**Xdel Documentation**

**Content Page**

**A. Front end and Code Behind**

1. Show.aspx & Show.aspx.cs
2. Prompt.aspx & Prompt.aspx.cs
3. Validate.aspx & Validate.aspx.cs
4. Dashboard.aspx & Dashboard.aspx.cs
5. ManageCluster.aspx & ManageCluster.aspx.cs
6. ManageUser.aspx & ManageUser.aspx.cs
7. ManageDriver.aspx & ManageDriver.aspx.cs
8. Include.aspx & Include.aspx.cs
9. Login.aspx & Login.aspx.cs

**B. Controller and DAO [DAOs are where needs to be edited when adding database]**

1. PostalCodeInitializer [Controller]
2. ClusterDAO
3. DriverDAO
4. PolygonODAO
5. UserDAO

**C. Model**

1. Cluster
2. ClusterDetails
3. Driver
4. PointD
5. PolygonO
6. PostalJob
7. User

**D. Libraries & Plugins used**

1. Important CSS files to note
2. W3.css
3. Important JS file to note
4. For The Leaflet Map
5. When first open project

**E. Singapore Postal Code Boundary Details**

1. Singapore Postal Code Boundary Details – [methods on how to create new postal code boundary.csv]

**Front End and Code Behind**

**Show.aspx**

7 Hidden Fields

1. HiddenField1

HiddenField1 contains a string of driver locations

1. HiddenField2

HiddenField2 contains a string of traffic incidents from LTA

1. HiddenField3

HiddenField3 contains a string of driver routes

1. HiddenField4

HiddenField4 contains a string of jobs for clustering

Buttons that do a postback

1. ChkTraffic does a postback, invoking the method getTrafficMethod().
2. toggleRoute does a postback, invoking the method route\_click().
3. The checkboxlist1 is beside the names of the drivers. Upon selected it will do a postback, invoking the method CheckBoxList\_Click() method.
4. Polygonclick turns the selected polygon white. Upon clicking a single polygon, it will do a postback invoking the method Polygon\_click().
5. Button1 is a submit button under the zone tab. Upon clicking submit, it will do a postback and invoke the method Color\_Click which changes the color of the polygon to the selected zone.

**Show.aspx.cs**

21 methods

1. getDriversByCoordinator()

Return type: ArrayList. This method returns an ArrayList of drivers specific to the user that is logged in.

1. getDriverArray()

Return type: LukeRefL2.DriverObject[]. This method returns an array of all the drivers through invoking the webservice GetL2Drivers.

1. getTrafficConditions()

Return type: Void. This method concatenates and assigns all the LTA incident message, latitude and longitude to HiddenField2.

1. retrieveClusters()

Return type: ArrayList of clusters (cluster class). This method returns an ArrayList of clusters (cluster class) kept within ClusterDAO.

1. getListOfClusterId()

Return type: ArrayList of cluster’s ID (String). This method returns an ArrayList of cluster’s ID (String) to be used to bind within Grouping (Dropdownlist) in Page\_Load().

1. Route\_Click()

Return type: Void. This method uses ST\_GetSol from the web service to obtain the routes of all the drivers. It then concatenates into a long string that contains a particular job’s postal code, JobIDX, full address, latitude and longitude, arrive time and depart time. The long string is assigned to HiddenField3.

1. CheckBoxList\_Click()

Return type: Void. This method concatenates driver’s name, latitude and longitude. It is assigned to HiddenField1 which is used in the code in front to display the driver’s location on the leaflet map upon clicking the checkbox beside their name.

1. GetPolygons()

Return type: ArrayList. This method invokes GetPostalCodes() method from the model PostalCodeInitializer and returns an ArrayList of polygons.

1. Color\_Click()

Return type: Void. This method is triggered from a button (change of cluster) from Show.aspx. This method invokes changeCluster from PostalCodeInitializer with a parameter of selected value from grouping (dropdownlist).

