the core idea of the system is allow users to request bookings for a principal to view and manage them.
2. The principal having the ability to view booking requests users have sent. This is also a core feature of the system, as the principal needs to view all the requests in one place in order to make decision on managing them.
3. The principal being able to approve or deny requests. This feature is a high priority as in order to have a schedule created for the space in question, the principal must have the ability to choose which booking requests get the space at the desired time, especially in the case of multiple requests for the same space during the same time.
4. The user should have a way of allowing them to identify themselves, so the principal can see who has made the request to help his decision. (He may decide to give the coach priority for gym bookings compared to other users, for example). This is not quite as high priority as the system does not required it to perform its core functions.
5. The principal should be able to edit the times that a space is available, or if desired disable the room entirely in needed (For example in the situation of renovations/repairs).
6. The system should send users the result of their request, if it was canceled or approved.
7. The principal should have the ability to add new rooms to the system. This is not a high priority as it is possible for all the spaces in the building to be hard-coded into the system, but having the ability to add new ones to the system is good option to have.
8. The system provides the ability for users to cancel pending or previously approved requests. This feature is not high in priority as the principal can retroactively delete approved requests if needed but it is easier.
9. The system provides the ability for users to view a calendar showing their requests. This is a low priority as the system sends a notification to users (by e-mail) about their requests so it is not required in order for the user to keep track of their requests and bookings.

We decided on this priority of features as what we believe to be the core required features of the system (Being able to send and manage requests) are the most important features that are required for the system to perform what the client desires, while other features such as adding new spaces and having a personal calendar for users to view not being as important as a feature, but good options after the core features are implemented.

**Design Patterns:**

One design pattern we could potentially use for our system is the Decorator Pattern. This pattern allows for the addition of new functionality to the system without having to make large alterations, which is important in making software. In the case of our system adding new features that we come up with or new ones the client requests without having to spend large amounts of time altering other features is an important task.

Another design pattern that could be used for our system is the Facade Pattern. This pattern hides the complexity of the system and provides an interface for the client(s) to use to interact with the system without having to see the real complexity that makes the software.

The Observer Pattern is another pattern that can be used. Once one object changes, all of its dependents are notified and updated automatically. For example, once a new room is added into the system, the room and booking classes are updated automatically to allow requests.

**Key decisions:**

Some of the important decisions we made were choosing potential design patterns that we believe will allow our software to do what the client requires while also having the ability to be extended in the future with new features that the client may require without it being a large time commitment.

The reason why we made the Principal class inherit the User class is because a principal has many of the same attributes and behaviours as any user does except with added functionality. This additional functionality includes the ability to add users and rooms to the system, as well as to approve or cancel bookings requested by the users.

We decided to have User objects contain a System object because the Users are what interact with the System, so if a user requests or cancels a booking it automatically calls a method within the System class to store the requested booking inside the object. Since the principal inherits from the User class, this makes it possible for him to access the same feature, but with extended functionality of being able to see every user’s requested bookings. This applies to all other features of the System such as the cancellation or removal of bookings, and in general, the overall schedule which is contained inside the System object. In this case, any additional features to be added to the System will also be available to the users.

We also encapsulated the graphics components of the GUI inside of its own class to decouple logic and graphics flow This works better than putting it all inside the System class as this reduces overall clutter inside the project, and enables future extensibility of the system without these modifications possibly interfering with the graphics side.