# **Use case #1:** UC1\_User\_Starts\_app

## Requirements:

When the application is launched the user should be greeted with the MainView with three labels: **staff list, unit timetable and (NTH) activity heat maps.**

## Overview:

The Human Resource Information System will consist of two main components: staff lists and unit timetable display. These may be enhanced by the addition of another component: activity heat maps.

## Preconditions:

The application is being executed from the SOE of the school (Apple, windows 7 and windows 8).

## Scenario

|  |  |
| --- | --- |
| Action | Software Reaction |
| 1. User starts up. | 1. Display the MainView of the app. 2. The main view should have two labels:    1. Staff list    2. Unit timetable    3. (NTH) Activity heat maps |
| 1. User selects ‘staff list’ | **Use case #8: UC8\_User\_views\_StaffList** |
| 1. User selects Unit timetable | **Use case #23: UC23\_User\_views\_UnitList** |
| 1. (NTH) User selects Activity heat maps | **Use case #28: UC28\_User\_generates\_HeatMap (NTH)** |

## Scenario Notes:

Have not included any connection to database or retrieval of data on start-up of the application this is because at this point we don’t know what the user will choose. The query we send the database will be dependent on what the user selects and getting all the data records from database seems pointless and unnecessary.

Action 6 shouldn’t be added until at least iteration 2-3 (get the barebones working first).

## Post Conditions:

MainView should be displayed with all three labels (two if on first iteration) and simply waiting for input from the user (a click).

## Required Views: (GUIs)

MainView

## Exceptions:

App not opened in the schools SOE

## Use Cases utilised:

**Use case #8:** UC8\_User\_views\_StaffList

**Use case #28:** UC28\_User\_generates\_HeatMap (NTH)

# Use case #8: UC8\_User\_views\_StaffList

## Requirements:

The user shall be able to view an interactive list of staff employed by the School

## Overview:

The user shall be able to view an interactive list of staff employed by the School. The list will be accessed by selecting a tab labelled ‘Staff’, and should be visible upon application start up.

## Preconditions:

Database exists and contains data.

## Scenario

|  |  |
| --- | --- |
| Action | Software Reaction |
| 1. User selects tab labelled ‘Staff’ | 1. Query the MySQL database for all Staff. 2. Store the returned information in a data structure. 3. Open StaffTableView 4. Sort the data structure in alphabetical order using ‘family name’ followed by given name. 5. Display on StaffTableView with the format Family name, given and title. 6. Each item in list is ‘clickable’. 7. Scale font based on StaffTableView size. 8. Display category labelled, academic, technical, administrative and casual should be displayed. It should filter staff based on which category. 9. (NTH) Display a search for staff label. |
| 1. User clicks on a category label | 1. Create a set for all staff that are a member of Category X and display only those. 2. Refresh StaffTableView |
| 1. (NTH) User clicks on search label | 1. Opens a text search field at the top of the StaffTableView 2. Should remove names as the user types in a name by excluding list items that do not contain characters entered by the user. 3. For convenience when comparing text from the user and the list of names the search should ignore all upper and lower case. |
| 1. User clicks on list element (staff name) | **Use case #16: UC16\_User\_selects\_StaffDetails** |

## Scenario Notes:

Action 1 is the precondition for all the others. I think HashSet<T> would be the best approach since we only want to send one query to the database. If stored inside a set we can use set mathematics to query and the sets rather than going back and forth on the database.

## Post Conditions:

StaffTableView displays the set returned from the database or the subset of the set if a category tab was selected.

## Required Views: (GUIs)

StaffTableView.

## Exceptions:

Possible issue with database connectivity when making the query before Tabel1 is displayed. If this is the case, we should inform the user of the issue and display a dialog box. This should trigger a timer for 20 seconds and attempt to connect again, or allow the user to press cancel to shut down the application.

## Use Cases utilised:

**Use case #16: UC16\_User\_selects\_StaffDetails**

# **Use case #16: UC16\_User\_selects\_StaffDetails**

## Requirements:

When the users’ selects a name in the list the system will show more details about the staff member (referred to as the Staff Details view), which should include: Name; Campus; Phone Number; Room Location; Email Address; Photo; Consultation hours; Table of units he or she is involved with in the current semester

## Overview:

When the user selects a name in the list the system will show more details about the staff member (referred to as the Staff Details view), which should include:

* Name
* Campus
* Phone Number
* Room Location
* Email Address
* Photo
* Consultation hours
* Table of units he or she is involved with in the current semester

## Preconditions:

UC1\_User\_Starts\_app: user has started the application.

UC8\_User\_views\_StaffList: user has selected to view the staff list.

