# Online examination file submission system
# PROJECT TECHNICAL DOCUMENTATION

10/26/2022

*Part 1*

# Project

# Project

## Code

### Initializing Online examination file submission system

```
gnUserID          is int
gsSession              is string
gsUser                 is string
gsUserType             is string
gnTimer                is int
gConnection            is Connection

// Free Email smtp test server powered by WP Oven
// https://www.wpoven.com/tools/free-smtp-server-for-testing
CONSTANT
     SERVERADDRESS = "smtp.freesmtpservers.com"
     SERVEREMAIL = "UNISA@test.com"
     SERVERPASSWORD = ""
     SERVERPORT = "25"
     SERVERTESTEMAIL = "65614968@mylife.unisa.ac.za"
END

CONSTANT
     UNISAAdminNumber      = 1000000
     UNISAAdminPassword         = "kJh6HJh56df"
     UNISAAdminUser             = "UNISAAdmin"

     ExamDeptNumber             = 9999999
     ExamDeptPassword      = "acd25AFG"
     ExamDeptUser          = "ExamDept"
END

gConnection..Provider     = hAccessHFClientServer
gConnection..User         = "UNISAAdmin"
gConnection..Password     = "kJh6HJh56df"
gConnection..Server            = "localhost"
gConnection..Database     = "ICT3715_DB"
gConnection..CryptMethod       = hEncryptionNO

// Open the connection
HOpenConnection(gConnection)

// Assign the connection to all data files
HChangeConnection("*", gConnection)
```

### Initializing the project after connection to the site of Online examination file submission system

```
HCreationIfNotFound("*")
```

# Project

## Code statistics

| | Lines [1] | % comm. [1] | Lin./proc. [1] |
|---|---|---|---|
| AutomaticServerProcedures | 86 | 23 | 43 |
| ServerProcedures | 550 | 29 | 42 |
| PAGE_Admin | 79 | 37 | 3 |
| PAGE_ExamDept_UploadExamSchedule | 59 | 22 | 4 |
| PAGE_Exam_Dashboard | 146 | 12 | 5 |
| PAGE_Exam_Department_Login | 92 | 12 | 7 |
| PAGE_Iframe_PDF | 14 | 0 | 14 |
| PAGE_InfoWait | 31 | 0 | 15 |
| PAGE_Lecturer_Dashboard | 144 | 18 | 7 |
| PAGE_Lecturer_Exam_PDFViewer | 62 | 21 | 5 |
| PAGE_Lecturer_Exam_Viewer | 96 | 14 | 8 |
| PAGE_Lecturer_Login | 100 | 12 | 7 |
| PAGE_Lecturer_SummaryReport | 162 | 12 | 8 |
| PAGE_Login | 165 | 15 | 10 |
| PAGE_StatisticalReport | 142 | 15 | 7 |
| PAGE_Student_Dashboard | 329 | 19 | 10 |
| PAGE_Student_Login | 102 | 12 | 7 |
| PAGE_WeeklyReport | 173 | 13 | 8 |
| PAGE_YesNoDialog | 21 | 0 | 5 |
| PAGE_index | 26 | 23 | 4 |
| IPAGE_DailyReport | 99 | 11 | 16 |
| IPAGE_Lecturer_SummaryReport | 107 | 16 | 21 |
| IPAGE_StatisticalReport | 87 | 22 | 29 |
| IPAGE_WeeklyReport | 116 | 16 | 19 |
| PAGETPL_AWP | 15 | 0 | 0 |
| PAGETPL_Session | 62 | 3 | 10 |
| Online examination file submission system | 50 | 8 | 25 |
| | **2781** | **20** | 9 |

[1] Lines : Total number of lines of code.
% comm. : Percentage of comments in code.
Lin./proc. : Number of lines of code per process.

