SMU
SINGAPORE MANAGEMENT UNIVERSITY

IS442 Object Oriented Programming

Project Report

Prepared for:

Professor Lee Yeow Leong

Prepared by:

Daryl Ang Jun Hao

Job Seow Jian Liang

Tan Li Zhen

Tiffany Tan

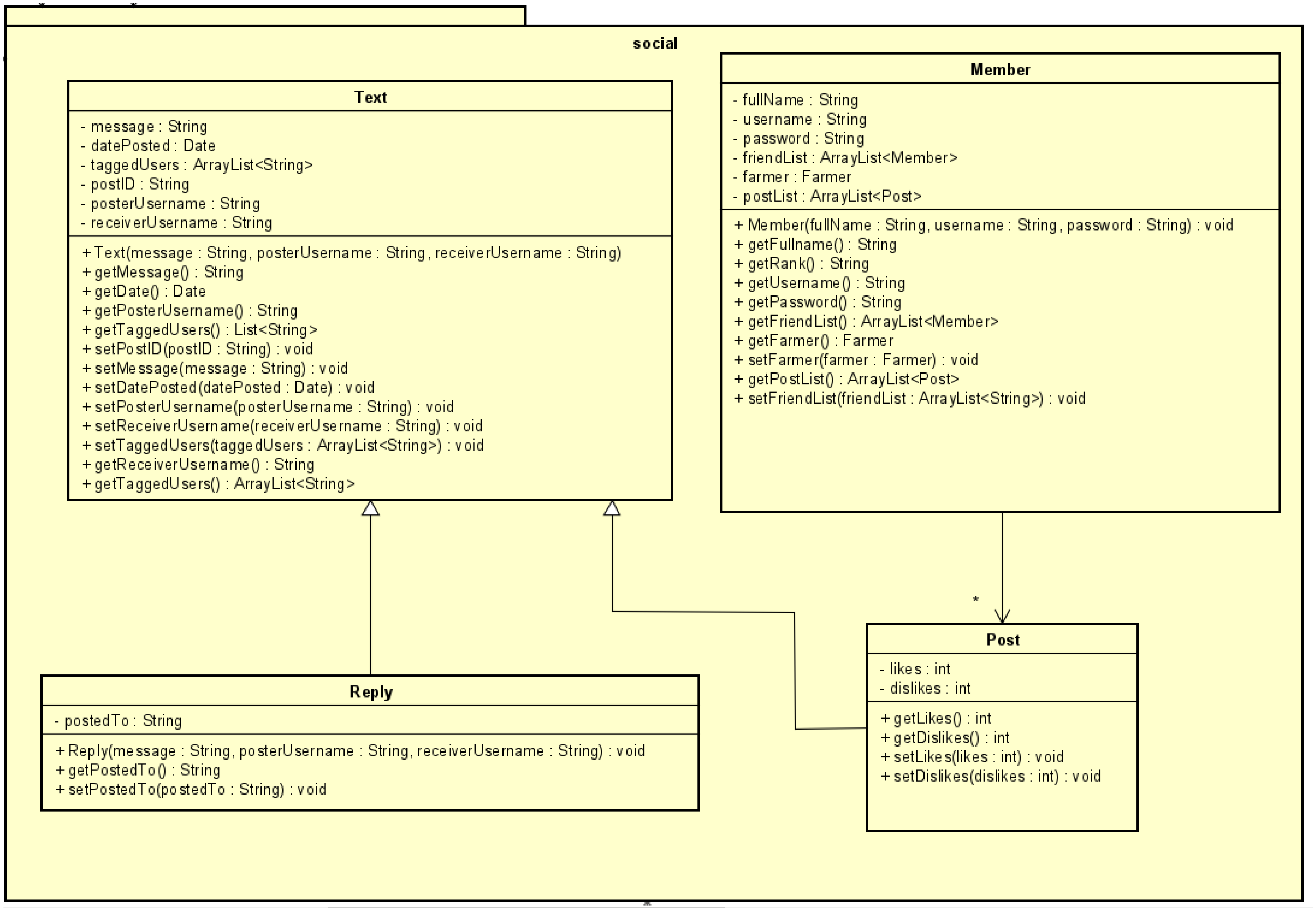
Date of Submission:

