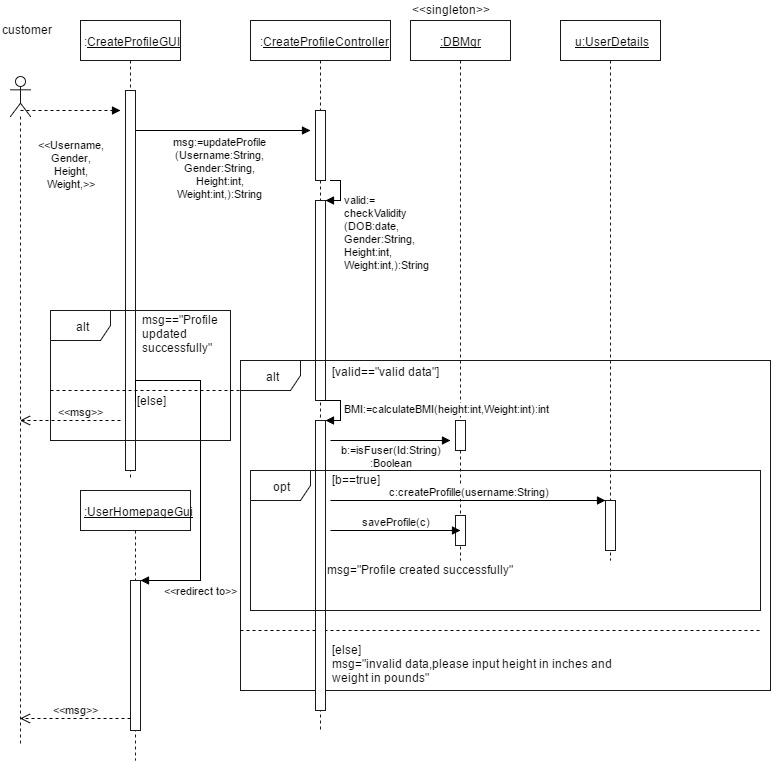
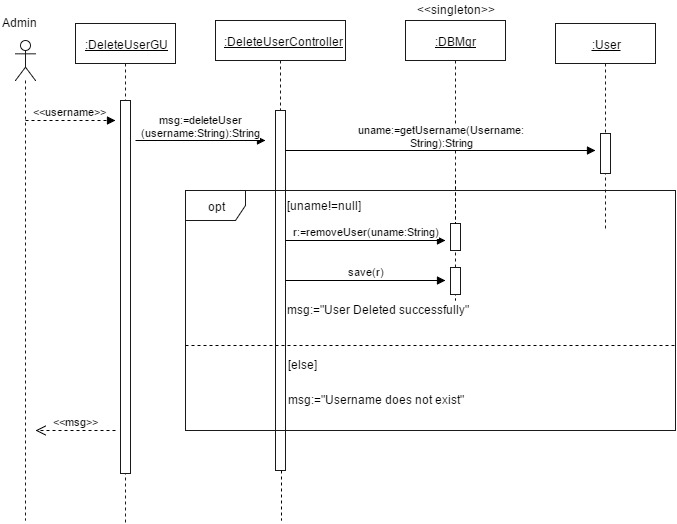
**Create Profile Sequence Diagram:**

Tradeoffs:

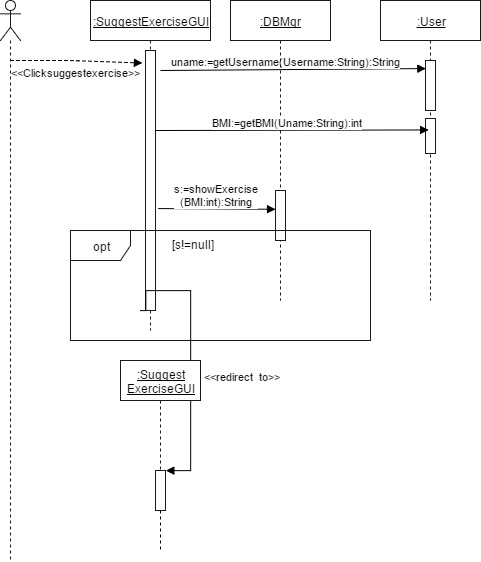
|  |  |  |  |
| --- | --- | --- | --- |
| Implement In | Advantages | Disadvantages | Conclusion |
| Controller | Maintains all the business logic. Decouples logic from the UI. | The controller is now addressing logic that is in the business layer | Can be used if the controller is light weight |
| DBMgr | Most efficient since the DB has all the information about the book being checked out | 1) We have a lot of unrelated logic in the DBMgr about various objects in the system. 2) DBMgr is prone to change everytime an object usage rule changes | For very simple verifications it might be acceptable to do this in the DBMgr (e.g. Login) if rules get more complicated move this into an object |
| UserDetails | 1) All of the processing related to the object is contained within that class. It insulates the rest of the system from change.  2) DBMgr simply becomes a series of gets/sets of data in the DB | Not as efficient as capturing this in the DB Mgr | For non-trivial processing its best to keep the logic internal to that object and use the Expert pattern. |

