Textual Analysis

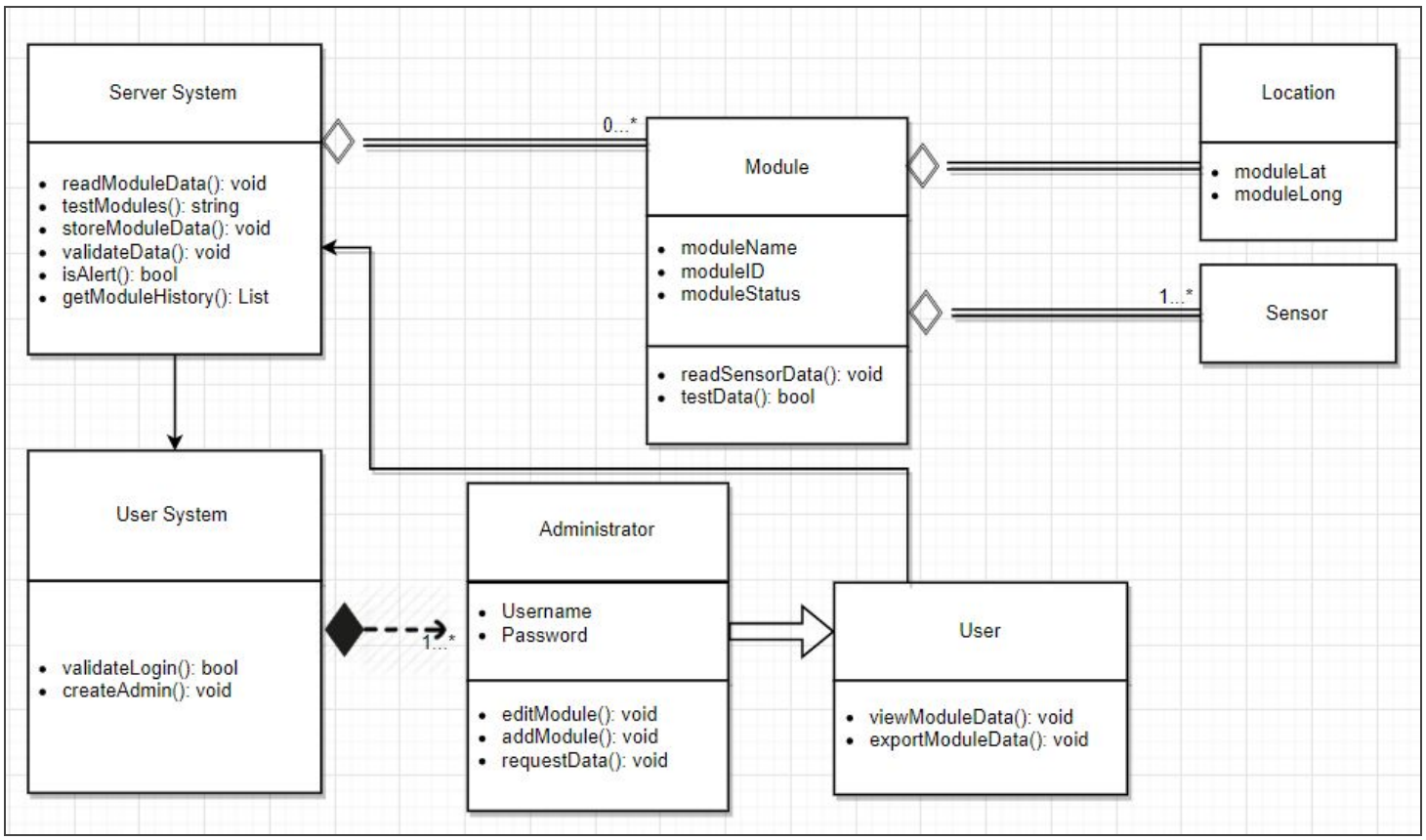
CS320 - 102: Software Engineering, Spring 2020

Due: 03/07/20

Dylan Bieber, David McHugh, Mikayla Trost, Trevor Gerst

Methods Classes Attributes Interfaces Unused

Nouns	Verbs	
user	validate	select
password	login (account)	store
module	browse	compare
username	returns (data)	stops (module)
administrator	solicit	toggle
account	add	edits (module)
email	tests (module)	prompt
error	tell	
reading	respond	
moduleName	remove	
location	alert	
moduleStatus	export	
moduleID	send	
moduleLat	create	
moduleLong	inputs	


Server System
readModuleData(): void
testModules(): string
storeModuleData(): void
validateData(): void
isAlert(): bool
getModuleHistory(): List
0...*
Module
moduleName
moduleID
moduleStatus
readSensorData(): void
testData(): bool
Location
moduleLat
moduleLong
1...*
Sensor
User System
validateLogin(): bool
createAdmin(): void
1...*
Administrator
Username
Password
editModule(): void
addModule(): void
requestData(): void
User
viewModuleData(): void
exportModuleData(): void

Our UML Diagram works off of 2 major sections; the server system and the user system. These two systems have simple but very important tasks in running the build of our web application. The server system is set up to control more of the daily tasks for our system. It will be in charge of the modules and all the information and duties that come with it. The client system will be in charge of keeping track of users and admins; it is there to validate login credentials so admins can have ease of access to our application features. Both these systems will know of each other but are not reliant on one another. In order to perform certain tasks, the server system will be in contact with the client system to check what privileges a user has been granted and if they may proceed.

The server system class is the core class of our web application. Most of the operations are either running from this class or dependent on this class. This class contains methods to obtain and store data from the modules along with methods to test the validity of the readings. This includes adding current data and retrieving past data from modules. This class serves as the middleman between the project's front end and back end.

The placement of our important methods are done so that most of our methods end up having to go through or communicate with the server system. When it comes to the user's methods, the user is able to view and export a module's air quality history. In our set up, the user's methods will have to go through the server system to retrieve the module history. The reason we did this was so the user is only able to communicate with the server system instead of being able to communicate with the modules specifically. This way the ease of access between what the user is able to do is simplified instead of allowing the user to access any of the module information directly.