08 April 2020

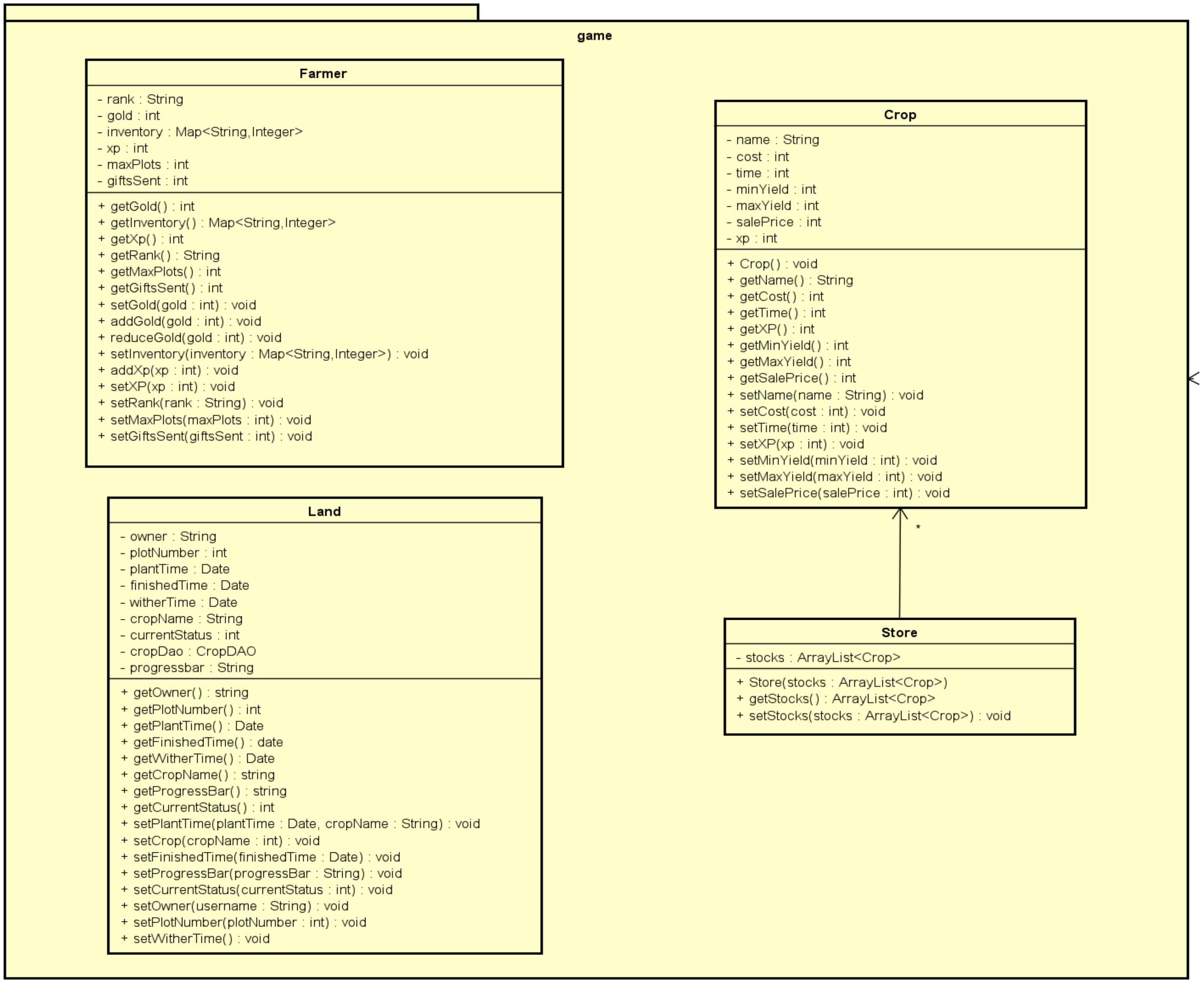
Contents

1. Class Diagram

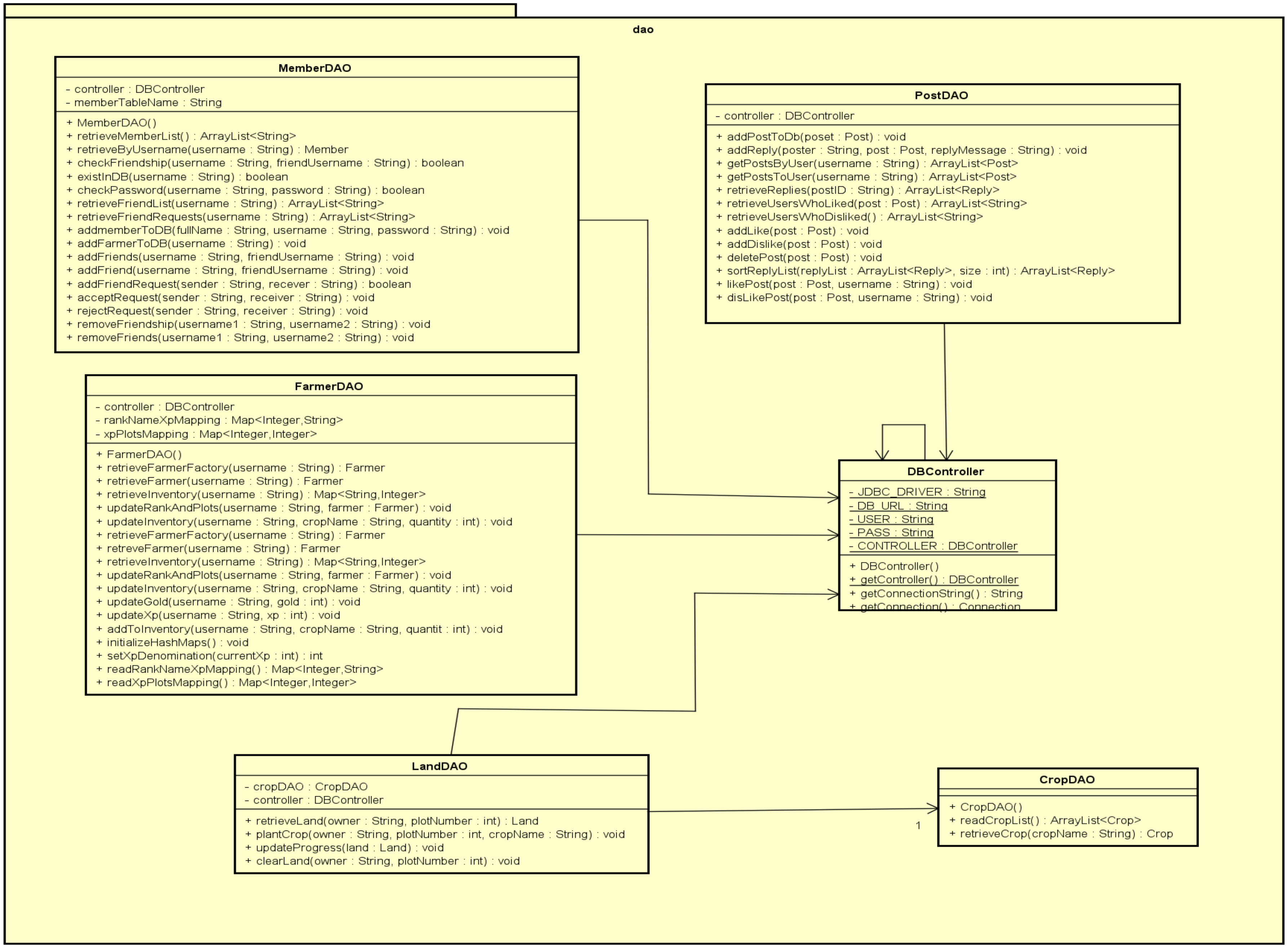
- Social Package


social
Text
- message : String
- datePosted : Date
- taggedUsers : ArrayList<String>
- postID : String
- posterUsername : String
- receiverUsername : String
+ Text(message : String, posterUsername : String, receiverUsername : String)
+ getMessage() : String
+ getDate() : Date
+ getPosterUsername() : String
+ getTaggedUsers() : List<String>
+ setPostID(postID : String) : void
+ setMessage(message : String) : void
+ setDatePosted(datePosted : Date) : void
+ setPosterUsername(posterUsername : String) : void
+ setReceiverUsername(receiverUsername : String) : void
+ setTaggedUsers(taggedUsers : ArrayList<String>) : void
+ getReceiverUsername() : String
+ getTaggedUsers() : ArrayList<String>
Member
- fullName : String
- username : String
- password : String
- friendList : ArrayList<Member>
- farmer : Farmer
- postList : ArrayList<Post>
+ Member(fullName : String, username : String, password : String) : void
+ getFullname() : String
+ getRank() : String
+ getUsername() : String
+ getPassword() : String
+ getFriendList() : ArrayList<Member>
+ getFarmer() : Farmer
+ setFarmer(farmer : Farmer) : void
+ getPostList() : ArrayList<Post>
+ setFriendList(friendList : ArrayList<String>) : void
*
Post
- likes : int
- dislikes : int
+ getLikes() : int
+ getDislikes() : int
+ setLikes(likes : int) : void
+ setDislikes(dislikes : int) : void
Reply
- postedTo : String
+ Reply(message : String, posterUsername : String, receiverUsername : String) : void
+ getPostedTo() : String
+ setPostedTo(postedTo : String) : void

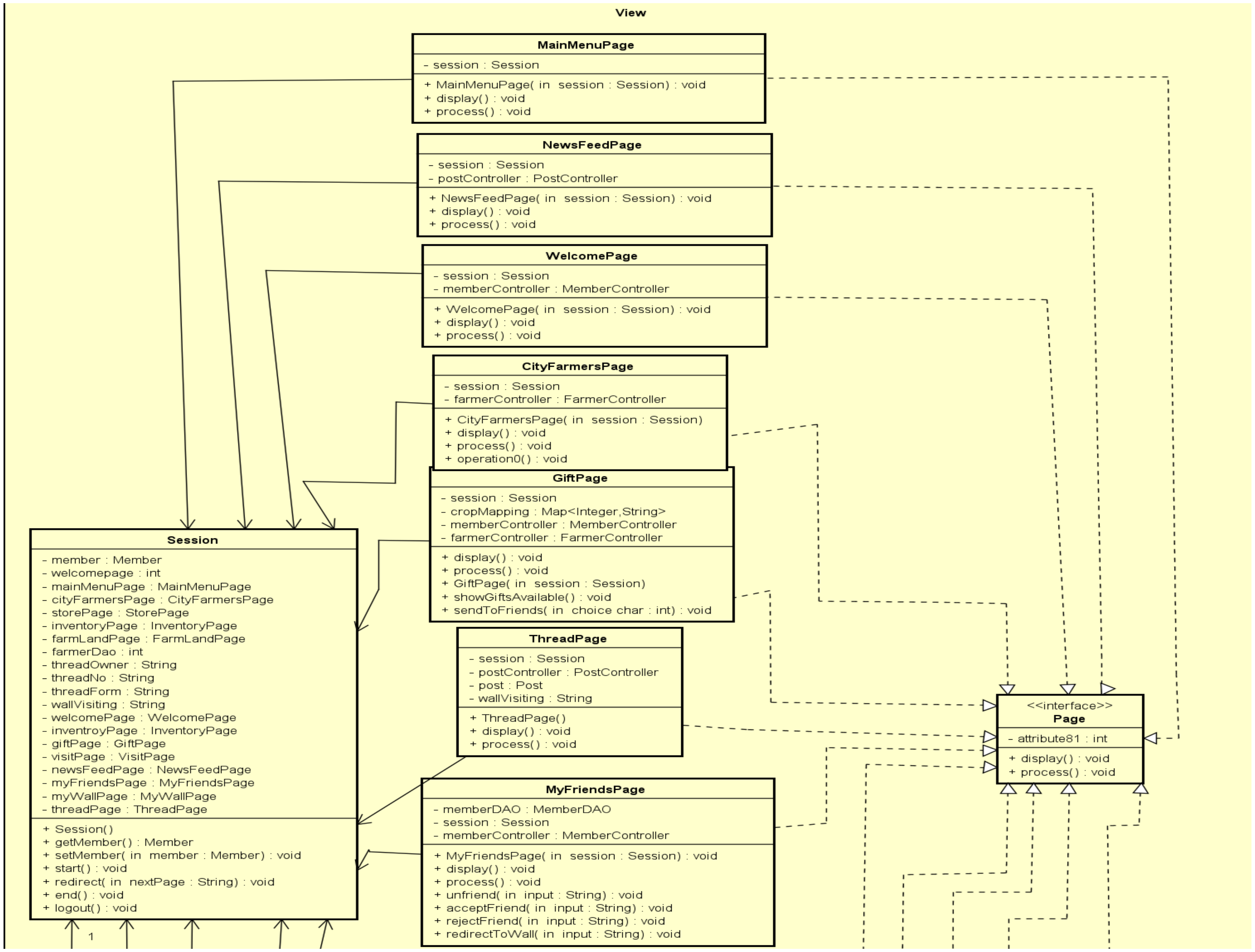
- Game Package

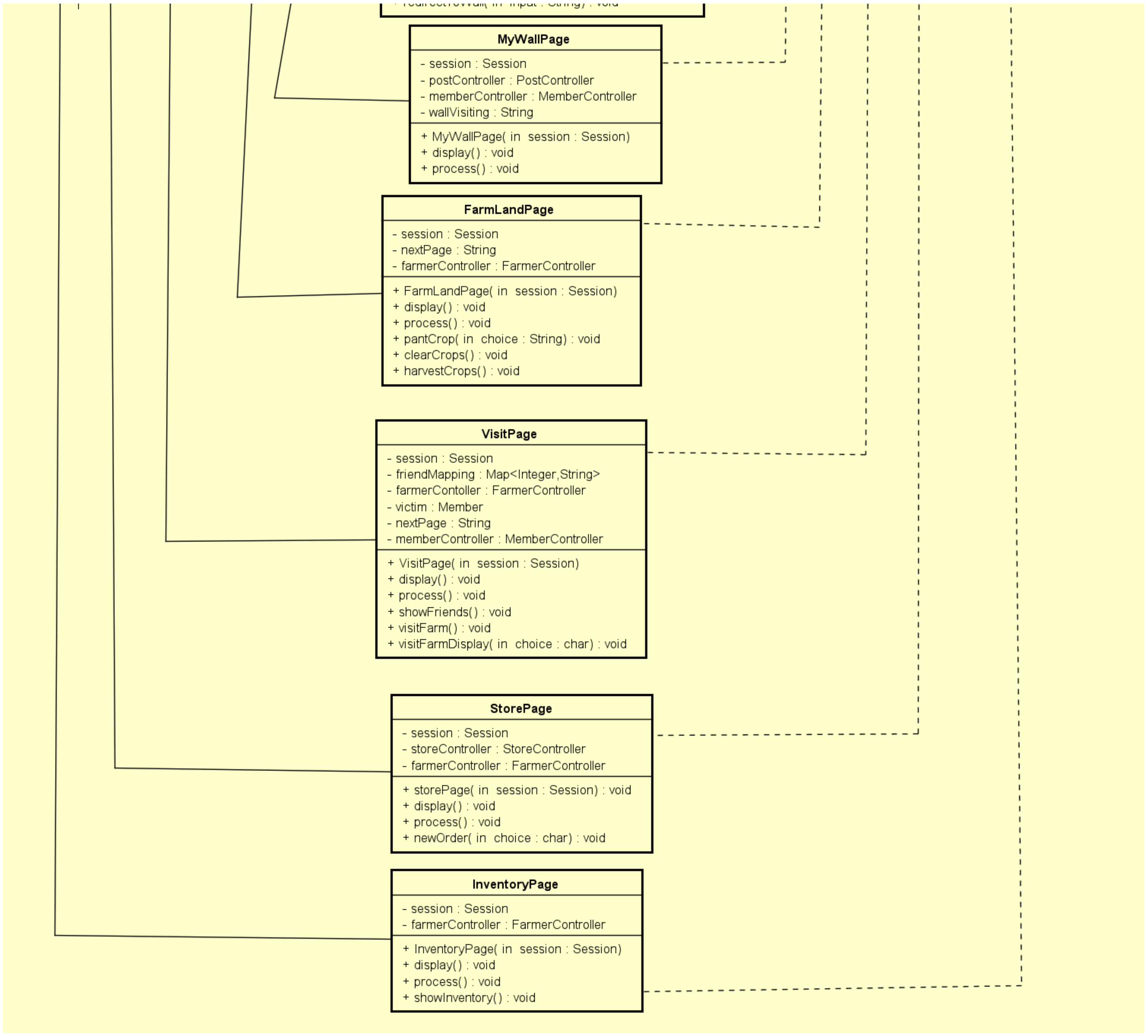

game
Farmer
- rank : String
- gold : int
- inventory : Map<String,Integer>
- xp : int
- maxPlots : int
- giftsSent : int
+ getGold() : int
+ getInventory() : Map<String,Integer>
+ getXp() : int
+ getRank() : String
+ getMaxPlots() : int
+ getGiftsSent() : int
+ setGold(gold : int) : void
+ addGold(gold : int) : void
+ reduceGold(gold : int) : void
+ setInventory(inventory : Map<String,Integer>) : void
+ addXp(xp : int) : void
+ setXP(xp : int) : void
+ setRank(rank : String) : void
+ setMaxPlots(maxPlots : int) : void
+ setGiftsSent(giftsSent : int) : void
Crop
- name : String
- cost : int
- time : int
- minYield : int
- maxYield : int
- salePrice : int
- xp : int
+ Crop() : void
+ getName() : String
+ getCost() : int
+ getTime() : int
+ getXP() : int
+ getMinYield() : int
+ getMaxYield() : int
+ getSalePrice() : int
+ setName(name : String) : void
+ setCost(cost : int) : void
+ setTime(time : int) : void
+ setXP(xp : int) : void
+ setMinYield(minYield : int) : void
+ setMaxYield(maxYield : int) : void
+ setSalePrice(salePrice : int) : void
*
Land
- owner : String
- plotNumber : int
- plantTime : Date
- finishedTime : Date
- witherTime : Date
- cropName : String
- currentStatus : int
- cropDao : CropDAO
- progressbar : String
+ getOwner() : string
+ getPlotNumber() : int
+ getPlantTime() : Date
+ getFinishedTime() : date
+ getWitherTime() : Date
+ getCropName() : string
+ getProgressBar() : string
+ getCurrentStatus() : int
+ setPlantTime(plantTime : Date, cropName : String) : void
+ setCrop(cropName : int) : void
+ setFinishedTime(finishedTime : Date) : void
+ setProgressBar(progressBar : String) : void
+ setCurrentStatus(currentStatus : int) : void
+ setOwner(username : String) : void
+ setPlotNumber(plotNumber : int) : void
+ setWitherTime() : void
Store
- stocks : ArrayList<Crop>
+ Store(stocks : ArrayList<Crop>)
+ getStocks() : ArrayList<Crop>
+ setStocks(stocks : ArrayList<Crop>) : void

- DAO package

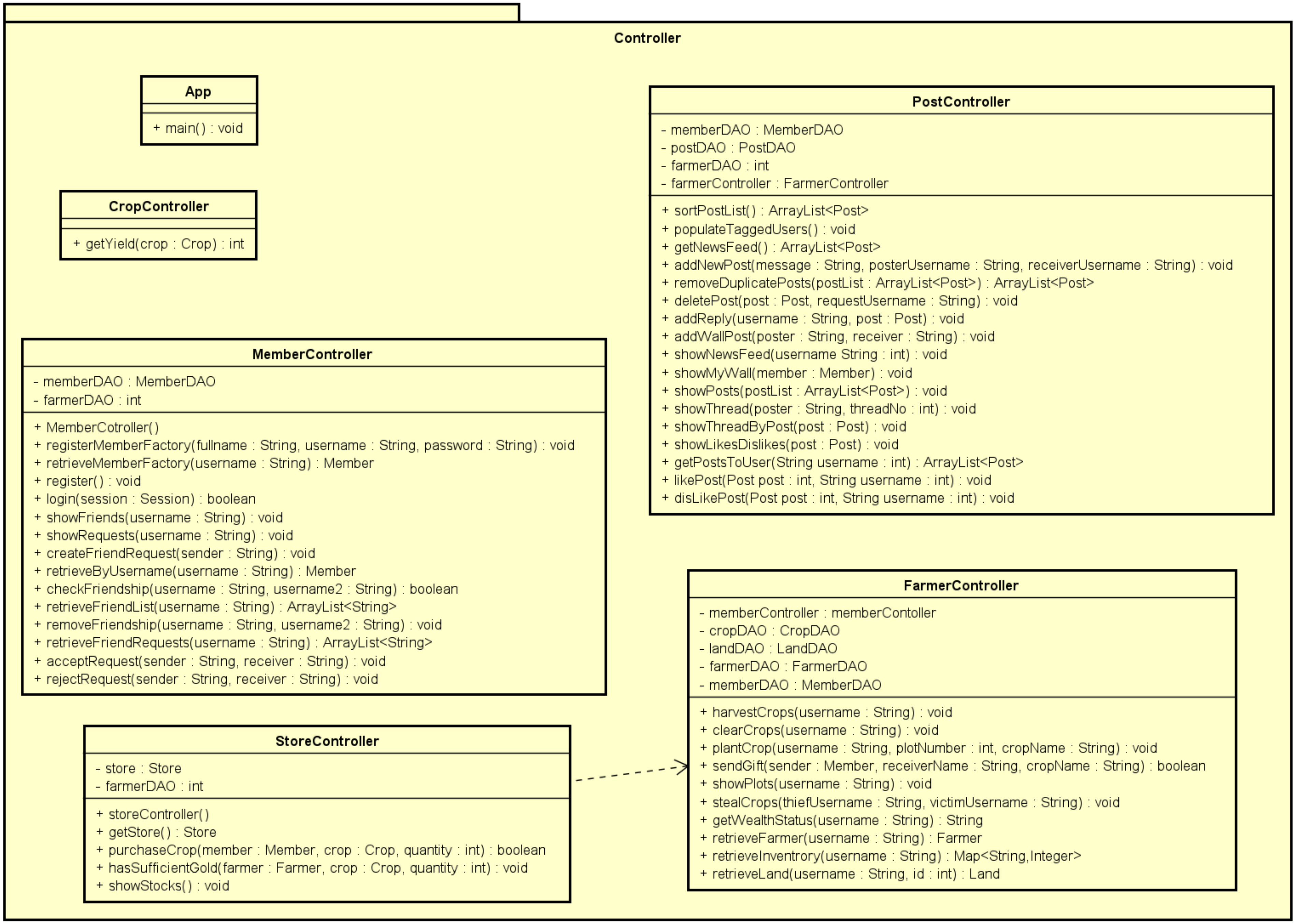

dao
MemberDAO
- controller : DBController
- memberTableName : String
+ MemberDAO()
+ retrieveMemberList() : ArrayList<String>
+ retrieveByUsername(username : String) : Member
+ checkFriendship(username : String, friendUsername : String) : boolean
+ existInDB(username : String) : boolean
+ checkPassword(username : String, password : String) : boolean
+ retrieveFriendList(username : String) : ArrayList<String>
+ retrieveFriendRequests(username : String) : ArrayList<String>
+ addmemberToDB(fullName : String, username : String, password : String) : void
+ addFarmerToDB(username : String) : void
+ addFriends(username : String, friendUsername : String) : void