*Part 2*

# Page

# PAGE_Login

## Code

### Global declarations of PAGE_Login (server)

```
PROCEDURE MyPage()
gbOnUser is boolean
```

# PAGE_Login

## Control code

### Click on IMG_Logo (server) (PAGETPL_Session template)

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    IF YesNo("You are currently logged in. Returning to Home page will Log you out. Continue?") = Yes THEN
        // End session and logout
        gnUserID          = 0
        gsSession             = ""
        gsUser                = ""
        gsUserType            = ""

        HCloseConnection(gConnection)
        PageDisplay(PAGE_Login)
    END
END
```

### Initializing of Link_Logout (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Initializing of Link_Logout (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Click on Link_Logout (onclick browser event)

```
//Run the process defined in the template
ExecuteAncestor
```

### Click on Link_Logout (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Click on Link_Logout (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Return from AJAX process after clicking on Link_Logout (browser)

```
//Run the process defined in the template
ExecuteAncestor
```

### Initializing of STC_WelcomeUser (server) (PAGETPL_Session template)

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    STC_WelcomeUser.Caption = "Welcome " + gsUser
ELSE
    STC_WelcomeUser.Caption = ""
END
```

### Initializing of MENU_Nav (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Initializing of MENU_Nav (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (server) (PAGETPL_Session template)

```
IF gsUser = ExamDeptUser THEN
    PageDisplay(PAGE_Exam_Dashboard)
END
IF gsUserType = "Lecturer" THEN
     PageDisplay(PAGE_Lecturer_Dashboard)
END
IF gsUserType = "Student" THEN
    PageDisplay(PAGE_Student_Dashboard)
END
```

### Initializing of BTN_ExamDeptLogin (server)

```
// Version 1
// Description
// Button that triggers a server action
```

### Initializing of BTN_LecturerLogin (server)

```
// Version 1
// Description
// Button that triggers a server action
```

### Initializing of BTN_StudentLogin (server)

```
// Version 1
// Description
// Button that triggers a server action
```

### Click on BTN_CANCEL (onclick browser event) ( CELL_NoName1 )

```
history.back();
```

### Click on BTN_OK (onclick browser event) ( CELL_NoName1 )

```
IF EDT_USERNUMBER = "" THEN
    ReturnToCapture(EDT_USERNUMBER)
END
IF EDT_PASSWORD = "" THEN
    ReturnToCapture(EDT_PASSWORD)
END
```

### Click on BTN_OK ( CELL_NoName1 ) (server)

```
gbOnUser = Connection()

IF gbOnUser = False THEN
    EDT_PASSWORD                = ""
    STC_Error                       = "Invalid user or password"
    STC_Error..Visible              = True
ELSE
    SWITCH gsUserType
        CASE "UNISAAdmin"
            PageDisplay(PAGE_Admin)
        CASE "ExamDept"
            PageDisplay(PAGE_Exam_Dashboard)
        CASE "Staff"
            PageDisplay(PAGE_Lecturer_Dashboard)
        CASE "Student"
            PageDisplay(PAGE_Student_Dashboard)
    END
END
```

### Add a token in EDT_PASSWORD ( CELL_NoName1 ) (browser)

```
PROCEDURE AjoutJeton (MonJeton is a Token)

//RENVOYER Faux pour interdire l'ajout du jeton
RESULT True
```

### Click on a token of EDT_PASSWORD ( CELL_NoName1 ) (browser)

```
PROCEDURE ClicJeton (MonJeton is a Token)
```

### Delete a token in EDT_PASSWORD ( CELL_NoName1 ) (browser)

```
PROCEDURE SuppressionJeton (MonJeton is a Token)

//RENVOYER Faux pour interdire la suppression du jeton
RESULT True
```

### Add a token in EDT_USERNUMBER ( CELL_NoName1 ) (browser)

```
PROCEDURE AjoutJeton (MonJeton is a Token)

//RENVOYER Faux pour interdire l'ajout du jeton
RESULT True
```

### Click on a token of EDT_USERNUMBER ( CELL_NoName1 ) (browser)

```
PROCEDURE ClicJeton (MonJeton is a Token)
```

**Delete a token in EDT_USERNUMBER ( CELL_NoName1 ) (browser)**

```
PROCEDURE SuppressionJeton (MonJeton is a Token)

//RENVOYER Faux pour interdire la suppression du jeton
RESULT True
```

# PAGE_Login

## Procedures

**Local procedure Connection (server)**

```
PROCEDURE Connection()
bOnUser is boolean = False

// Finds the user
HReadSeekFirst(StaffInfo,StaffNumber,EDT_USERNUMBER,hIdentical)
IF HFound(StaffInfo) THEN

    // Checks the password
    IF StaffInfo.Password = EDT_PASSWORD THEN

        // Generates a session
        gsUserType        = "Staff"
        gnUserID      = StaffInfo.StaffNumber
        gsUser            = StaffInfo.Name

        GenerateSession()
        bOnUser = True
    END
ELSE
    HReadSeekFirst(StudentInfo,StudentNumber,EDT_USERNUMBER,hIdentical)
    IF HFound(StudentInfo) THEN

        // Checks the password
        IF StudentInfo.StudentPassword = EDT_PASSWORD THEN

            // Generates a session
            gsUserType        = "Student"
            gnUserID      = StudentInfo.StudentNumber
            gsUser            = StudentInfo.StudentName

            GenerateSession()
            bOnUser = True
        END
    ELSE
        IF EDT_USERNUMBER = UNISAAdminNumber _AND_ EDT_PASSWORD = UNISAAdminPassword THEN
            // Generates a session
            gsUserType        = UNISAAdminUser
            gnUserID      = UNISAAdminNumber
            gsUser            = UNISAAdminUser
```

```
            GenerateSession()
            bOnUser = True
        ELSE
            IF EDT_USERNUMBER = ExamDeptNumber _AND_ EDT_PASSWORD = ExamDeptPassword THEN
                // Generates a session
                gsUserType        = ExamDeptUser
                gnUserID      = ExamDeptNumber
                gsUser            = ExamDeptUser

                GenerateSession()
                bOnUser = True
            END
        END
    END
END

RETURN bOnUser
```

**Local procedure GenerateSession (server)**

```
PROCEDURE GenerateSession()
// Initialize new connection
New_connection is Connection
// Connection parameters
New_connection..Provider       = hAccessHFClientServer
New_connection..User           = gsUser
New_connection..Password       = EDT_PASSWORD
New_connection..Server             = "localhost"
New_connection..Database       = "ICT3715_DB"
New_connection..CryptMethod        = hEncryptionNO

// Generates a session identifier
dtdhDateTimeCurrent is DateTime = Today() + TimeSys()
gsSession = dtdhDateTimeCurrent + TAB + gnUserID + "-" + gsUser

SWITCH gsUserType
    CASE "Student"
        HChangeConnection("ExamOutput,ExamSetup,StudentModule",New_connection)
    CASE "Staff"
        HChangeConnection("ExamSetup,ModuleLeader",New_connection)
    CASE "ExamDept"
        HChangeConnection("ModuleLeader,ExamSetup,ModuleInfo,StaffInfo,StudentModule",New_connection)
    CASE "UserLogin"
        HChangeConnection("StaffInfo,StudentInfo",New_connection)
    OTHER CASE
        HChangeConnection("*",New_connection)
END

gConnection = New_connection

// Backup
CookieWrite("session", gsSession, 1)
```

# PAGE_index

## Code

**Global declarations of PAGE_index (server)**

```
PROCEDURE MyPage()
```

# PAGE_index

## Control code

**Initializing of ZONE_Header (server)**

```
//Run the process defined in the template
ExecuteAncestor
```

**Initializing of ZONE_Header (server)**

```
//Run the process defined in the template
ExecuteAncestor
```

**Click on ZONE_Header (onclick browser event)**

```
//Run the process defined in the template
ExecuteAncestor
```

**Move mouse over ZONE_Header (onmousemove browser event)**

```
//Run the process defined in the template
ExecuteAncestor
```