1. Polygon\_Click()

Return type: Void. This method is triggered from clicking on any cluster from Show.aspx. This method invokes changePostal from PostalCodeInitializer with a parameter of cluster id from the cluster selected.

1. getClusters()

Return type: ArrayList of clusters (cluster class). This method returns an ArrayList of clusters (cluster class) kept within ClusterDAO.

1. getClusterId()

Return type: cluster’s id (string). This method invokes the getClusterId() from PostalCodeInitializer with a parameter of postal code id. This method will eventually returns cluster’s ID based on the postal code entered.

1. getColor

Return type: cluster’s color (string). This method invokes the getColor() from PostalCodeInitializer with a parameter of postal code id. This method will eventually returns cluster’s color based on the postal code entered.

1. getClusterDetails()

Return type: ArrayList of clusterDetail (clusterDetail class). This method invokes getClusterLatCenter and getClusterLngCenter to retrieve the middle point of all postal codes within a specific cluster and stores them in clusterDetail. This method also invokes getPostalJobs to get the numbers of new job, pip and dip per postal code. This method then add the numbers of new job, pip and dip into the clusterDetail so that we can retrieve the total number of new job, pip and dip in show.aspx.

1. getPostalJobs()

Return type: ArrayList of postalJobs (postalJobs class). This method invokes Luke getDistrictJobs to retrieve all districtJobs from luke. This method further breaks the districtJobs into different jobs such as newjob, pip and dip and store them into postalJobs. This method also track if user logged in have the rights to view the specific postal code district.

1. uncheckAll\_Click()

Return type: Void. This method clears and uncheck everything on the check box list beside the drivers’ name. It also assigns HiddenField1 an empty value.

1. selectAll\_Click()

Return type: Void. This method checks everything on the check box list and populates HiddenField1 with the location of all the drivers. Similar to CheckBoxList\_Click method above.

1. getBattery()

Return type: Void. This method obtains the battery level of all the drivers’ handheld devices. It then places it in a session.

1. logout()

Return type: Void. This method clears all session and logs the current user out.

1. checkPostalCodeBoundaryStatus()

Return type: Boolean. Check if hidden status is true or false;

**Validate.aspx**

12 Hidden Fields

1. HiddenField1

HiddenField1 contains a string of driver locations.

1. HiddenField3

HiddenField3 contains a string of all the new job locations.

1. Driver1/2/3/4/5

Driver1/2/3/4/5 contains the existing solution from the RSVP along with all the details regarding the top 5 drivers. This includes, name, distance away from the job selected, job id of its current jobs, postal code, latitude, longitude, arrive time, depart time, pick up or delivery job.

1. IDX2/3/4/5

IDX2/3/4/5 contains a string that is similar to Driver1/2/3/4/5, except this one contains the temporary and validated solution using the VRP.

1. NewJobsIDX

NewJobsIDX contains a string that has the Job IDX of the job selected on the radio list.

Buttons that do a postback

None

**Validate.aspx.cs**

10 methods

1. getDriverArray()

Return type: LukeRefL2.DriverObject[]. This method returns an array of all the drivers through invoking the webservice GetL2Drivers.

2. getBattery()

Return type: Void. This method obtains the battery level of all the drivers’ handheld devices. It then places it in a session.

3. getTempSolution()

Return type: String. This method returns the temporary calculated solution using the VRP. The string contains, job IDX, job postal code, job full address, arrival time, departure time and job type.

4. getSolution()

Return type: String. This method returns the actual calculated solution using the VRP. The string contains, job IDX, job postal code, job full address, arrival time, departure time and job type.

5. RadioList\_Click()

Return type: Void. This method will select the top 5 nearest drivers, invoke the methods getTempSolution() & getSolution and assign the values for HiddenField1, HiddenField3, Driver1, Driver2, Driver3, Driver4, Driver5, IDX2, IDX3, IDX4 & IDX5.

6. ValidateSecond()

Return type: Void. This method assigns IDX2’s(the temporary validated solution) value to Driver2 for display after clicking the button validate.