+ addFriend(username : String, friendUsername : String) : void
+ addFriendRequest(sender : String, recever : String) : boolean
+ acceptRequest(sender : String, receiver : String) : void
+ rejectRequest(sender : String, receiver : String) : void
+ removeFriendship(username1 : String, username2 : String) : void
+ removeFriends(username1 : String, username2 : String) : void
PostDAO
- controller : DBController
+ addPostToDb(poset : Post) : void
+ addReply(poster : String, post : Post, replyMessage : String) : void
+ getPostsByUser(username : String) : ArrayList<Post>
+ getPostsToUser(username : String) : ArrayList<Post>
+ retrieveReplies(postID : String) : ArrayList<Reply>
+ retrieveUsersWhoLiked(post : Post) : ArrayList<String>
+ retrieveUsersWhoDisliked() : ArrayList<String>
+ addLike(post : Post) : void
+ addDislike(post : Post) : void
+ deletePost(post : Post) : void
+ sortReplyList(replyList : ArrayList<Reply>, size : int) : ArrayList<Reply>
+ likePost(post : Post, username : String) : void
+ disLikePost(post : Post, username : String) : void
FarmerDAO
- controller : DBController
- rankNameXpMapping : Map<Integer,String>
- xpPlotsMapping : Map<Integer,Integer>
+ FarmerDAO()
+ retrieveFarmerFactory(username : String) : Farmer
+ retrieveFarmer(username : String) : Farmer
+ retrieveInventory(username : String) : Map<String,Integer>
+ updateRankAndPlots(username : String, farmer : Farmer) : void
+ updateInventory(username : String, cropName : String, quantity : int) : void
+ retrieveFarmerFactory(username : String) : Farmer
+ retreveFarmer(username : String) : Farmer
+ retrieveInventory(username : String) : Map<String,Integer>
+ updateRankAndPlots(username : String, farmer : Farmer) : void
+ updateInventory(username : String, cropName : String, quantity : int) : void
+ updateGold(username : String, gold : int) : void
+ updateXp(username : String, xp : int) : void
+ addToInventory(username : String, cropName : String, quantit : int) : void
+ initializeHashMaps() : void
+ setXpDenomination(currentXp : int) : int
+ readRankNameXpMapping() : Map<Integer,String>
+ readXpPlotsMapping() : Map<Integer,Integer>
DBController
- JDBC_DRIVER : String
- DB_URL : String
- USER : String
- PASS : String
- CONTROLLER : DBController
+ DBController()
+ getController() : DBController
+ getConnectionString() : String
+ getConnection() : Connection
LandDAO
- cropDAO : CropDAO
- controller : DBController
+ retrieveLand(owner : String, plotNumber : int) : Land
+ plantCrop(owner : String, plotNumber : int, cropName : String) : void
+ updateProgress(land : Land) : void
+ clearLand(owner : String, plotNumber : int) : void
CropDAO
+ CropDAO()
+ readCropList() : ArrayList<Crop>
+ retrieveCrop(cropName : String) : Crop
1

