IskUber
Software Architecture

Submitted to:

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

Submitted by:
Bilaw, Nicole
del Rosario, Luis Gabriel
Tamayo, Juan Gabriel

In partial fulfillment of Academic Requirements
for the course
CS 191 Software Engineering I
of the
1st Semester, AY 2017-2018

Revision Control

History Revision:

Revision Date	***Person Responsible***	***Version Number***	***Modification***
12/01/2017	Nicole Bilaw	1.0	Initial Document; Added Software Architecture Diagram; Added User Interface Classes; Added Business Logic Classes; Added Data Design Classes

Purpose:

This document serves as the official Software Architecture document for IskUber. This document provides a comprehensive overview of the architecture of the software system. It serves as a communication medium between the software architect and other project team members regarding architecturally significant decisions which have been made on the project.

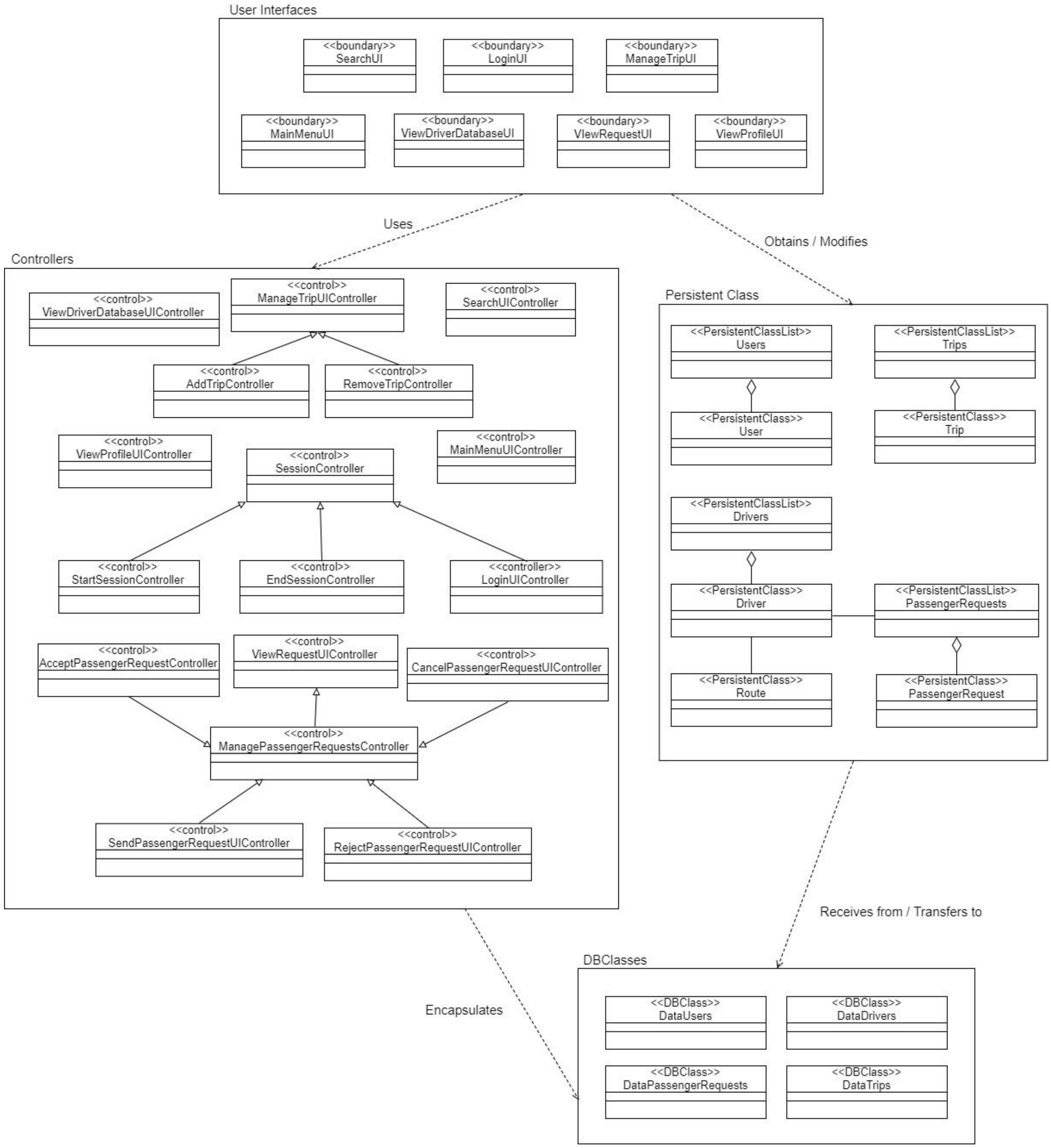
Audience:

This document is targeted to the team members of the project and the software architect of the project.

System Name: IskUber

Description: IskUber is a transportation mobile application that serves as the medium of communication between UP Diliman students who are commuters and students who are drivers. Students who want to get to a certain location inside the UPD campus can use the app to search and contact drivers to get them where they need to be. IskUber is an alternative to the Ikot and Toki jeeps inside the campus.

Software Architecture Diagram:


User Interfaces
<<boundary>> SearchUI
<<boundary>> LoginUI
<<boundary>> ManageTripUI
<<boundary>> MainMenuUI
<<boundary>> ViewDriverDatabaseUI
<<boundary>> VIewRequestUI
<<boundary>> ViewProfileUI
Uses
Obtains / Modifies
Controllers
<<control>> ViewDriverDatabaseUIController
<<control>> ManageTripUIController
<<control>> SearchUIController
<<control>> AddTripController
<<control>> RemoveTripController
<<control>> ViewProfileUIController
<<control>> SessionController
<<control>> MainMenuUIController
<<control>> StartSessionController
<<control>> EndSessionController
<<controller>> LoginUIController
<<control>> AcceptPassengerRequestController
<<control>> ViewRequestUIController
<<control>> CancelPassengerRequestUIController
<<control>> ManagePassengerRequestsController
<<control>> SendPassengerRequestUIController
<<control>> RejectPassengerRequestUIController
Persistent Class
<<PersistentClassList>> Users
<<PersistentClassList>> Trips
<<PersistentClass>> User
<<PersistentClass>> Trip
<<PersistentClassList>> Drivers
<<PersistentClass>> Driver
<<PersistentClassList>> PassengerRequests
<<PersistentClass>> Route
<<PersistentClass>> PassengerRequest
Receives from / Transfers to
Encapsulates
DBClasses
<<DBClass>> DataUsers
<<DBClass>> DataDrivers
<<DBClass>> DataPassengerRequests
<<DBClass>> DataTrips

User Interface Design Classes:

Screen Name	**Description**
SearchUI	This class displays the search menu of drivers.
LoginUI	This class displays the login menu of the system.
ManageTripUI	This class displays the user interface for when the driver wants to manage his trips.
MainMenuUI	This class displays the main menu of the application.
ViewDatabaseUI	This class displays the list of all drivers registered to the system.
ViewRequestUI	This class displays the list of passenger requests of the driver.
ViewProfileUI	This class displays the profile of a driver.

Business Logic Classes:

Class Name	**Description**
LoginUIController	This control class can be performed by the user to log-in to the system.
SessionController	This control class handles the user login sessions.
StartSessionController	This control class activates during the login process.
EndSessionController	This control class activates during the logout process.
MainMenuUIController	This control class can be performed by the user to navigate the main menu options of the system.
ViewRequestUIController	This control class can be performed by the user to view the current requests.
ViewProfileUIController	This control class can be performed by the user to view his/her profile.
SearchUIController	This control class can be performed by the user to search for drivers.
ViewDriverDatabaseUIController	This control class can be performed by the user to view the list of all registered drivers in the database.
ManageTripUIController	This control class can be performed by the user to manage his/her current trips.
AddTripController	This control class is automatically activated upon accepting a passenger request.
RemoveTripController	This control class can be performed by the user to remove any of their current trips.
ManagePassengerRequestsController	This control class handles the driver and passenger requests.
AcceptPassengerRequestController	This control class can be performed by the driver to accept a passenger request.
RejectPassengerRequestController	This control class can be performed by the driver to reject a passenger request.
SendPassengerRequestController	This control class can be performed by the user to send a passenger request to drivers.
CancelPassengerRequestController	This control class can be performed by the user to cancel a passenger request.

Data Design Classes:

Class Name	**Description**
DataUsers	This class is in charge of managing the users of the system. DataUsers may add, edit or delete users from the database. Relevant information for users are as follows: username, full name, address, telephone, email address, birthday.
DataDrivers	This class is in charge of managing the drivers of the system. DataDrivers may add, edit or delete drivers from the database. Relevant information for drivers are as follows: username, full name, address, telephone, email address, birthday, and the route the driver is taking.
DataTrips	This class is in charge of managing the overall list of trips in the system. Variables such as user involved, driver involved, and path information are stored here.
DataPassengerRequests	This class is in charge of managing the overall list of passenger requests in the system. Each request consists of specific IDs that map it to specific users and drivers.