UP Bike Share – Android App

Analysis Model

Submitted to:

Asst. Prof. Ma. Rowena C. Solamo
Faculty Member
Department of Computer Science
College of Engineering
University of the Philippines, Diliman

Submitted by:
Barozzo, Steven
Mamac, Mark Anton
San Gabriel, Jaypee Renz

In partial fulfillment of Academic Requirements
for the course
CS 191 Software Engineering I
of the
1st Semester, AY 2015-2016

Unique Reference:

The documents are stored in:
https://github.com/devsofup/UPBike-Share-Android/tree/master/03-Design-Engineering

Document Purpose:

This document serves to show the interactions of the different aspects of the system. It will guide the development team in creating the system.

Target Audience:

This document is mainly for the viewing of the development team. It will also be viewed by the guiding faculty. Possible users may also view this document but they are not part of the target audience, as it is an advanced topic.

Revision Control

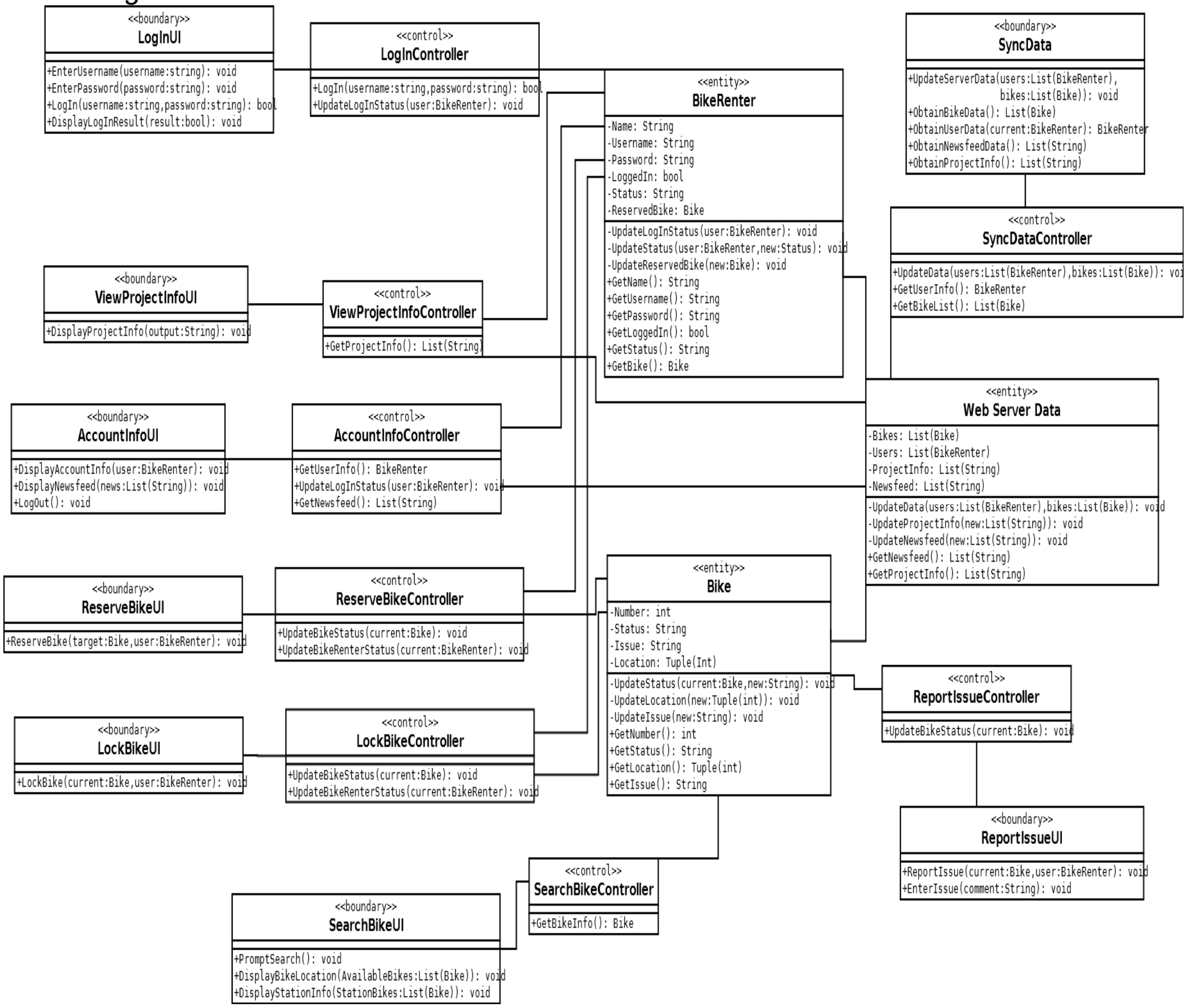
History Revision:

Revision Date	***Person Responsible***	***Version Number***	***Modification***
10/29/15	Jaypee San Gabriel	1.0	Initial Document. Created class diagram and descriptions for the classes.

System Name: Bike Sharing System

Description: The Bike Share system would allow a user, who would be a bike renter, to reserve bikes for travel within UP. Through the system, a bike renter after registering an account can log-in. He/she can then locate nearby bikes and reserve a bike from a particular station. The system accounts for possible problems encountered with the bikes, allowing a renter to return a broken bike, and relay information about its issues. A bike renter would also have information at hand such as the reports of other renters, information about the project, and news about the Bike Share project.

Class Diagram:



<<boundary>>
LogInUI
+EnterUsername(username:string): void
+EnterPassword(password:string): void
+LogIn(username:string,password:string): bool
+DisplayLogInResult(result:bool): void
<<control>>
LogInController
+LogIn(username:string,password:string): bool
+UpdateLogInStatus(user:BikeRenter): void
<<entity>>
BikeRenter
-Name: String
-Username: String
-Password: String
-LoggedIn: bool
-Status: String
-ReservedBike: Bike
-UpdateLogInStatus(user:BikeRenter): void
-UpdateStatus(user:BikeRenter,new:Status): void
-UpdateReservedBike(new:Bike): void
+GetName(): String
+GetUsername(): String
+GetPassword(): String
+GetLoggedIn(): bool
+GetStatus(): String
+GetBike(): Bike
<<boundary>>
SyncData
+UpdateServerData(users:List(BikeRenter),
bikes:List(Bike)): void
+ObtainBikeData(): List(Bike)
+ObtainUserData(current:BikeRenter): BikeRenter
+ObtainNewsfeedData(): List(String)
+ObtainProjectInfo(): List(String)
<<control>>
SyncDataController
+UpdateData(users:List(BikeRenter),bikes:List(Bike)): voi
+GetUserInfo(): BikeRenter
+GetBikeList(): List(Bike)
<<boundary>>
ViewProjectInfoUI
+DisplayProjectInfo(output:String): void
<<control>>
ViewProjectInfoController
+GetProjectInfo(): List(String)
<<entity>>
Web Server Data
-Bikes: List(Bike)
-Users: List(BikeRenter)
-ProjectInfo: List(String)
-Newsfeed: List(String)
-UpdateData(users:List(BikeRenter),bikes:List(Bike)): void
-UpdateProjectInfo(new:List(String)): void
-UpdateNewsfeed(new:List(String)): void
+GetNewsfeed(): List(String)
+GetProjectInfo(): List(String)
<<boundary>>
AccountInfoUI
+DisplayAccountInfo(user:BikeRenter): void
+DisplayNewsfeed(news:List(String)): void
+LogOut(): void
<<control>>
AccountInfoController
+GetUserInfo(): BikeRenter
+UpdateLogInStatus(user:BikeRenter): void
+GetNewsfeed(): List(String)
<<boundary>>
ReserveBikeUI
+ReserveBike(target:Bike,user:BikeRenter): void
<<control>>
ReserveBikeController
+UpdateBikeStatus(current:Bike): void
+UpdateBikeRenterStatus(current:BikeRenter): void
<<entity>>
Bike
-Number: int
-Status: String
-Issue: String
-Location: Tuple(Int)
-UpdateStatus(current:Bike,new:String): void
-UpdateLocation(new:Tuple(int)): void
-UpdateIssue(new:String): void
+GetNumber(): int
+GetStatus(): String
+GetLocation(): Tuple(int)
+GetIssue(): String
<<control>>
ReportIssueController
+UpdateBikeStatus(current:Bike): void
<<boundary>>
LockBikeUI
+LockBike(current:Bike,user:BikeRenter): void
<<control>>
LockBikeController
+UpdateBikeStatus(current:Bike): void
+UpdateBikeRenterStatus(current:BikeRenter): void
<<boundary>>
ReportIssueUI
+ReportIssue(current:Bike,user:BikeRenter): void
+EnterIssue(comment:String): void
<<control>>
SearchBikeController
+GetBikeInfo(): Bike
<<boundary>>
SearchBikeUI
+PromptSearch(): void
+DisplayBikeLocation(AvailableBikes:List(Bike)): void
+DisplayStationInfo(StationBikes:List(Bike)): void