- View Package

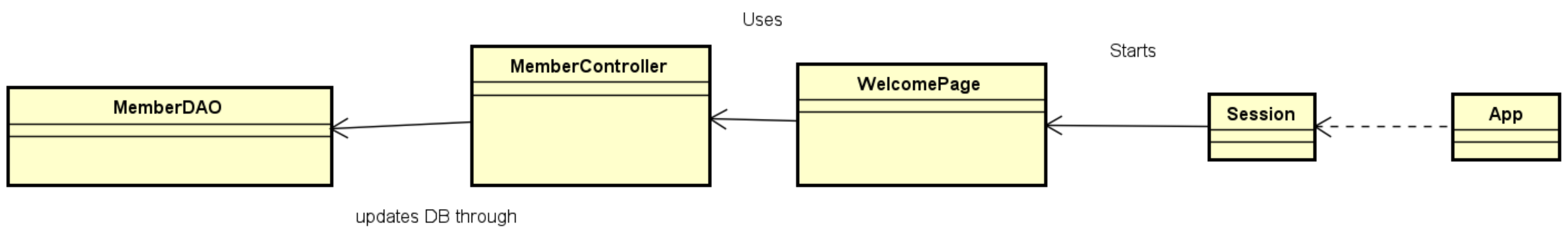

View
MainMenuPage
- session : Session
+ MainMenuPage(in session : Session) : void
+ display() : void
+ process() : void
NewsFeedPage
- session : Session
- postController : PostController
+ NewsFeedPage(in session : Session) : void
+ display() : void
+ process() : void
WelcomePage
- session : Session
- memberController : MemberController
+ WelcomePage(in session : Session) : void
+ display() : void
+ process() : void
CityFarmersPage
- session : Session
- farmerController : FarmerController
+ CityFarmersPage(in session : Session)
+ display() : void
+ process() : void
+ operation0() : void
GiftPage
- session : Session
- cropMapping : Map<Integer,String>
- memberController : MemberController
- farmerController : FarmerController
+ display() : void
+ process() : void
+ GiftPage(in session : Session)
+ showGiftsAvailable() : void
+ sendToFriends(in choice char : int) : void
ThreadPage
- session : Session
- postController : PostController
- post : Post
- wallVisiting : String
+ ThreadPage()
+ display() : void
+ process() : void
MyFriendsPage
- memberDAO : MemberDAO
- session : Session
- memberController : MemberController
+ MyFriendsPage(in session : Session) : void
+ display() : void
+ process() : void
+ unfriend(in input : String) : void
+ acceptFriend(in input : String) : void
+ rejectFriend(in input : String) : void
+ redirectToWall(in input : String) : void
Session
- member : Member
- welcomepage : int
- mainMenuPage : MainMenuPage
- cityFarmersPage : CityFarmersPage
- storePage : StorePage
- inventoryPage : InventoryPage
- farmLandPage : FarmLandPage
- farmerDao : int
- threadOwner : String
- threadNo : String
- threadForm : String
- wallVisiting : String
- welcomePage : WelcomePage
- inventroyPage : InventoryPage
- giftPage : GiftPage
- visitPage : VisitPage
- newsFeedPage : NewsFeedPage
- myFriendsPage : MyFriendsPage
- myWallPage : MyWallPage
- threadPage : ThreadPage
+ Session()
+ getMember() : Member
+ setMember(in member : Member) : void
+ start() : void
+ redirect(in nextPage : String) : void
+ end() : void
+ logout() : void
1
<<interface>>
Page
- attribute81 : int
+ display() : void
+ process() : void


MyWallPage
- session : Session
- postController : PostController
- memberController : MemberController
- wallVisiting : String
+ MyWallPage(in session : Session)
+ display() : void
+ process() : void
FarmLandPage
- session : Session
- nextPage : String
- farmerController : FarmerController
+ FarmLandPage(in session : Session)
+ display() : void
+ process() : void
+ pantCrop(in choice : String) : void
+ clearCrops() : void
+ harvestCrops() : void
VisitPage
- session : Session
- friendMapping : Map<Integer,String>
- farmerContoller : FarmerController
- victim : Member
- nextPage : String
- memberController : MemberController
+ VisitPage(in session : Session)
+ display() : void
+ process() : void
+ showFriends() : void
+ visitFarm() : void
+ visitFarmDisplay(in choice : char) : void
StorePage
- session : Session
- storeController : StoreController
- farmerController : FarmerController
+ storePage(in session : Session) : void
+ display() : void
+ process() : void
+ newOrder(in choice : char) : void
InventoryPage
- session : Session
- farmerController : FarmerController
+ InventoryPage(in session : Session)
+ display() : void
+ process() : void
+ showInventory() : void