**Delete User Sequence Diagram:**



Tradeoffs:

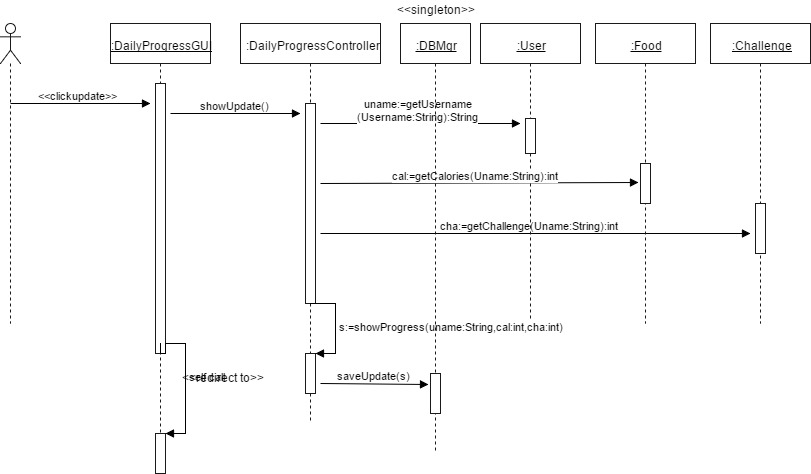
|  |  |  |  |
| --- | --- | --- | --- |
| Implement In | Advantages | Disadvantages | Conclusion |
| Controller | Maintains all the business logic. Decouples logic from the UI. | The controller is now addressing logic that is in the business layer | Can be used if the controller is light weight |
| DBMgr | Most efficient since the DB has all the information about the book being checked out | 1) We have a lot of unrelated logic in the DBMgr about various objects in the system. 2) DBMgr is prone to change everytime an object usage rule changes | For very simple verifications it might be acceptable to do this in the DBMgr (e.g. Login) if rules get more complicated move this into an object |
| User | 1) All of the processing related to the object is contained within that class. It insulates the rest of the system from change.  2) DBMgr simply becomes a series of gets/sets of data in the DB | Not as efficient as capturing this in the DB Mgr | For non-trivial processing its best to keep the logic internal to that object and use the Expert pattern. |

**Suggest Exercise Sequence Diagram:**

Tradeoffs:

|  |  |  |  |
| --- | --- | --- | --- |
| Implement In | Advantages | Disadvantages | Conclusion |
| Controller | Maintains all the business logic. Decouples logic from the UI. | The controller is now addressing logic that is in the business layer | Can be used if the controller is light weight |
| DBMgr | Most efficient since the DB has all the information about the book being checked out | 1) We have a lot of unrelated logic in the DBMgr about various objects in the system. 2) DBMgr is prone to change everytime an object usage rule changes | For very simple verifications it might be acceptable to do this in the DBMgr (e.g. Login) if rules get more complicated move this into an object |
| User | 1) All of the processing related to the object is contained within that class. It insulates the rest of the system from change.  2) DBMgr simply becomes a series of gets/sets of data in the DB | Not as efficient as capturing this in the DB Mgr | For non-trivial processing its best to keep the logic internal to that object and use the Expert pattern. |

**Daily Progress Sequence Diagram**



Tradeoffs:

|  |  |  |  |
| --- | --- | --- | --- |
| Implement In | Advantages | Disadvantages | Conclusion |
| Controller | Maintains all the business logic. Decouples logic from the UI. | The controller is now addressing logic that is in the business layer | Can be used if the controller is light weight |
| DBMgr | Most efficient since the DB has all the information about the book being checked out | 1) We have a lot of unrelated logic in the DBMgr about various objects in the system. 2) DBMgr is prone to change everytime an object usage rule changes | For very simple verifications it might be acceptable to do this in the DBMgr (e.g. Login) if rules get more complicated move this into an object |
| User,Food,Challenge | 1) All of the processing related to the object is contained within that class. It insulates the rest of the system from change.  2) DBMgr simply becomes a series of gets/sets of data in the DB | Not as efficient as capturing this in the DB Mgr | For non-trivial processing its best to keep the logic internal to that object and use the Expert pattern. |