**Delayed loading of a plane of ZONE_Header (server)**

```
//Run the process defined in the template
ExecuteAncestor
```

**Delayed loading of a plane of ZONE_Header (browser)**

```
//Run the process defined in the template
ExecuteAncestor
```

**Initializing of BTN_ServerAction (server)**

```
// Version 1
// Description
```

```
// Button that triggers a server action
```

**Click on BTN_ServerAction (onclick browser event)**

```
DynamicSiteDisplay("","","",NewBrowser)
```

# PAGE_Admin

## Code

**Global declarations of PAGE_Admin (server)**

```
PROCEDURE MyPage()
```

# PAGE_Admin

## Control code

**Click on IMG_Logo (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    IF YesNo("You are currently logged in. Returning to Home page will Log you out. Continue?") = Yes THEN
        // End session and logout
        gnUserID          = 0
        gsSession             = ""
        gsUser                = ""
        gsUserType            = ""

        HCloseConnection(gConnection)
        PageDisplay(PAGE_Login)
    END
END
```

**Initializing of Link_Logout (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    Link_Logout..Visible = True
END
```

**Click on Link_Logout (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    // End session and logout
    gnUserID          = 0
    gsSession             = ""
    gsUser                = ""
    gsUserType            = ""

    HCloseConnection(gConnection)
    PageDisplay(PAGE_Login)
END
```

**Initializing of STC_WelcomeUser (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    STC_WelcomeUser.Caption = "Welcome " + gsUser
ELSE
    STC_WelcomeUser.Caption = ""
END
```

### Initializing of MENU_Nav (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Initializing of MENU_Nav (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (browser)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Return from AJAX process after selecting the ZONE_Menu.MENU_Nav.Home menu ( MENU_Nav ) (browser)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (browser)

```
//Run the process defined in the template
ExecuteAncestor
```

### Return from AJAX process after selecting the ZONE_Menu.MENU_Nav.Home menu ( MENU_Nav ) (browser)

```
//Run the process defined in the template
ExecuteAncestor
```

### Initializing of BTN_UploadExamOutput (server)

```
// Version 1
// Description
// Button that triggers a server action
```

### Click on BTN_UploadExamOutput (server)

```
PageDisplayDialog(PAGE_InfoWait,EDT_File_Location,"ExamOutput")
```

### Initializing of BTN_UploadExamSetup (server)

```
// Version 1
// Description
// Button that triggers a server action
```

### Click on BTN_UploadExamSetup (server)

```
PageDisplayDialog(PAGE_InfoWait,EDT_File_Location,"ExamSetup")
```

### Initializing of BTN_UploadModuleInfo (server)

```
// Version 1
// Description
// Button that triggers a server action
```

### Click on BTN_UploadModuleInfo (server)

```
PageDisplayDialog(PAGE_InfoWait,EDT_File_Location,"ModuleInfo")
```

### Initializing of BTN_UploadModuleLeader (server)

```
// Version 1
// Description
// Button that triggers a server action
```

### Click on BTN_UploadModuleLeader (server)

```
PageDisplayDialog(PAGE_InfoWait,EDT_File_Location,"ModuleLeader")
```

### Initializing of BTN_UploadStaffInfo (server)

```
// Version 1
// Description
// Button that triggers a server action
```

### Click on BTN_UploadStaffInfo (server)

```
PageDisplayDialog(PAGE_InfoWait,EDT_File_Location,"StaffInfo")
```

### Initializing of BTN_UploadStudentInfo (server)

```
// Version 1
// Description
// Button that triggers a server action
```

### Click on BTN_UploadStudentInfo (server)

```
PageDisplayDialog(PAGE_InfoWait,EDT_File_Location,"StudentInfo")
```

### Initializing of BTN_UploadStudentModule (server)

```
// Version 1
// Description
// Button that triggers a server action
```