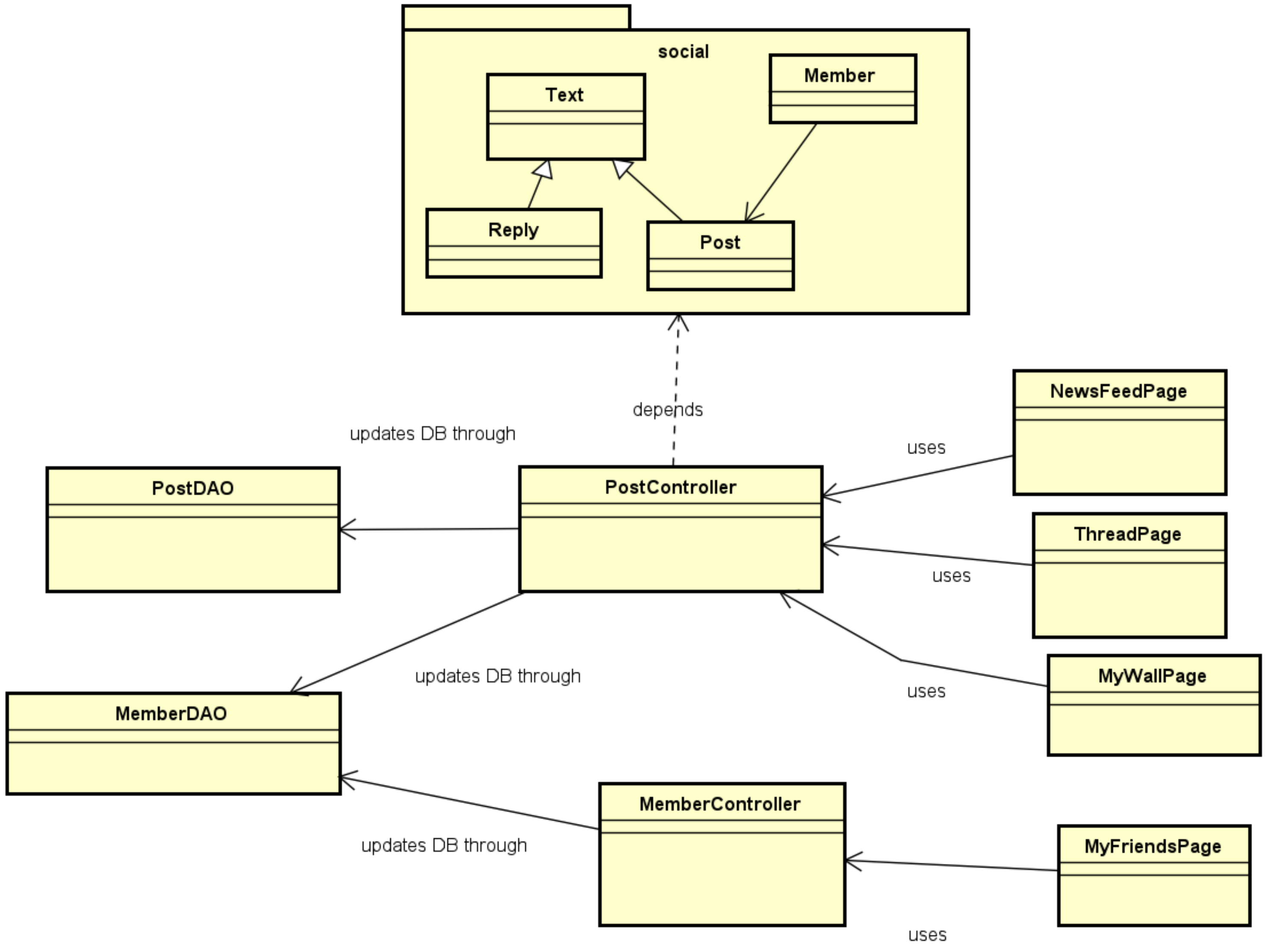
- Controller Package


Controller
App
+ main() : void
CropController
+ getYield(crop : Crop) : int
MemberController
- memberDAO : MemberDAO
- farmerDAO : int
+ MemberCotroller()
+ registerMemberFactory(fullname : String, username : String, password : String) : void
+ retrieveMemberFactory(username : String) : Member
+ register() : void
+ login(session : Session) : boolean
+ showFriends(username : String) : void
+ showRequests(username : String) : void
+ createFriendRequest(sender : String) : void
+ retrieveByUsername(username : String) : Member
+ checkFriendship(username : String, username2 : String) : boolean
+ retrieveFriendList(username : String) : ArrayList<String>
+ removeFriendship(username : String, username2 : String) : void
+ retrieveFriendRequests(username : String) : ArrayList<String>
+ acceptRequest(sender : String, receiver : String) : void
+ rejectRequest(sender : String, receiver : String) : void
StoreController
- store : Store
- farmerDAO : int
+ storeController()
+ getStore() : Store
+ purchaseCrop(member : Member, crop : Crop, quantity : int) : boolean
+ hasSufficientGold(farmer : Farmer, crop : Crop, quantity : int) : void
+ showStocks() : void
PostController
- memberDAO : MemberDAO
- postDAO : PostDAO
- farmerDAO : int
- farmerController : FarmerController
+ sortPostList() : ArrayList<Post>
+ populateTaggedUsers() : void
+ getNewsFeed() : ArrayList<Post>
+ addNewPost(message : String, posterUsername : String, receiverUsername : String) : void
+ removeDuplicatePosts(postList : ArrayList<Post>) : ArrayList<Post>
+ deletePost(post : Post, requestUsername : String) : void
+ addReply(username : String, post : Post) : void
+ addWallPost(poster : String, receiver : String) : void
+ showNewsFeed(username String : int) : void
+ showMyWall(member : Member) : void
+ showPosts(postList : ArrayList<Post>) : void
+ showThread(poster : String, threadNo : int) : void
+ showThreadByPost(post : Post) : void
+ showLikesDislikes(post : Post) : void
+ getPostsToUser(String username : int) : ArrayList<Post>
+ likePost(Post post : int, String username : int) : void
+ disLikePost(Post post : int, String username : int) : void
FarmerController
- memberController : memberContoller
- cropDAO : CropDAO
- landDAO : LandDAO
- farmerDAO : FarmerDAO
- memberDAO : MemberDAO
+ harvestCrops(username : String) : void
+ clearCrops(username : String) : void
+ plantCrop(username : String, plotNumber : int, cropName : String) : void
+ sendGift(sender : Member, receiverName : String, cropName : String) : boolean
+ showPlots(username : String) : void
+ stealCrops(thiefUsername : String, victimUsername : String) : void
+ getWealthStatus(username : String) : String
+ retrieveFarmer(username : String) : Farmer
+ retrieveInventrory(username : String) : Map<String,Integer>
+ retrieveLand(username : String, id : int) : Land

