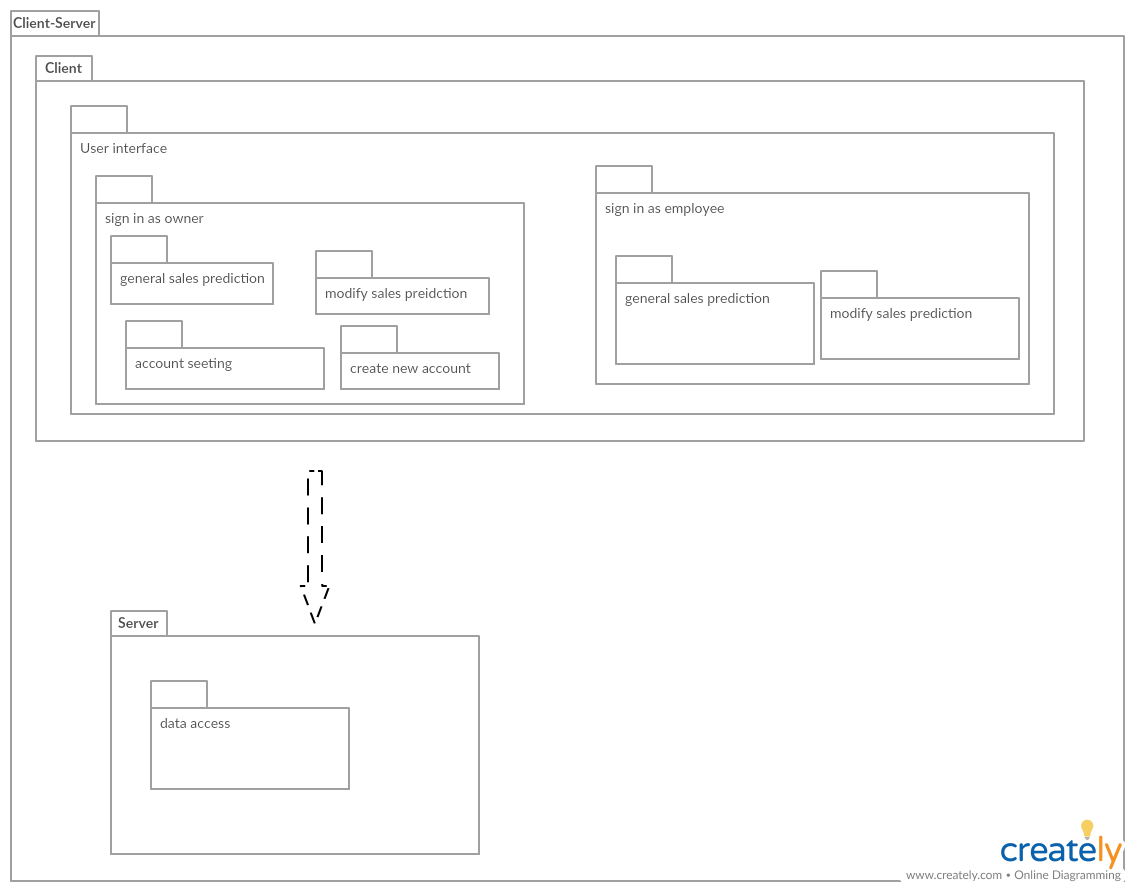
Software architecture



The architecture pattern we choose is Client-Server. As a client, go to the sign in screen and due to the different position: owner of employee, User may go to different screens.

As owners, they will have two more options, account settings and create new account. When user go to general sales prediction, it will connect to the server and according to the statistics from the previous gross sales. The system will predicate the period of average gross sales. The owner can also choose the date to input the gross sales of that day. The owner can choose account setting to show all the username, password and position which were stored in Database. The owner can click create new account, after entering user name, password and choose a role then the new user’s all information will be stored in database

As employees, they just have two options, general sale prediction and upload sales data which are same with the owners.

Design pattern:

**Creational pattern:** Inabstract factory, it provides an interface for creating families of related of dependent objects, but we do not have any interface to use. In builder design pattern, we also need to have an interface to build a complex object using simple objects, but in our project, we did not use either. In factory method, define an interface for creating an object, but let subclasses decide which class to instantiate and we do use either. In prototype pattern, we need to have an abstract class, but we do not any abstract class we do not use that pattern either. In **Singleton pattern**, it involves only one class which is responsible to instantiate itself. In our application, we have a branch of static classes and single object can be used by all other classes.

**public** String getUser() {  
        **return user**.get();  
    }  
  
    **public** String getPass() {  
        **return pass**.get();  
    }  
  
    **public** String getPos() {  
        **return pos**.get();  
    }

*//updates an entire month for the new data input***public static boolean** MonthUpdate(String date) {  
   String month = *WhatMonth*(date);  
   String array[] = date.split(**"/"**);  
   System.***out***.println(month);  
   String days[] = {**"mon"**, **"tue"**, **"wed"**, **"thu"**, **"fri"**, **"sat"**, **"sun"**};  
   ArrayList<dailyavg> hold = **new** ArrayList<dailyavg>();  
   hold = DbManager.*filllist*();  
   *//String months[] = { "01", "02", "03", "04", "05", "06", "07", "08", "09", "10", "11", "12" };* **float** total = 0;  
   **int** count = 0;  
   **float** checker = 0;  
  
   **for** (**int** z = 1; z <= 5; z++)  
      **for** (**int** y = 0; y <= 6; y++) {  
         **float** avg = DbManager.*GetAvg*(month, z + days[y]);  
         String query = **"SELECT** *\** **FROM "** + month + **" WHERE DayOfMonth = '"** + z + days[y] + **"'"**;  
         *//System.out.println(z+days[y]);* **float** high = 0;  
         **try** {  
            avg = DbManager.*GetAvg*(month, z + days[y]);  
            ResultSet rs = *st*.executeQuery(query);  
            *//System.out.println("Hello");* **if** (rs.next())  
               checker = rs.getFloat(**"AvgGrossSales"**);  
            rs.close();  
            **if** (checker > 0)  
               avg = checker;  
         } **catch** (SQLException e) {  
            System.***out***.println(e);  
            *//continue;* **return false**;  
  
         }  
         query = **new** String();  
         query = **"SELECT** *\** **FROM dailyinformation WHERE Date LIKE '"** + array[0] + **"%' AND DayOfMonth = '"** + z + days[y] + **"'"**;  
         System.***out***.println(array[0]);  
         **try** {  
            ResultSet rs = *st*.executeQuery(query);  
            **if** (rs.next()) {  
               high = rs.getFloat(**"GrossSales"**);  
               DbManager.*UpdateOneDay*(month, high, z + days[y], avg);  
               System.***out***.println(high);  
            }  
            rs.close();  
         } **catch** (SQLException e) {  
            System.***out***.println(e);  
         }  
  
  
         *//System.out.println(temp.grosssales);* DbManager.*UpdateOneDay*(month, high, z + days[y], avg);  
  
      }  
   **return true**;  
}  
*//****todo: update every year***