7. ValidateThird()

Return type: Void. This method assigns IDX3’s(the temporary validated solution) value to Driver3 for display after clicking the button validate.

8. ValidateFourth()

Return type: Void. This method assigns IDX4’s(the temporary validated solution) value to Driver4 for display after clicking the button validate.

9. ValidateFifth()

Return type: Void. This method assigns IDX5’s(the temporary validated solution) value to Driver5 for display after clicking the button validate.

10. Logout()

Return type: Void. This method clears all session and logs the current user out.

**Prompt.aspx**

7 Hidden Fields

<!-- Delay field -->

<asp:HiddenField ID="Delays" runat="server" />

<!-- red delays -->

<asp:HiddenField ID="RedDelays" runat="server" />

<asp:HiddenField ID="RedDelaysLatLong" runat="server" />

<!-- amber delays -->

<asp:HiddenField ID="AmberDelays" runat="server" />

<asp:HiddenField ID="AmberDelaysLatLong" runat="server" />

<!-- green delays -->

<asp:HiddenField ID="GreenDelays" runat="server" />

<asp:HiddenField ID="GreenDelaysLatLong" runat="server" />

Buttons that do a postback

None

**Prompt.aspx.cs**

3 methods

1. getBattery()

Get Battery saves the drivers who’s BATTPCT < 50% in an ArrayList into a session

1. getDelays()

getDelays will calculate the red(>45 mins late) amber(>15 <45) and green(<15) delays which will then save into the different hidden fields this method will be called at pageload

1. logout()

Logout button to redirect to login page

**Dashboard.aspx (only have template)**

0 Hidden Fields

Buttons that do a postback

**None**

**Dashboard.aspx.cs**

2 methods

1. getDriverArray()

returns all driver in an array

1. getBattery()

Get Battery saves the drivers who’s BATTPCT < 50% in an ArrayList into a session

**ManageCluster.aspx**

Hidden Fields

None

Buttons that do a postback

1. postal does a postback, invoking the method addPostal() at code behind.

**ManageCluster.aspx.cs**

6 methods

1. getDriverArray()

returns all driver in an array

1. getBattery()

Get Battery saves the drivers who’s BATTPCT < 50% in an ArrayList into a session

1. logout()

Logout button to redirect to login page

1. retrieveCluster()

returns an ArrayList of clusters stored in clusterDAO (except the last one – the last one is “white” border. To be used for highlighting selected clusters)

1. getListOfClusterId()

returns an ArrayList of clusters id (String) that user logged in has the rights for

1. addPostal()

Invokes addPostal in PostalCodeInitializer to edit a postal code from its current cluster to the selected cluster.

**ManageUser.aspx**

Hidden Fields

None

Buttons that do a postback

1. editUserButton does a postback, invoking the method editUser\_Click() at code behind.

**ManageUser.aspx.cs**

10 methods

1. getDriverArray()

returns all driver in an array

1. getBattery()

Get Battery saves the drivers who’s BATTPCT < 50% in an ArrayList into a session

1. logout()

Logout button to redirect to login page

1. retrieveUserInfo()

returns an ArrayList of an ArrayList of users details such as (username [string], role [string] and postalcoderights [arraylist of string]) stored in userDAO.

1. retrieveClusters()

returns an ArrayList of clusters id (String) that user logged in has the rights for

1. retrieveUsers()

returns an ArrayList of users (User Class) stored in usersDAO by invoking PostalCodeInitializer’s retrieveAllUsers().

1. editUser\_Click()

edit user’s details such as role and postal code rights by invoking PostalCodeInitializer’s updateUser

1. getListofUserNames()

returns an ArrayList of usernames (String) that User who logged in has the rights to edit.

1. getListofRole()

returns an ArrayList of roles (String) that User who logged in has the rights to edit.

1. getListofRights()

returns an ArrayList of postalCodeRights (String) that User who logged in has the rights to edit.