### Click on BTN_UploadStudentModule (server)

```
PageDisplayDialog(PAGE_InfoWait,EDT_File_Location,"StudentModule")
```

### Initializing of EDT_File_Location (server)

```
// Version 1
// Description
// Edit control for plain single-line text
```

### Add a token in EDT_File_Location (browser)

```
PROCEDURE AjoutJeton (MonJeton is a Token)

//RENVOYER Faux pour interdire l'ajout du jeton
RESULT True
```

### Click on a token of EDT_File_Location (browser)

```
PROCEDURE ClicJeton (MonJeton is a Token)
```

### Delete a token in EDT_File_Location (browser)

```
PROCEDURE SuppressionJeton (MonJeton is a Token)

//RENVOYER Faux pour interdire la suppression du jeton
RESULT True
```

# PAGE_InfoWait

## Code

**Global declarations of PAGE_InfoWait (server)**

```
PROCEDURE MyPage(sPath, sUpload are strings)
```

# PAGE_InfoWait

## Control code

**Initializing of BTN_Ok ( CELL_Zone1 ) (server)**

```
// Version 1
// Description
// Button that triggers a server action
```

**Click on BTN_Ok ( CELL_Zone1 ) (server)**

```
SWITCH sUpload
    CASE "StaffInfo"
        UploadStaffInfo(sPath)
    CASE "StudentInfo"
        UploadStudentInfo(sPath)
    CASE "ModuleInfo"
        UploadModuleInfo(sPath)
    CASE "ExamSetup"
        UploadExamSetup(sPath)
    CASE "ModuleLeader"
        UploadModuleLeader(sPath)
    CASE "StudentModule"
        UploadStudentModule(sPath)
    CASE "ExamOutput"
        UploadExamOutput(sPath)
    CASE "ExamSchedule"
        UploadExamSchedule(sPath)
END
PageCloseDialog(True)
```

# PAGE_Exam_Dashboard

## Code

**Global declarations of PAGE_Exam_Dashboard (server)**

```
PROCEDURE MyPage()
gDate is Date
```

**Initializing of PAGE_Exam_Dashboard (server)**

```
BTN_DailyReport.State = Active
LoadDateValue()
```

# PAGE_Exam_Dashboard

## Control code

**Click on IMG_Logo (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    IF YesNo("You are currently logged in. Returning to Home page will Log you out. Continue?") = Yes THEN
        // End session and logout
        gnUserID          = 0
        gsSession             = ""
        gsUser                = ""
        gsUserType            = ""

        HCloseConnection(gConnection)
        PageDisplay(PAGE_Login)
    END
END
```

**Initializing of Link_Logout (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    Link_Logout..Visible = True
END
```

**Click on Link_Logout (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    // End session and logout
    gnUserID          = 0
    gsSession             = ""
    gsUser                = ""
```

```
    gsUserType            = ""

    HCloseConnection(gConnection)
    PageDisplay(PAGE_Login)
END
```

### Initializing of STC_WelcomeUser (server) (PAGETPL_Session template)

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    STC_WelcomeUser.Caption = "Welcome " + gsUser
ELSE
    STC_WelcomeUser.Caption = ""
END
```

### Initializing of MENU_Nav (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Initializing of MENU_Nav (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (browser)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Return from AJAX process after selecting the ZONE_Menu.MENU_Nav.Home menu ( MENU_Nav ) (browser)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (browser)

```
//Run the process defined in the template
ExecuteAncestor
```

### Return from AJAX process after selecting the ZONE_Menu.MENU_Nav.Home menu ( MENU_Nav ) (browser)

```
//Run the process defined in the template
ExecuteAncestor
```