## Scenario

|  |  |
| --- | --- |
| Action | Software Reaction |
| 1. User has clicked element from staff list | 1. Open StaffDetailsView 2. Display staff information in format:    1. Name    2. Campus    3. Phone number    4. Room location    5. Email address    6. Photo    7. Consultation hours    8. Table of current semester units involved 3. (NTH) Show the current availability of the staff member based on their time table and the current time.    1. Display appropriate label on staff’s current activity:       1. Teaching       2. Consolation       3. Lecturing    2. Display associated information with current label:       1. Details of unit the staff member is teaching, lecturing.       2. Which room the staff member is teaching, lecturing or his/her consolation room. 4. Availability of staff member is clickable. |
| 1. (NTH) User clicks on staff’s availability status | **Use case #19: (NTH) UC19\_User\_shows\_ActivityGrid** |

## Scenario Notes:

Need to consider how StaffDetailsView will be ‘refreshed’ so that the availability of the staff member is fairly accurate. We can consider doing it dynamically using current time of display and the expected finish time of the activity allotted time to calculate when the view should be refreshed.

Access time = 1pm

Staff member activity ends at 1:30pm

Activity ends – access time = 30.

So the view should update (if left open) in 30 minutes.

## Post Conditions:

StaffDetailsView has the selected staff member’s details displayed and waiting for further instructions.

## Required Views: (GUIs)

StaffDetailsView

## Exceptions:

Semantic concerns: This information displayed in this view could be misleading if the information on the database isn’t up to date.

## Use Cases utilised:

**Use case #19: (NTH) UC19\_User\_shows\_ActivityGrid**

# **Use case #19: (NTH) UC19\_User\_shows\_ActivityGrid**

## Requirements:

(NTH) It would enhance the Staff Details view if the staff member’s activity (classes and consultation times) across a week could be displayed in a colour coded grid.

## Overview:

It would enhance the Staff Details view if the staff member’s activity (classes and consultation times) across a week could be displayed in a colour-coded grid. This grid should be toggled (displayed or hidden) via a button on the Staff Detail view. The grid should have days of the week (Monday through Friday) as columns and hours of the day (9am until 4pm) as rows, with each cell’s colour indicating the kind of activity at that time, but no other details shown. Free time should be shown in white, while teaching and consultation times should be shaded in distinct colours that are distinguishable by those with common forms of colour blindness.

## Preconditions:

(NTH) User clicks on staff’s availability status from **Use case #16: UC16\_User\_selects\_StaffDetails**

## Scenario

|  |  |
| --- | --- |
| Action | Software Reaction |
| 1. User clicks on staff’s availability status | 1. Open ActivityGridView 2. A toggle button to hide or show display should be displayed. 3. Display toggle button should set to ‘show’ by default:    1. This may be achieved by using a flag Boolean variable called toggle which is set to true. 4. The ActivityGridView should display a staff members schedule for a whole week in a coloured grid format. |
| 1. User clicks toggle | 1. If toggle flag is true, set to false and close ActivityGridView else set toggle flag true and open ActivittyGridView. 2. The ActivityGridView should be formatted as such:    1. columns should display the days (Monday to Friday).    2. rows should display hour blocks (from 9am to 4pm).    3. Each (row, column) should contain activities.    4. Each (block) should be coloured appropriately.       1. Free time should be white.       2. Teaching and consultation times should be distinctly different and colour blind friendly. |

## Scenario Notes:

Don’t use different colours, use the same colour in different shades. This is more colour blind friendly.

This grid would be very easy to create using a 2d matrix.

## Post Conditions:

The StaffDetailsView is currently displayed and ActivityGridView is either displayed or not displayed.

## Required Views: (GUIs)

ActivityGridView and StaffDetailsView

## Exceptions:

None.

## Use Cases utilised:

None.

# **Use case #23: UC23\_User\_views\_UnitList**

## Requirements:

The system should be able to generate a list of the units under the control of the School, ordered alphanumerically.

## Overview:

## The system should be able to generate a list of the units under the control of the School, ordered alphanumerically. Selecting a unit from this list should bring up a list of classes for that unit, ordered chronologically.

## Preconditions:

* Must have connection to database
* Database/table needs to exists and is populated
* User has selected Unit timetable label from MainView

## Scenario

|  |  |
| --- | --- |
| Action | Software Reaction |
| 1. User selects Unit timetable | 1. Query database and return a set of all units available at school 2. Store results in data structure 3. Sort stored units in alphanumerical order. 4. Display UnitListView 5. Display list of units in list format. 6. Each list element on view is clickable. |
| 1. User clicks on a unit in list. (NTH) | **Use case #24: UC24\_User\_selects\_Unit** |

## Scenario Notes:

Possible candidate for quicksort algorithm.

## Post Conditions:

A list of alphanumerically ordered units are displayed in the user interface and are clickable by the user.

## Required Views: (GUIs)

UnitListView is used for viewing the ordered list of units.

## Exceptions:

Database cannot be connected to.

Database is corrupt or no data.

## Use Cases utilised:

**Use case #24: UC24\_User\_selects\_Unit**

# **Use case #24: UC24\_User\_selects\_Unit**

## Requirements:

Selecting a unit from this list should bring up a list of classes for that unit, ordered chronologically.