**ManageDriver.aspx**

Hidden Fields

None

Buttons that do a postback

1. DriverNameButton does a postback, invoking the method driverName\_Click() at code behind.

**ManageDrivers.aspx.cs**

6 methods

1. getDriverList()

returns an ArrayList of all drivers (Driver Class) stored in DriverDAO by invoking getAllDrivers from PostalCodeInitializer

1. getListOfDriversNames()

returns an ArrayList of drivers’s name (String) based on the user logged in rights.

1. getListOfUsers()

returns an ArrayList of all user’s username (String)

1. getDriverArray()

returns all driver in an array

1. getBattery()

Get Battery saves the drivers who’s BATTPCT < 50% in an ArrayList into a session

1. logout()

Logout button to redirect to login page

**Include.aspx**

Hidden Fields

None

Buttons that do a postback

None

**Include.aspx.cs**

Include.aspx.cs acts as a protect page to guard against users from typing a specific URL without authenticating from login.aspx

**Login.aspx**

Hidden Fields

None

Buttons that do a postback

1. Login button does a postback, invoking the method Authenticate () at code behind.

**Login.aspx.cs**

1 method

1. Authenticate()

Authenticates login. If user is authenticated, store user in session and redirect user to show. Else redirect user back to login with an error message.

**Controller and DAO**

**PostalCodeInitializer.cs [this is where most logic of the project lies and all linkage between code behind and DAOs]**

14 methods

1. InitialisePostalCodes()

Create and initialise new polygonODAO, ClusterDAO, UserDAO and DriverDAO

[Due to no database, data initialised are all hard coded. Once database is linked. There is no need for this method]

1. getClusters()

returns an ArrayList of Clusters (cluster class) from ClusterDAO

1. getPostalCodes()

returns an ArrayList of Polygons (PolygonO class) from PolygonODAO

1. changePostal()

edit a specific postal code from it’s current cluster into the selected clusters using changePostal in ClusterDAO

1. changeCluster()

edit all selected clusters (clusters that have been turned white) from it’s current cluster into the selected clusters using changeCluster in ClusterDAO

1. getColor()

returns a cluster’s color (string) based on postal code id (int) by using getColor from ClusterDAO

1. updateUser()

edit user role and rights based on username by using updateUser from UserDAO

1. retrieveAllUsers()

returns an ArrayList of Users (User class) stored in UserDAO by using retrieveUsers from UserDAO

1. getAllDrivers()

returns an ArrayList of Drivers (Driver class) stored in DriverDAO by using getDrivers from DriverDAO

1. changeDriverCluster()

edit driver’s coordinator based on driver’s name by using editDriver from DriverDAO

1. getClusterLatCenter()

return a latitude value (double) based on cluster id. This method will calculate the average latitude of all middle point of each postal code (by using getLatcenterOfPolygon from PolygonODAO) within the cluster.

Calculation: Total Latitude Value / Total Number of Postal Code within Cluster

1. getClusterLngCenter()

return a longitude value (double) based on cluster id. This method will calculate the average longitude of all middle point of each postal code (by using getLngcenterOfPolygon from PolygonODAO) within the cluster.

Calculation: Total Longitude Value / Total Number of Postal Code within Cluster

1. readCoordinates()

retrieve the postal code boundary coordinates by reading file from the Postal Code Boundary Coordinates.csv. All coordinates will then be stored in each polygonO class stored in PoylgonODAO with the following parameters (postal code, latitude, longitude).

1. addPostal()

edit the postal code’s cluster from its original cluster into the new cluster [aka zone] by using changePostal in ClusterDAO.