Boundary Classes:

Class Name	Description
ViewProjectInfoUI	A small part that any user can interact with even while not logged in. It displays information about the Bike Share project along with the latest news about the project.
LogInUI	Interacts with the user/bike renter. Allows the user to log in provided he/she has a registered account.
AccountInfoUI	Interacts with the user/bike renter. The user can view information in his/her account such as the current status, a Bike Use History, and a Newsfeed. The user can also log out.
SearchBikeUI	Interacts with the user/bike renter. The user can prompt the system to search for bikes, displaying nearby stations and their respective available bikes.
ReserveBikeUI	Interacts with the user/bike renter. The user can choose and reserve a particular bike.
LockBikeUI	Interacts with the user/bike renter. When the user is finished renting a bike he/she can lock it back at a specific station by prompting it through the system.
ReportIssueUI	Interacts with the user/bike renter. The user can report an issue with a bike; he/she is able to write a comment to detail the problem.
SyncData	Interacts with the web server. The web server contains the database, and thus there is a need to synchronize the data within the system and that of the true database within the web server. This boundary class sends updated data to the web server and also obtains data from the web server.

Control Classes:

Class Name	**Description**
ViewProjectInfoController	Responsible for obtaining the latest information available from the web server.
LogInController	Responsible for the process of logging in, updating the log in status of the user’s account upon success or sending back a message upon failure.
AccountInfoController	Responsible for obtaining the user’s information, the latest Newsfeed and the process of logging out. It updates the log in status of the user’s account upon success in logging out.
SearchBikeController	Responsible for obtaining the information of bikes.
ReserveBikeController	Responsible for updating both the user’s status and the respective bike’s status once there is a request for bike reservation.
LockBikeController	Responsible for updating both the user’s status and the respective bike’s status when the user finishes renting a bike.
ReportIssueController	Responsible for updating the bike’s status when a comment is inputted for said bike.
SyncDataController	Responsible for updating the inner data using the obtained data from the web server. It also obtains inner data so that the data in the web server can be updated.

Entity Classes:

Class Name	**Description**
BikeRenter	Contains information on the users of the system. This includes their name, username, password, whether they are currently reserving a bike or not, and which bike they have reserved if applicable.
Bike	Contains information on the various bikes of the project. This includes the bike number, its status or whether it is Available, Reserved, Under Investigation or Invalid, any comment on its problems if applicable, and its location.
Web Server Data	Holds other miscellaneous information such as the project information and the newsfeed. It also keeps a reference to the full list of bikes and the bike renter in the inner system so as to easily update the data in the actual web server.