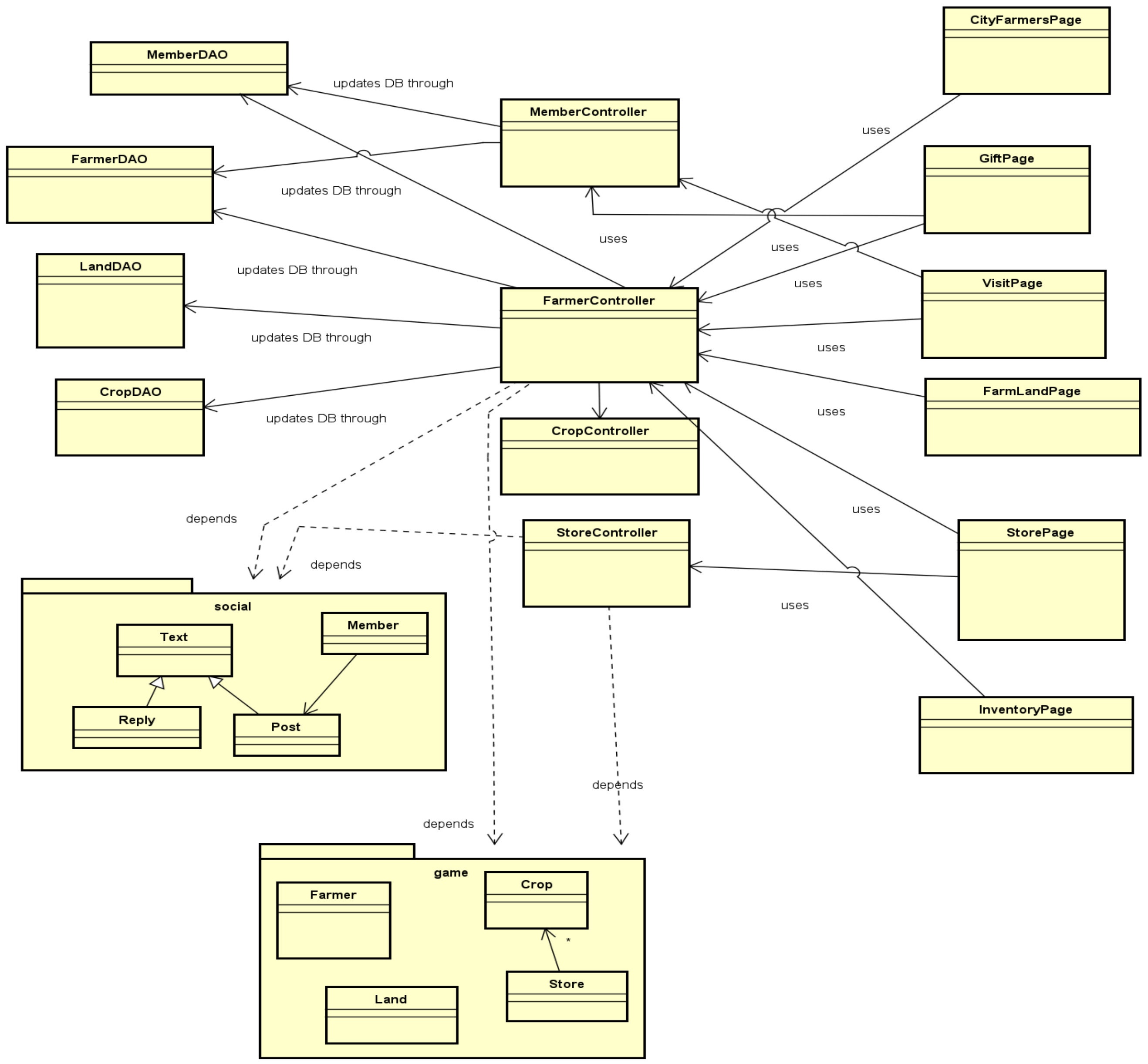
- Class Diagram (Start)



- Class Diagrams (Social)



- Class Diagrams (Game)



2. Object Oriented Design Considerations

a. Approach Considerations

a. MVC (Model-View-Controller) Design Pattern

i. For Design choice, we tried implementing the MVC pattern. This pattern helps us to separate the font-end from back-end logic, thus, also separating application's concern.
Model - Model represents an object carrying data. It can also have logic to update controller if its data changes.
View - View represents the visualization of the data that model contains.
Controller - Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.

ii. **Reasons:**
1. Helps us to separate view logic from business logic
2. Allow simultaneous work between developers who are responsible for different components (such as UI layer and core logic)
3. Easier to maintain, easy debugging, separation of concerns

b. DAO (Data Access Object) Pattern

i. The DAO design pattern is used to separate the data persistence logic in a separate layer. This way, the main application will not need to be concern about how the low -level operations to access the database is done. Thereby allowing for further separation of Concerns.

ii. Reasons:
1. The service layer does not need to know where the data comes from. Allows for the shifting from MySQL to other DBs e.g. MongoDB.
2. Ensures loose coupling between different components of an application. View layer has no dependency on the DAO layer

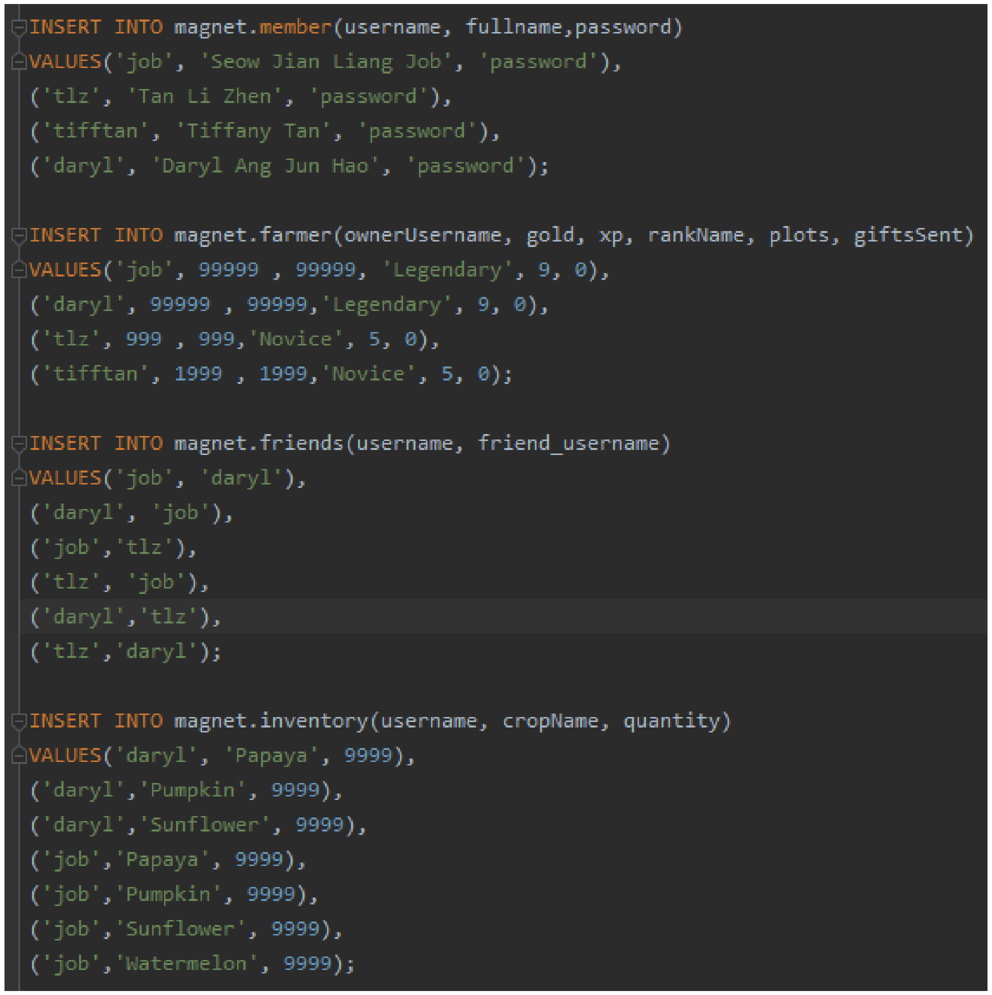
c. Singleton Pattern

i. Singleton pattern in DBController when calling getConnection method

ii. Reason:
1. Only needs to create the connection once and not many times. Helping in performance and maintainability.

