Use Case 1: Logging into the System

There is a session creator, and a session manager. In tandem, these two classes facilitate the login process for a given employee. Once control is handed over to the employee object, and the actual employee actor enters his or her credentials, the employee object calls on a related session creator and its login methods (or its login class, depending on implementation). The session creator will call on the user list manager (otherwise known as the employee list controller) to check for the user ID, then the password. This means that very little data must be passed, which leads to moderately low coupling. Assuming the credentials are verified, the session creator finishes its duties by building the requisite virtual session during which the employee will perform his or her duties, and the session manager takes control of that session from within the session itself. This helps maintain high cohesion in all but the session (and session manager) class, which contain bits too small to be separated into other tiny classes.

Use Case 2: Logging out of the System

Assuming a user has completed his or her tasks and the system is not busy elsewhere, the employee/user will initiate the logout sequence. At this point, the session manager will take control of the session, making sure that all data created during the session has been written to disk. After ensuring this is true, the session manager kills the session, asks the employee manager if it is busy or if it is free, then kills the rest of the session. At that point, the employee class takes control again, starting from scratch.

Use Case 3: Adding a Patient

Assuming a user is logged in and has the necessary credentials/permissions, the session manager hands the task of adding a patient off to the patient editor. It is called with the necessary arguments, knowing it can handle things on its own from that point. As the expert, the patient editor then checks to see if the patient exists by polling the patient list controller. The patient list controller will verify that the ID doesn’t exist, and then the patient editor will create a new patient object. Once the new patient object has been created, the patient editor passes the new object to the patient list controller for insertion into the patient list, then commands the patient list controller to write the updated patient list out. Finally, once all this is done, control is returned from the patient editor to the session manager as necessitated by the design.