## Overview:

## This view should be tabular, showing the following information about the selected unit in each column:

* day
* start and end time in 24-hour format, such as ‘12:00–14:00’ or ‘12:00–13:50’
* type (Lecture, Tutorial, Practical, Workshop)
* room location
* campus
* staff member

## Preconditions:

* Must be viewing the UnitListView
* Must click a unit in the UnitListView
* UnitListView must have entires.

## Scenario

|  |  |
| --- | --- |
| Action | Software Reaction |
| 1. User Clicks on Unit | 1. Opens UnitDetailsView 2. Information should be displayed in tabular form and display the following.    1. Day    2. Start-Finish    3. Type (Lecture, tutorial etc.)    4. Room Location    5. Campus    6. Staff Member |
| 2. User clicks staff member name. | **Use case #16: UC16\_User\_selects\_StaffDetails** |
| 1. User clicks Campus filter | 1. Display dropdown list:    1. Hobart    2. Launceston |
| 1. User selects Campus from filter 2. User clicks on ClashMap | 1. Refresh UnitDetailsView to only show information about selected campus. 2. **Use case #: UC37\_User\_generates\_ClashMap (NTH)** |

## Scenario Notes:

N/A

## Post Conditions:

UnitDetailsView is refreshed showing only times related to selected campus.

## Required Views: (GUIs)

UnitDetailsView

## Exceptions:

N/A

## Use Cases utilised:

**Use case #16: UC16\_User\_selects\_StaffDetails**

# **Use case #28: UC28\_User\_generates\_HeatMap (NTH)**

## Requirements:

## It would enhance the application’s utility if it could generate ‘heat maps’ of activity across the week.

## Overview:

It would enhance the application’s utility if it could generate ‘heat maps’ of activity across the week. It should be possible to generate two different heat maps, one for unit classes and another for consultation times. It must be possible to generate different heat maps for all ICT Discipline classes, and those only occurring in Hobart or Launceston. All four heat maps shall be accessed via a menu or buttons on a dedicated tab.

## Preconditions:

* User has opened app
* User has clicked on HeatMap button from MainView

## Scenario

|  |  |
| --- | --- |
| Action | Software Reaction |
| 1. User click HeatMap button | 1. Open and display HeatMapView 2. Dialog is displayed with options:    1. Location: Launceston, Hobart    2. A small selection of colours. |
| 1. User selects Campus from filter (location) | 1. Display radio box options    1. Classes Times    2. Consultations Times |
| 1. User clicks on classes/consultation times | 1. Opens class heat map    1. Create 2d grid display(format):       1. Hours/days and contains events    2. Shade grid based on number of events in hour/day (format):       1. White for none       2. Light shade for intermediate values       3. Solid for many |
| 1. User selects colour (NTH) | 1. Colour attribute is set for grid base colour 2. Refresh grid with new colour |

## Scenario Notes:

Doesn’t matter what colour user selects, heatmap will shade so that it is colour blind friendly ☺

## Post Conditions:

A heatmap is displayed for the user’s pleasure

## Required Views: (GUIs)

HeatMapView

## Exceptions:

## Use Cases utilised:

# **Use case #: UC37\_User\_generates\_ClashMap (NTH)**

## Requirements:

It would further enhance the application’s utility if it could generate a variant of the heat map that shows clashes between a unit’s classes and the consultation

## Overview:

It would further enhance the application’s utility if it could generate a variant of the heat map that shows clashes between a unit’s classes and the consultation time of staff involved in teaching that unit. This would be accessed via a button on the Unit Timetable view, and use data for the currently displayed classes and staff who teach them (that is, both campuses, Hobart only or Launceston only). The cells in the grid should use the following colour scheme: white if no class or staff consultation occurs on that hour and day; bright green if that time contains either a class or consultation, but not both; and red if it contains both consultation and a class. Cells representing clashes should also include the text ‘clash’.

## Preconditions:

UnitTimeTableView is loaded and populated with data

## Scenario

|  |  |
| --- | --- |
| Action | Software Reaction |
| 1. User clicks time table clashes button | 1. Open ClashMapView 2. Based on which filter is currently selected (showing only Hobart or Launceston) selected in UnitTimetableView 3. Collect all the unit’s classes and the consultation times of the teacher. 4. Generate 2d Grid where columns represent days and rows represent hour slots.    1. Each cell is coloured according to the following:       * White – Not Consultation or class       * Green – Consultation or Class but not both       * Red - consultation & class at the same time       * **Clash text is also displayed in red cells.** |

## Scenario Notes:

Clashmap filters (both, Hobart only or Launceston only) are based on the UnitTimeTableView’s filters to avoid the user having to repeatedly specify the data they’re interested in.

Clash map button should be disabled (or greyed out) when pre-conditions on UnitTimetableView are not met.

The clash map can be generated using a trace matrix with values of 1 entered in on the correct row and column. If the intersection of the column and row is only 1 then no clash has occurred. Anything greater than 1 would indicate a clash.

## Post Conditions:

**ClashMap is generated and displayed.**

## Required Views: (GUIs)

ClashMapView

## Exceptions:

## Use Cases utilised: