M3

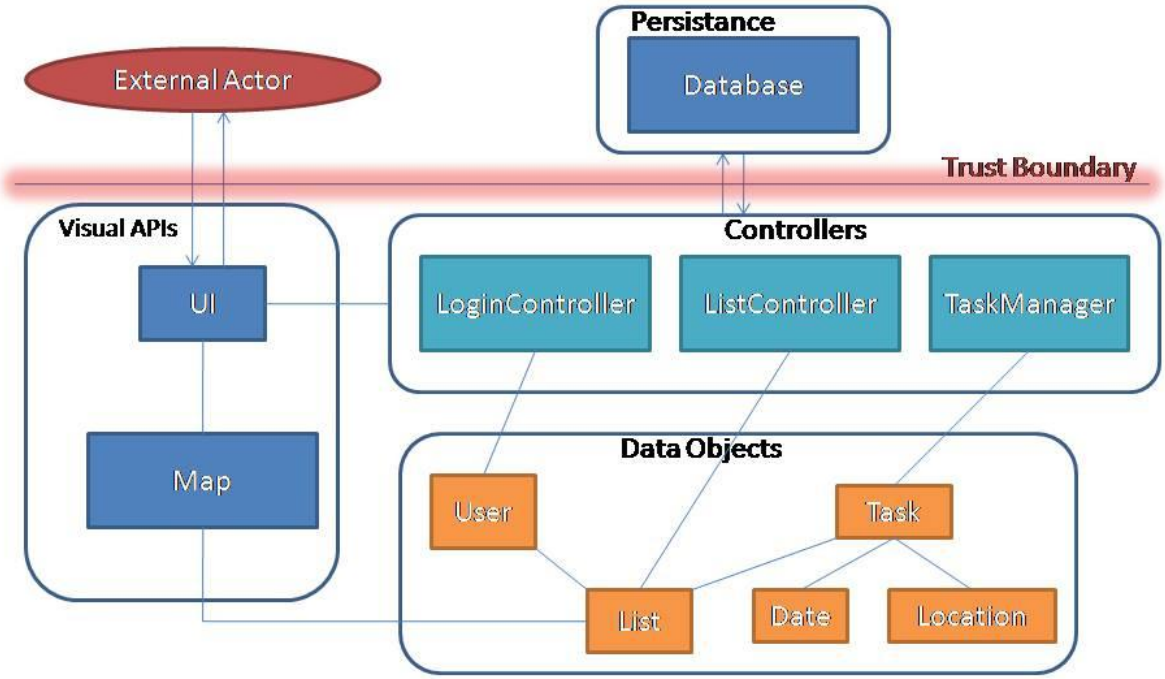
CS 2340

Prof Bob Waters

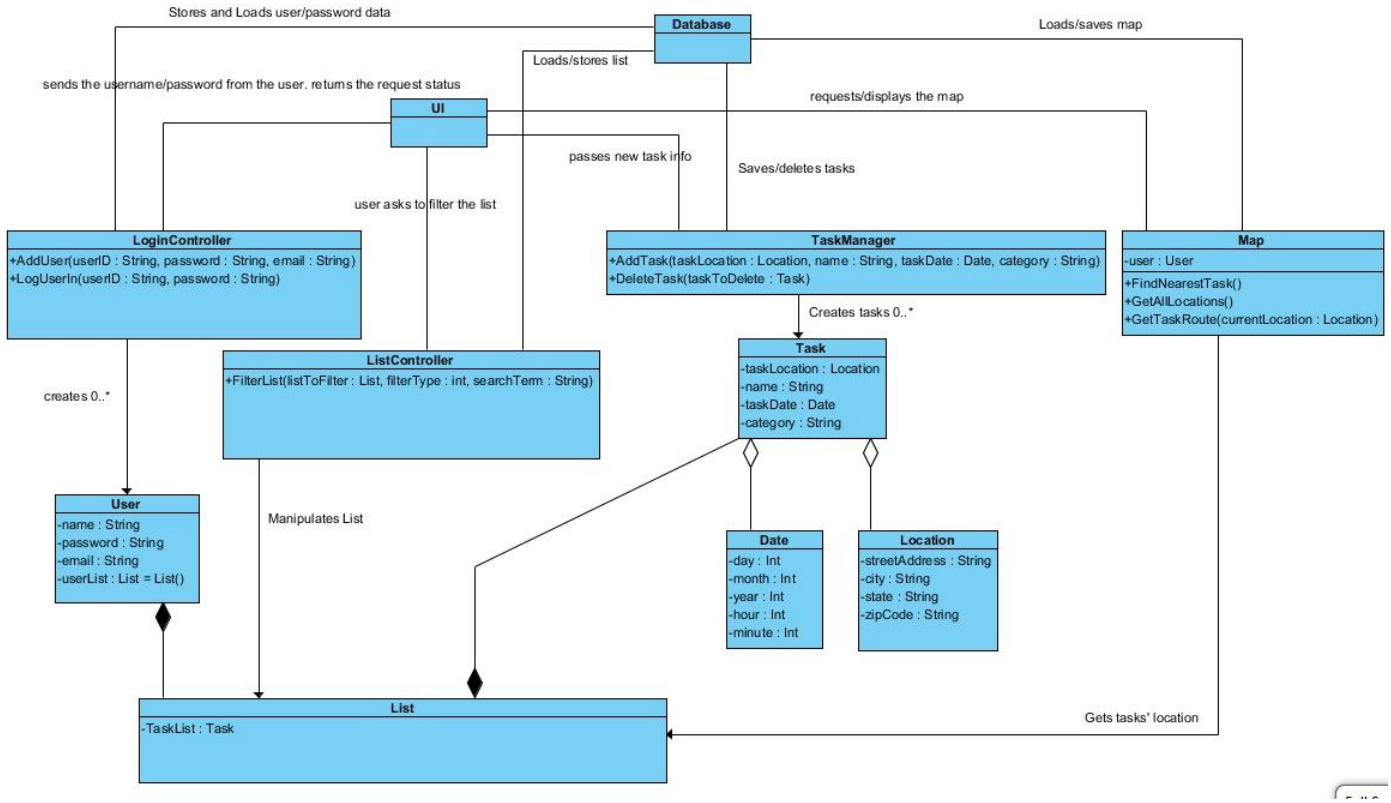
Team High Ceiling: Caleb Kirksey, Jeff Pansini, Dan Doozan, Sam Bell

February 8, 2012

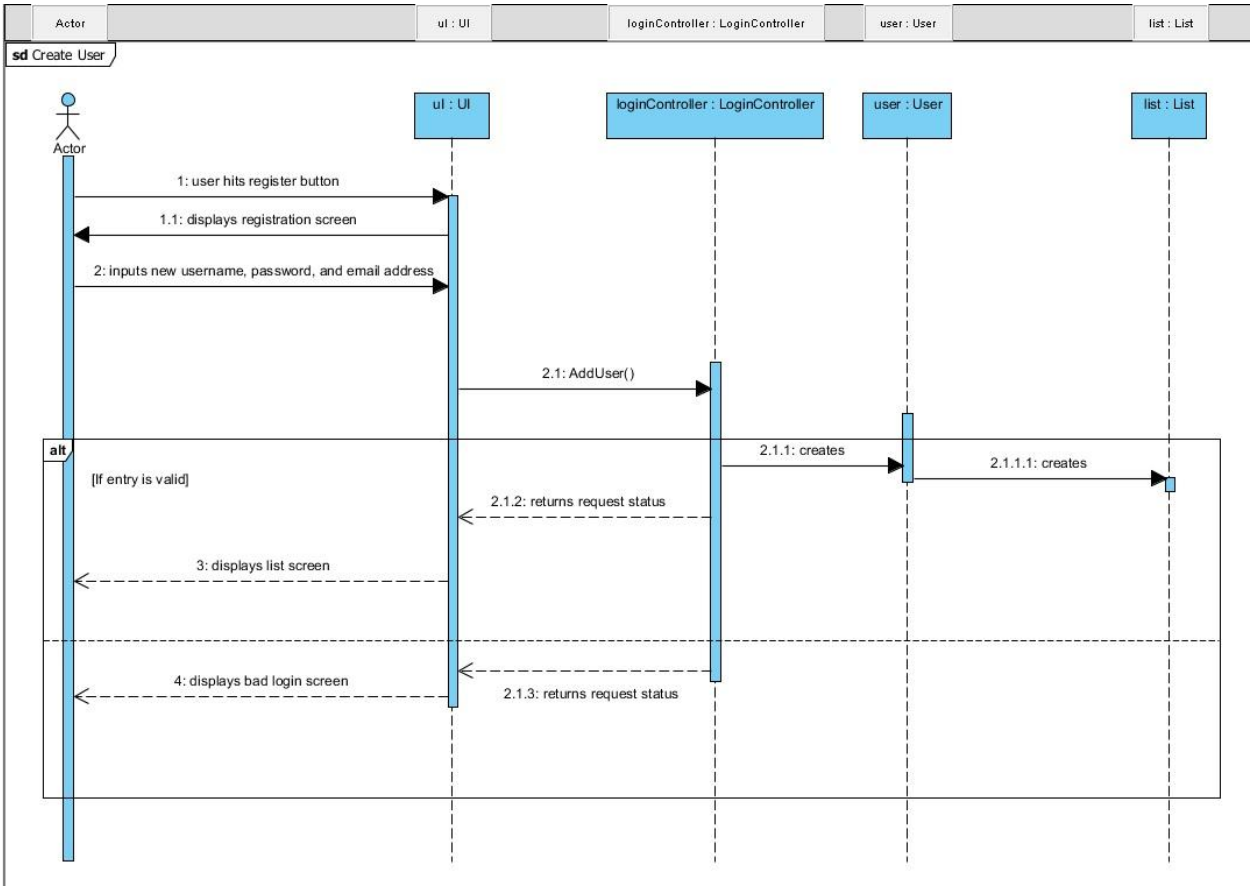
Architecture/Trust Boundaries

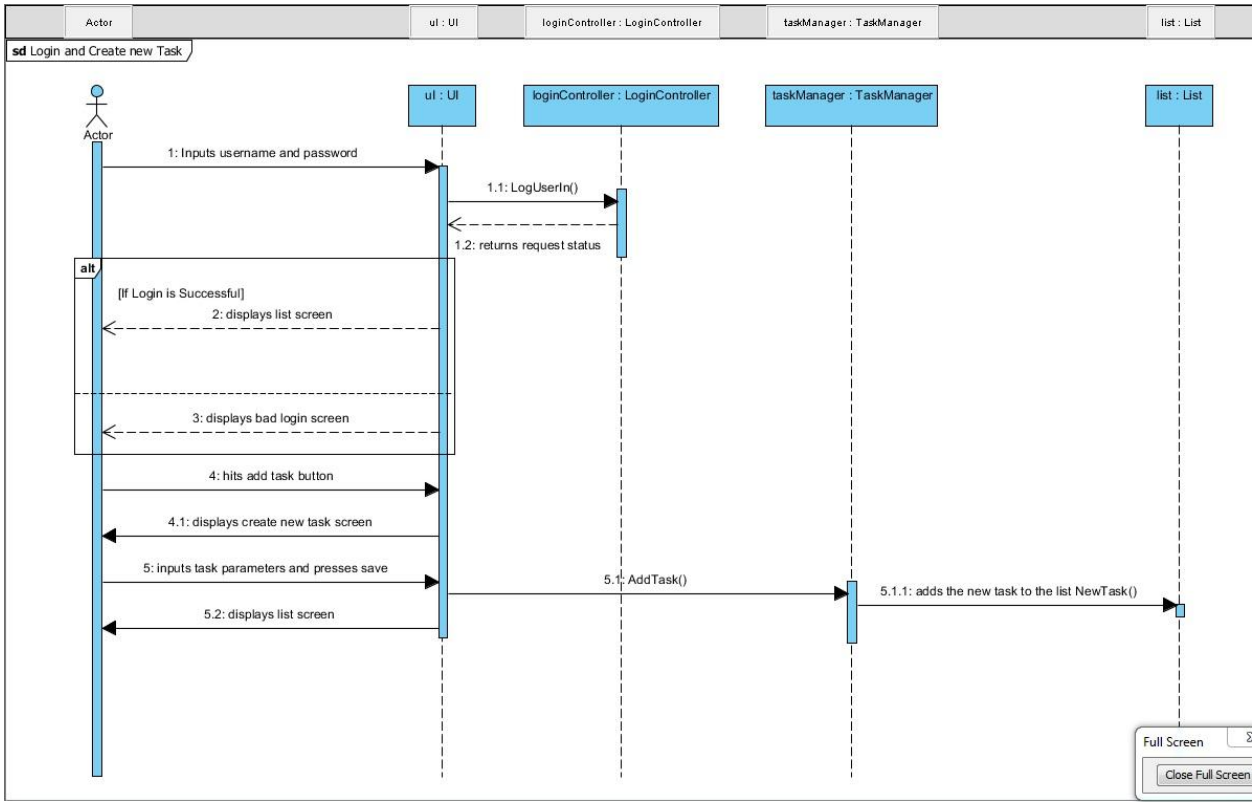


UML Class Diagram

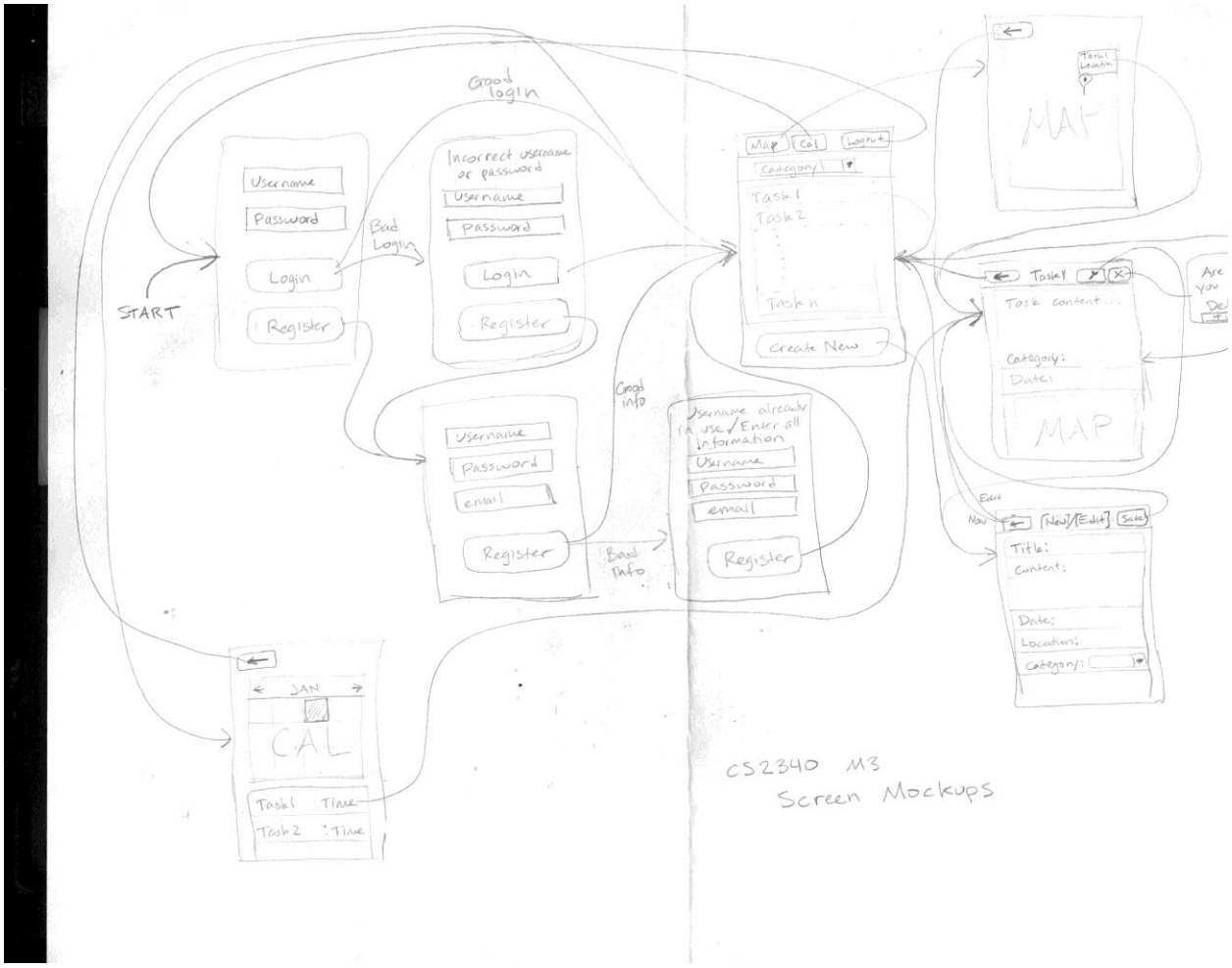


UML Sequence Diagrams




Actor
ui : UI
loginController : LoginController
taskManager : TaskManager
list : List
sd Login and Create new Task
Actor
ui : UI
loginController : LoginController
taskManager : TaskManager
list : List
1: Inputs username and password
1.1: LogUserIn()
1.2: returns request status
alt
[If Login is Successful]
2: displays list screen
3: displays bad login screen
4: hits add task button
4.1: displays create new task screen
5: inputs task parameters and presses save
5.1: AddTask()
5.1.1: adds the new task to the list NewTask()
5.2: displays list screen
Full Screen
Close Full Screen

User Interface Screen Prototypes



Contracts

+GetAllLocations() : ArrayList<Location> where GetAllLocations is a public instance method of Map class and returns a generic ArrayList containing the Location objects for every Task in the User's List that has location information not equal to null. If there are no Tasks that have location information, the method simply returns an empty ArrayList.

Precondition: User clicks the "Show All Locations" button

PostCondition: This method returns all of the locations so that they can be displayed on the map

+LogUserIn(userID : String, password : String): boolean where LogUserIn is a public instance method of LoginController class, userID is a String that contains a user's ID string, and password is a String that contains the user's attempted password. Method returns true if userID and password match an entry in the database, false if userID is not found or password does not match the password entry for a found userID in the database.

Precondition: false

Postcondition: None

+AddTask(taskLocation : Location, name : String, taskDate : Date, category : String): int where AddTask takes the input values from the user interface and creates a task object that is added to the current user's task list

Precondition: user has entered task information.

Postconditions: task is added to user's task list

+FilterList(listToFilter: List, filterType: int, searchTerm: String) : List where FilterList takes in a list and filters it in a specified way, then returns the list.

Precondition: unsorted list

Postcondition: list is in new sorted order, either by sorting according to the filterType (eg. alphabetical, distance to user, most recent, etc.) or by a search term (returns tasks relevant to the searchTerm)

Exceptions

Our major exceptions are expected to arise during interchanges between our program and either the user or the persistent storage. These two areas are very different, and as such, will be handled by two different types of error handling. Any method which takes user input will check the input string, and then it will return true if the input is valid or false if the input is invalid. This approach allows the code to easily know that the input may fail and will need to be re-acquired with the aid of useful error-correcting text displayed to the user. For database/file system errors, IO exceptions will be thrown and checked, as that is how the IO APIs work. Upon catching IO exceptions, the calling code will try again; if that fails, the program will abort the operation, and alert the user that the store/load was unsuccessful and that they should try again at a later time.

Updated Sprint Backlog

Sprint Backlog						
Tasks	Responsible	Status	Estimate	1	2	3
Team Contract:			1	1	0	0
- write contract	Caleb	Completed	1	1	0	0
Create Backlogs			1	1	0	0
- fill in hours	Caleb	Completed	1	1	0	0
ID Classes			1	1	1	0
- brainstorm requirements	Dan	Completed	1	1	1	0
Prepare CRC			2.5	2.5	2.5	0
- get physical cards	Jeff	Completed	0.5	0.5	0.5	0
- fill out role stereotypes	Sam	Completed	1	1	1	0
- fill out responsibilities/purpose	Sam	Completed	1	1	1	0
Write Use Cases			2	2	2	0
- create use case 1	Caleb	Completed	0.5	0.5	0.5	0
- create use case 2	Sam	Completed	0.5	0.5	0.5	0
- create use case 3	Jeff	Completed	0.5	0.5	0.5	0
- create use case 4	Dan	Completed	0.5	0.5	0.5	0
Create Architecture / Trust Boundaries			1	1	1	1
- create Architecture and Trust Boundaries	Sam	Completed	1	1	1	1
UML Diagrams			3.5	3.5	3.5	3.5
- download and learn how to use Visual Paradigm	Jeff/Caleb	Completed	0.5	0.5	0.5	0.5
- create UML Class diagram	Jeff/Caleb	Completed	2	2	2	2
- create UML Sequence diagram	Jeff/Caleb	Completed	1	1	1	1
User Interface Prototypes			1	1	1	0.75
- create handdrawn mockups of UI screens	Dan	Completed	1	1	1	0.75
Site Map / Page Navigation			1	1	1	0.25
- sketch sitemap (links between pages)	Dan	Completed	1	1	1	0.25
Method Contracts			4	4	4	4
- each team member writes a contract for a method	All	Completed	4	4	4	4
Exception Handling Strategy			2	2	2	2
- write out how to handle exceptions	Sam	Completed	2	2	2	2
Total			20	20	18	11.5