**ClusterDAO.cs**

DATABASE UPDATES SHOULD BE EDITED IN THIS FILE

9 methods

1. ClusterDAO() & InitialiseCluster()

Hard coded data to be used for display. Finalised database should store similar data [as zoning is retrieve from Peter’s current excel sheet]

1. getClusters()

returns clusters from ClusterDAO. This code needs to be edited to retrieve all clusters data from Cluster TABLE in database

1. trackPostal()

returns cluster that contains specific postal code. This code needs to be edited to retrieve cluster from Cluster TABLE in database

1. getColor()

returns color of cluster that contains specific postal code. This code needs to be edited to retrieve cluster’s color from Cluster TABLE in database

1. changePostal()

update the postal code’s cluster to the new given zone. This code needs to be edited to remove postal code from its cluster and add postal code into the specific cluster from Cluster TABLE in database

1. trackCluster()

returns cluster that contains specific cluster id. This code needs to be edited to retrieve cluster data that contains specific cluster id from Cluster TABLE in database

1. getClusterId()

returns cluster id that contains specific postal code id. This code needs to be edited to retrieve cluster id that contains specific postal code id from Cluster TABLE in database

1. changeCluster()

update all clusters that are currently “white” (selected by the users) to the new specific cluster id that user selects. This code needs to be edited to update ALL cluster that has id:white to the specific cluster id user select from Cluster TABLE in database

1. clusterExist()

returns if cluster exist within clusterDAO. This code needs to be edited to retrieve if a specific cluster id exist in Cluster TABLE in database

**DriverDAO.cs**

DATABASE UPDATES SHOULD BE EDITED IN THIS FILE

7 methods

1. DriverDAO() & InitialiseDrivers()

Driver data are pulled from current webservice. However due to Peter’s requirement of each driver is tied to 1 coordinator. I’ve hardcoded each driver to add a random coordinator to each driver. Finalised database should still retrieve driver data from current database. However the current database of driver needs to add a new coordinator field.

1. getDriver()

returns drivers from DriverDAO. This code needs to be edited to retrieve all drivers data from Driver TABLE in database

1. addDriver()

add new driver into drivers in DriverDAO. This code needs to be edited to add new line of driver into Driver TABLE in database

1. editDriver()

edit the driver’s coordinator from one user to another. This code needs to be edited to update driver’s coordinator to the new specific coordinator in Driver TABLE in database

1. getDriverArray()

returns all driver in an array from luke webservice

**PolygonODAO.cs**

6 methods

1. PolygonODAO()

Initialise a new ArrayList of PolygonO

1. addPolygon()

add new lat and lng coordinates into current PolygonO that has the same postal code id parameter. If none of the PolygonO has the same postal code id parameter, create new PolygonO.

1. getPolygons()

return an ArrayList of polygonO

1. getPolygonById()

return PolygonO based on polygon’s id given

1. getLatCenterOfPolygon()

return an average Latitude (double) of all latitude in polygon

Calculation: Sum of all Latitude in polygon / total count of latitude in polygon

1. getLngCenterOfPolygon()

return an average Longitude (double) of all longitude in polygon

Calculation: Sum of all Longitude in polygon / total count of longitude in polygon

**UserDAO.cs**

DATABASE UPDATES SHOULD BE EDITED IN THIS FILE

4 methods

1. UserDAO()

Random user data initialised into UserDAO. Finalised database should be current administrative user account in xdel’s database.

1. checkUser()

return User if username and password matches User in UserDAO. This code needs to be edited to retrieve user data if username and password matches a specific user in User TABLE in database

1. retrieveUsers()

returns an ArrayList of User (User Class) from UserDAO. This code needs to be edited to retrieve all users from User TABLE in database

1. updateUser()

edit the user’s role and postal code rights based on the given role and rights. This code needs to be edited to update user’s role and rights to the new give role and rights in User TABLE in database

**Model**

**Cluster.cs**

Parameters are as follows:

1. string id – [identification of cluster] – (example: “C1”)
2. string name – [full name of cluster] – (example: “Central 1”)
3. ArrayList postals – [postal codes that are within cluster]
4. string color – [color code of cluster] – (example: “#FFFFFF”)

Methods (basic getter and setter methods will not be mentioned):

1. containPostal()

Returns if cluster contains postal

**ClusterDetails.cs**

Parameters are as follows:

1. string id – [identification of cluster] – (example: “C1”)
2. double lat – [Latitude]
3. double lng – [Longitude]
4. int newJob – [number of new jobs]
5. int delJob – [number of dip jobs]
6. int puJob – [number of pip jobs]

Methods (basic getter and setter methods will not be mentioned):

Nil

**Driver.cs**

Parameters are as follows:

1. string name – [name of driver] – (example: “Gary”)
2. string coordinator – [name of coordinator] – (username of user)

Methods (basic getter and setter methods will not be mentioned):

Nil

**PolygonO.cs**

Parameters are as follows:

1. int id – [identification of polygon] – (example: 83)
2. ArrayList lat – [ArrayList of all Latitude]
3. ArrayList lng – [ArrayList of all Longitude]

Methods (basic getter and setter methods will not be mentioned):

1. addCoordinates()

add new latitude and longitude into arraylist of lat and arraylist of lng respectively

1. getCoordinates()

return a string concatenation of latitude and longitude

example: “[1.124,103.141],[1.125,103.145] ,[1.123,103.147]”

**PostalJob.cs**

Parameters are as follows:

1. string zone – [id of cluster] – (example: “C1”)
2. int district – [if of postal code] – (example: 6)
3. int newJob – [number of new jobs]
4. int delJob – [number of dip jobs]
5. int puJob – [number of pip jobs]

Methods (basic getter and setter methods will not be mentioned):

Nil

**User.cs**

Parameters are as follows:

1. string username – [username of user] – (example: “peter”)
2. string password – [password of user] – (example: “ilovexdel123”)
3. string role – [role of user] – (example: “Manager” / “admin”)
4. ArrayList postalcoderights – [ArrayList of postal codes that user have rights in]
5. ArrayList subodinates – [ArrayList of users that current user is in charged of]

Methods (basic getter and setter methods will not be mentioned):

Nil

**Libraries and Plugins**

**Important CSS files to note**

ShowMap.CSS

a) contains the CSS for table style in all pages

b) contains the style for toggle button for Traffic condition and Routes

c) contains style for w3.theme.orange to set color to orange

**W3.css**

Is the template for the webpage

*<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">*

CSS for the icons in the navbar

**Important JS file to note**

Search Bar.js

Is the JS code for the dynamic searchbar

**For The Leaflet Map**

Leaflet ->http://leafletjs.com/

Leaflet Routing Machine -> http://www.liedman.net/leaflet-routing-machine/

**When first open project**

Drag and drop leaflet-routing-machine.js & leaflet-routing-machine.min.js into Scripts/routing machine/leaflet-routing-machine-3.2.5/dist

**Singapore Postal Code Boundary**

**Singapore Postal Code Boundary Details**

Attached is the link of the all the markers plotted on Google Maps to draw out the postal code boundaries.

<https://drive.google.com/open?id=1dKljyFEYjiv0wN-gJ8IjkpB_xwA&usp=sharing>

*There are 6 tabs in total. Postal 1, 2, 3, 4, 5-42 & 43-83*

1. Postal 1/2/3/4

Postal 1/2/3/4 contains all the coordinates and lines required to mark out postal code boundary 1/2/3/4.

2. Postal 5-42, Postal 43-83

Postal 5-42 contains all the coordinates and lines required to mark out postal code boundaries 5 to 42. Upon clicking on the postal 5-42 tab. You will realised that the names of the markers are named in a logical and systematic manner. For example, Point 1 (5) represents the first point in postal code boundary 5. So Point 10 (6) would represent the tenth point in postal code boundary 6. The naming convention is the same for lines.

**Take Note**

Each and every point’s coordinates was manually copied onto a csv file. The csv file can be found in the project directory /img/Postal Code Boundary Coordinates.csv. Therefore, any changes or markers changed on the map has to be manually reflected into the csv file through copy and pasting.