integration test for user interface classes in “view” folder (test whether all interfaces is created as expected)

integration test for controller classes in “dao” folder (test whether the logic of each calculating method is correct)

sees

View

notifies

Controller

update

manipulates

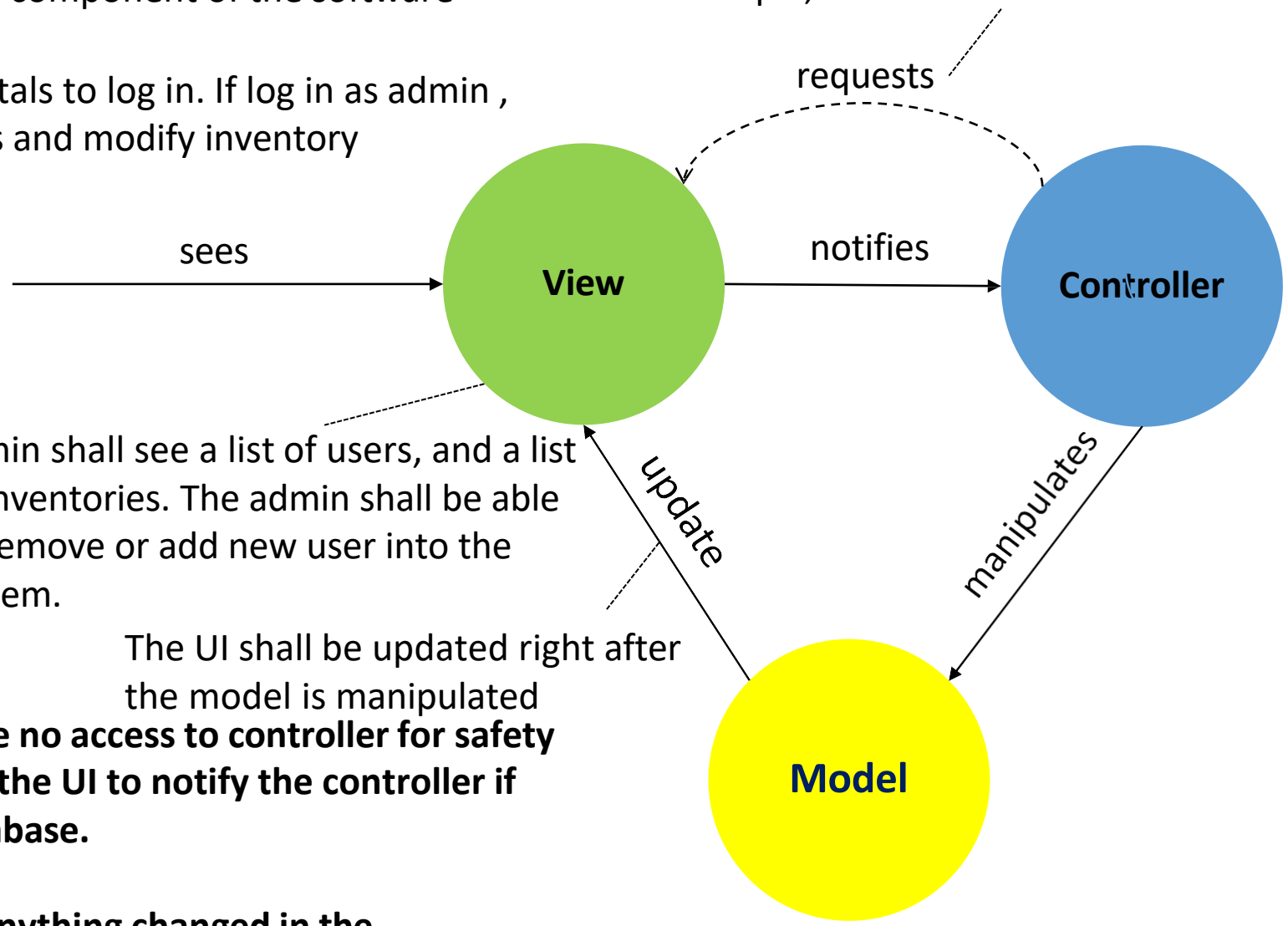
Model

integration test for classes that interacts with database in “bean” folder (test whether any modifications on database can be done successfully)

**NOTE:** we construct architecture sketch by using MVC
View represents the UI that client sees
Controller represents the internal component of the software
Model represents the database

There will be two different portals to log in. If log in as admin , he will be able to manage users and modify inventory

To be specific, controller will request the UI for further info to better locate the item. For example, which one should be removed




**As shown above, the admin have no access to controller for safety of the source code, he must use the UI to notify the controller if he wants to manipulate the database.**

**UI is always updated if there is anything changed in the database. We will design an update button on the main page so that the admin can update the page manually as well.**

e.g. pop a reminder to the client if a membership wants to enroll in a course but funds is not enough

If membership/coach/front desk/ janitor wants to log in, they will log in though user portal

sees

requests

notifies

View

Controller

their personal information including their title will be shown after they log in though user portal

update

manipulates

Model

**We divide the software into 2 parts since we only want the manger of the gym to have the power of modifying others. So, the architecture is constructed based on who is using the software. If it is the manger, he shall use the admin portal to log in. For other users including membership/coach/gym staff, they will log in through user portal.**