

# Phase 2

# SBM2S

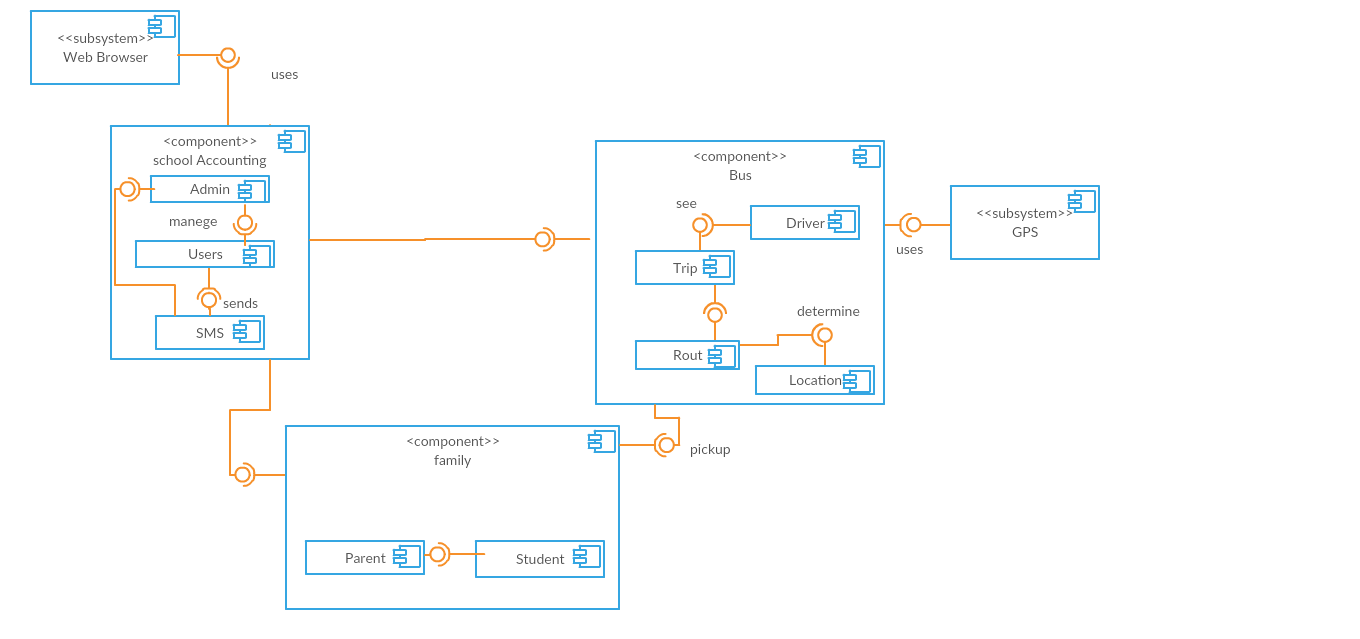
# 2017/april/9

|  |
| --- |
| **Student Name** |
| Norah alkhunifer |
| Raghad alsaleh |
| Reema al-swailam |

|  |  |
| --- | --- |
| Contents | page |
| Component diagram | 2 |
| deployment diagram | 3 |
| Sequence diagrams | 4-5 |
| VOPC diagrams | 6-7 |
| Chosen pattern | 8 |

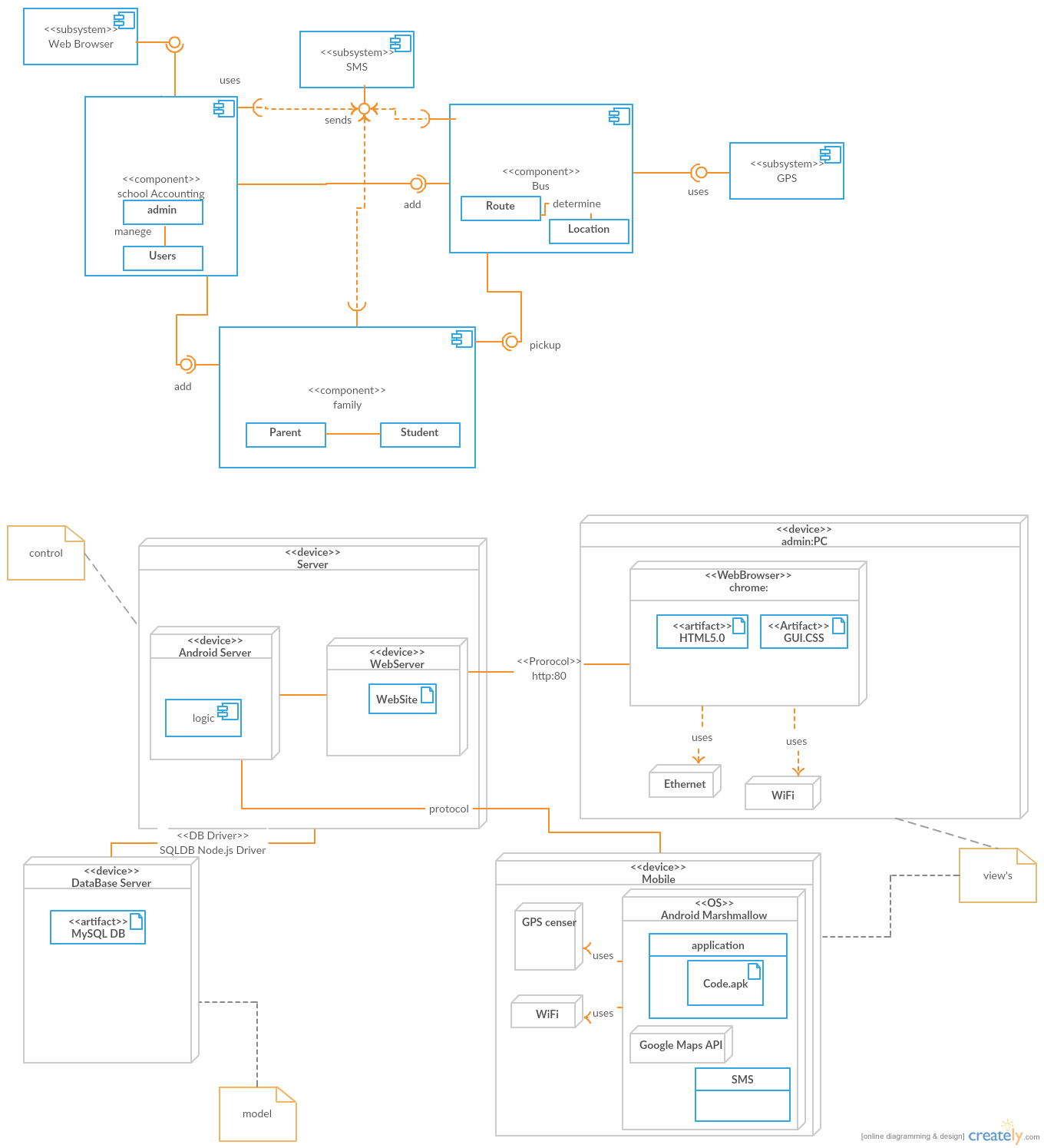


# Component diagram



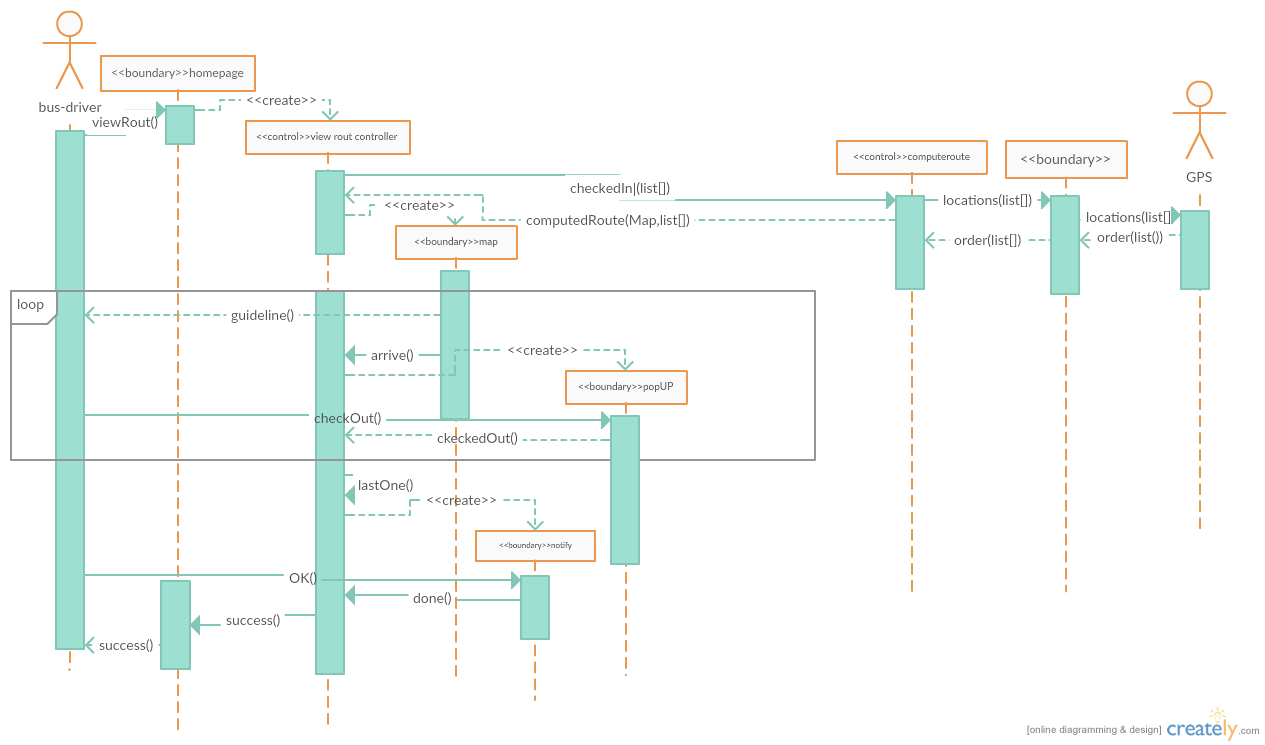


# deployment diagram



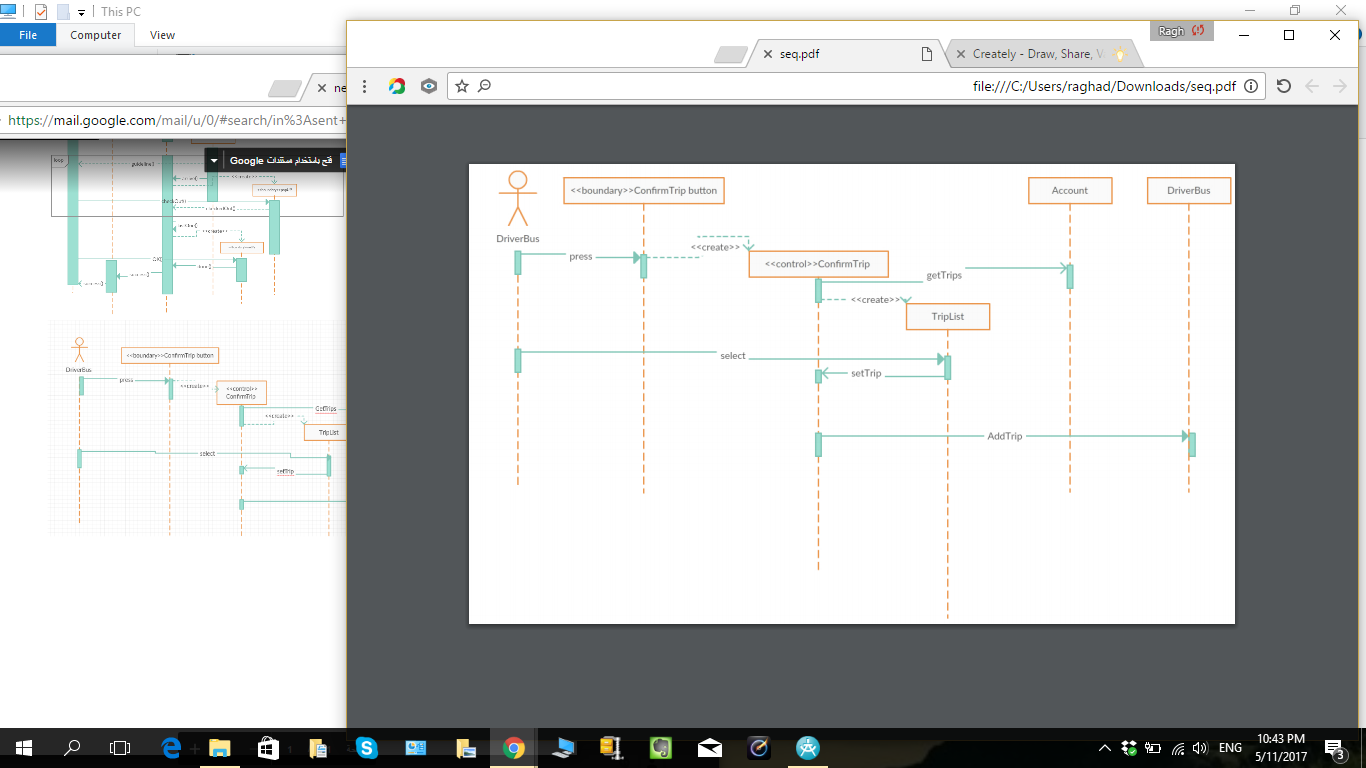


# Sequence diagrams



1.view route sequence diagram

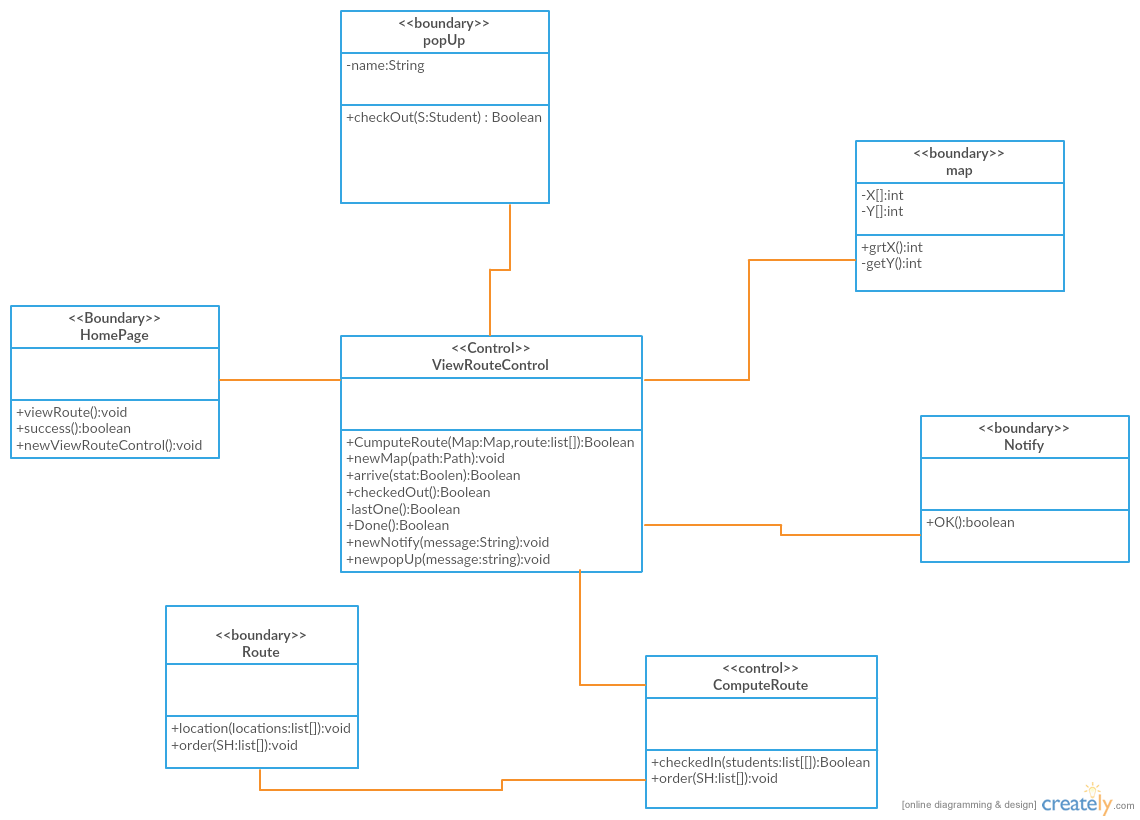


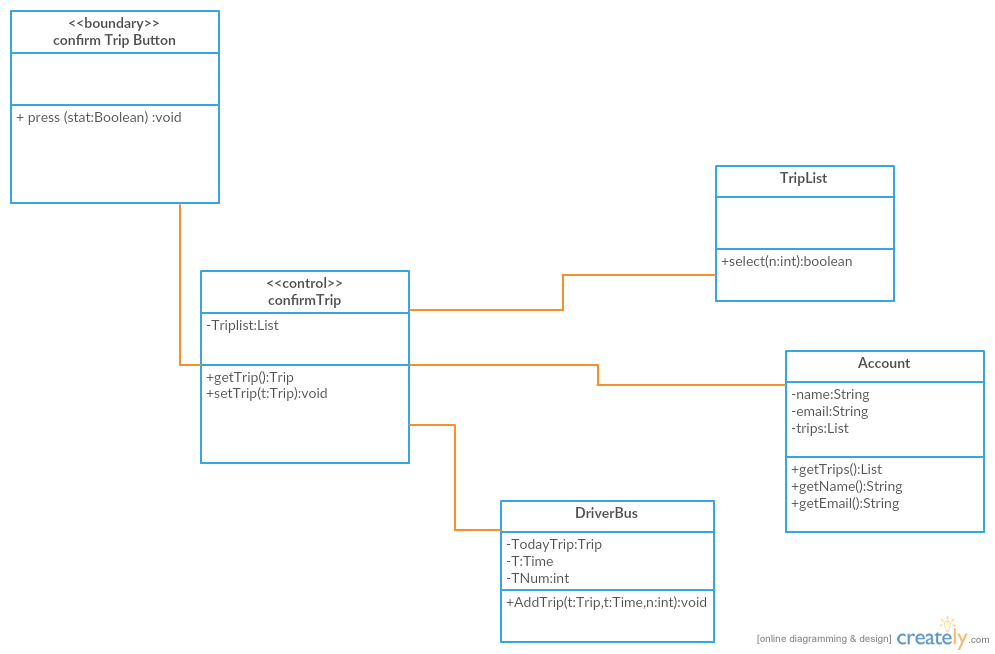


2.confern Trip sequence diagram



# VOPC diagrams

****1. view route VOPC diagram



2. confirm Trip VOPC diagram



# a design pattern of the Application

In our system I choose observer pattern

**First** since we choose MVC “model-view-controller” architectural for our system “SBM2S” the most Appropriate design pattern is observer pattern .

**Secondly** we have server , which provides data to different users . A user implemented as web application(admin) other mobile devices(bus driver and parent ) and the system need to notify the users with SMS we need to separate server from it’s observers (users) by adding new observer in the future will transparent for the server .

advantages:

* Supports the principle to strive for loosely coupled designs between objects that interact.
* Allows you to send data to many other objects in a very efficient mannor.
* No modification is need to be done to the subject to add new observers.
* You can add and remove observers at anytime.

disadvantages:

* Java’s built in class Observable forces the use of inheritance vs programming to an interface
* Observable protects crucial methods which means you can’t even create an instance of the Observable class and compose it with your own objects, you have to subclass.
* If not used carefully the observer pattern can add unnecessary complexity
* The order of Observer notifications is undependable