3. Test Cases

For testing purposes, a few test cases were written into the deploy.sql file.



```sql
INSERT INTO magnet.member(username, fullname,password)
VALUES('job', 'Seow Jian Liang Job', 'password'),
('tlz', 'Tan Li Zhen', 'password'),
('tifftan', 'Tiffany Tan', 'password'),
('daryl', 'Daryl Ang Jun Hao', 'password');

INSERT INTO magnet.farmer(ownerUsername, gold, xp, rankName, plots, giftsSent)
VALUES('job', 99999 , 99999, 'Legendary', 9, 0),
('daryl', 99999 , 99999,'Legendary', 9, 0),
('tlz', 999 , 999,'Novice', 5, 0),
('tifftan', 1999 , 1999,'Novice', 5, 0);

INSERT INTO magnet.friends(username, friend_username)
VALUES('job', 'daryl'),
('daryl', 'job'),
('job','tlz'),
('tlz', 'job'),
('daryl','tlz'),
('tlz','daryl');

INSERT INTO magnet.inventory(username, cropName, quantity)
VALUES('daryl', 'Papaya', 9999),
('daryl','Pumpkin', 9999),
('daryl','Sunflower', 9999),
('job','Papaya', 9999),
('job','Pumpkin', 9999),
('job','Sunflower', 9999),
('job','Watermelon', 9999);
```

5.1 Member Tests

Test Description	Tested Functions	Input	Expected Result	Actual Result
Test if we can check that “daryl” exists	memberDAO.existInDB()	“daryl”	True	True
Test if “daryl” and “job” are friends	memberDAO.checkFriendship()	“daryl”,”job”	True	True
Login test	memberDAO.checkPassword()	“daryl”,”password”	True	True

5.2 Post Tests

Test Description	Tested Functions	Input	Expected Result	Actual Result
Test if tags in message are handled properly (ie @daryl becomes daryl)	postController.populateTaggedUsers(), post.getMessage()	New Post("@daryl < will become daryl and @random < will stay as @random", “”, “”)	"daryl < will become daryl and @random < will stay as @random"	True
Extract the existing tagged users	postController.populateTaggedUsers(), post.getTaggedUsers().size()	New Post(“@daryl @job @random", “”, “”)	2	True

5.3 Farmer Tests

Test Description	Tested Functions	Input	Expected Result	Actual Result
Check the returning of wealth status	farmerController.getWealthStatus()	“daryl”	“Richest”	True
Test xp and rank name mapping that is taken from csv file	farmerDAO.readRankNameXpMapping(), rankNameXpMapping.get()	1000	“Apprentice”	True

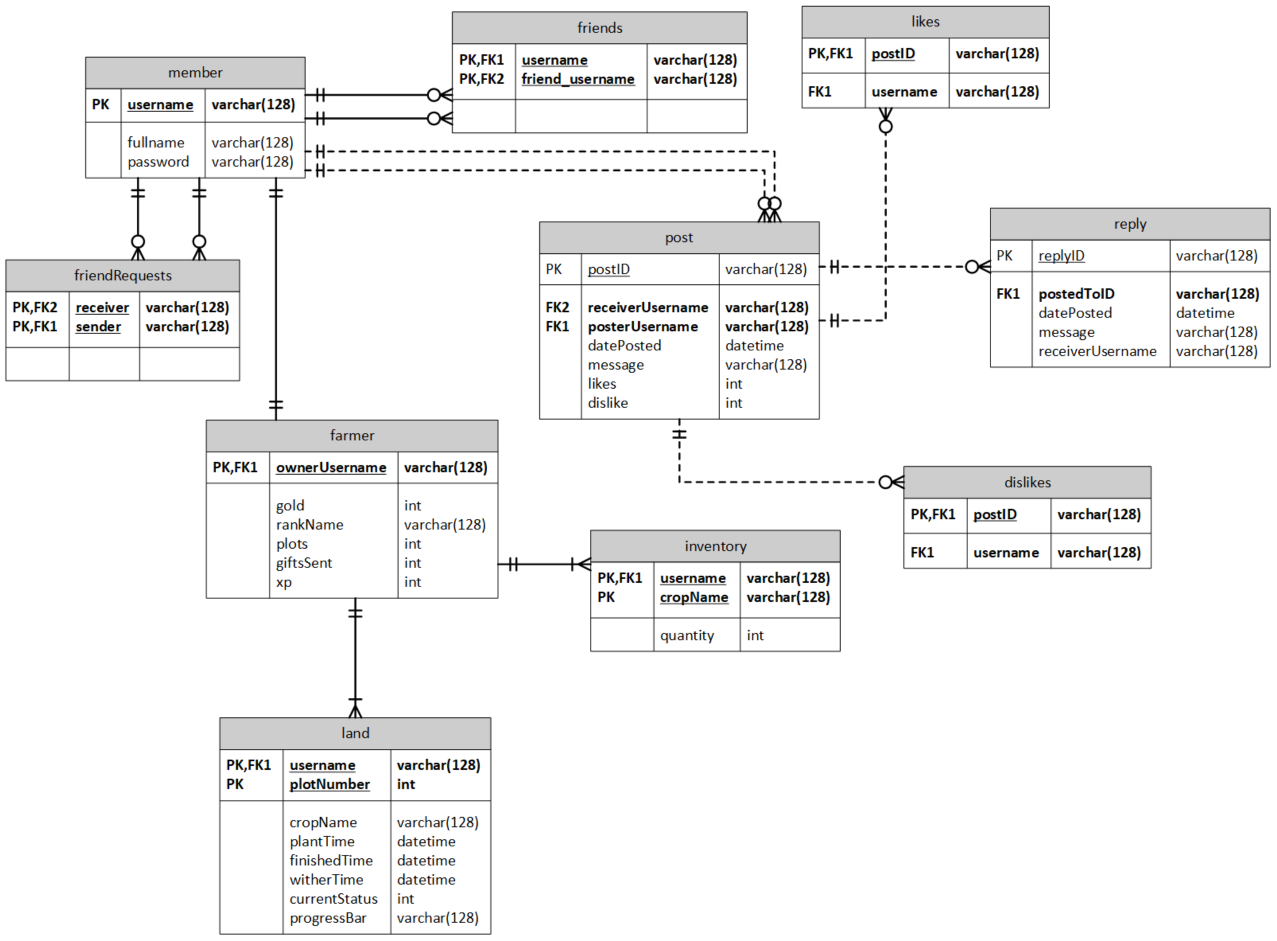
5.4 Crop Tests

Test Description	Tested Functions	Input	Expected Result	Actual Result

Populate the crop list by reading from csv file	cropDAO.readCropList(), cropList.size()	-	4	True

4. Additional

- ER Diagram



member
PK username varchar(128)
fullname varchar(128)
password varchar(128)
friends
PK,FK1 username varchar(128)
PK,FK2 friend_username varchar(128)
likes
PK,FK1 postID varchar(128)
FK1 username varchar(128)
friendRequests
PK,FK2 receiver varchar(128)
PK,FK1 sender varchar(128)
post
PK postID varchar(128)
FK2 receiverUsername varchar(128)
FK1 posterUsername varchar(128)
datePosted datetime
message varchar(128)
likes int
dislike int
reply
PK replyID varchar(128)
FK1 postedToID varchar(128)
datePosted datetime
message varchar(128)
receiverUsername varchar(128)
farmer
PK,FK1 ownerUsername varchar(128)
gold int
rankName varchar(128)
plots int
giftsSent int
xp int
dislikes
PK,FK1 postID varchar(128)
FK1 username varchar(128)
inventory
PK,FK1 username varchar(128)
PK cropName varchar(128)
quantity int
land
PK,FK1 username varchar(128)
PK plotNumber int
cropName varchar(128)
plantTime datetime
finishedTime datetime
witherTime datetime
currentStatus int
progressBar varchar(128)