### Click on BTN_DailyReport (server)

```
ChangeSourcePage(IPAGE_Reports,IPAGE_DailyReport)
```

### Click on BTN_LecturerSummaryReport (server)

```
ChangeSourcePage(IPAGE_Reports,IPAGE_Lecturer_SummaryReport)
```

### Click on BTN_StatisticalReport (server)

```
ChangeSourcePage(IPAGE_Reports,IPAGE_StatisticalReport)
```

### Click on BTN_WeeklyReport (server)

```
ChangeSourcePage(IPAGE_Reports,IPAGE_WeeklyReport)
```

### Click on BTN_NoName1 (onclick browser event)

```
PopupDisplay(POPUP_Combobox,popupFixed+popupDiscardable)
```

### Initializing of BTN_SetDay (server)

```
// Version 1
// Description
// Button that triggers a server action
```

### Click on BTN_SetDay (server)

```
LoadDateValue()
```

### Initializing of BTN_SetDay1 (server)

```
// Version 1
// Description
// Button that triggers a server action
```

### Click on BTN_SetDay1 (server)

```
IF COMBO_ModuleInfo.Select() <> 1 THEN
    QRY_StaffModules.ParamModuleCode = COMBO_ModuleInfo[COMBO_ModuleInfo.Value]
    HExecuteQuery(QRY_StaffModules)
    HReadFirst(QRY_StaffModules)
    IF HFound(QRY_StaffModules) THEN
        STC_LecturerName = QRY_StaffModules.LecturerName
        STC_LecturerEmail = QRY_StaffModules.LecturerEmail
        STC_ModuleCode = QRY_StaffModules.ModuleCode
        STC_ModuleDescription = QRY_StaffModules.Description
    END

    nStudents    is int
    // Expected Students
    QRY_DailyRpt_StudentsToWrite.ParamDateExam = gDate
```

```
    QRY_DailyRpt_StudentsToWrite.ParamModuleCode = COMBO_ModuleInfo[COMBO_ModuleInfo.Value]
    HExecuteQuery(QRY_DailyRpt_StudentsToWrite)
    FOR EACH QRY_DailyRpt_StudentsToWrite
        nStudents += 1
    END

    STC_StudentsToWriteModule.Value = "Expected number of students to write " + COMBO_ModuleInfo[COMBO_
    ModuleInfo.Value]
    STC_StudentsToWriteModuleValue = NumToString(nStudents)
    CELL_ModuleInfo.Visible = True

ELSE
    CELL_ModuleInfo.Visible = False
END
```

**Initializing of COMBO_ModuleInfo (server)**

```
// Version 1
// Description
// Combo box for selecting a continent

// Initialize Listbox value
ListSelectPlus(MySelf,0)
```

**Add a token in EDT_ChangeDay (browser)**

```
PROCEDURE AddToken (MyToken is Token)

//RETURN False to prevent from adding the token
RETURN True
```

**Click on a token of EDT_ChangeDay (browser)**

```
PROCEDURE ClickToken (MyToken is Token)
```

**Delete a token in EDT_ChangeDay (browser)**

```
PROCEDURE DeleteToken(MyToken is Token)

//RESULT False to prevent from deleting the token
RESULT True
```

**Click on BTN_Daily_Report (onclick browser event) ( POPUP_Combobox )**

```
// Functionality to implement as required
PopupClose()
```

**Click on BTN_Daily_Report ( POPUP_Combobox ) (server)**

```
PageDisplay(PAGE_Exam_Dashboard)
```

**Click on BTN_LecturerSummaryReport (onclick browser event) ( POPUP_Combobox )**

```
PopupClose()
```

**Click on BTN_LecturerSummaryReport ( POPUP_Combobox ) (server)**

```
PageDisplay(PAGE_Lecturer_SummaryReport)
```

**Click on BTN_StatisticalReport (onclick browser event) ( POPUP_Combobox )**

```
PopupClose()
```

**Click on BTN_StatisticalReport ( POPUP_Combobox ) (server)**

```
PageDisplay(PAGE_StatisticalReport)
```

**Click on BTN_WeeklyReport (onclick browser event) ( POPUP_Combobox )**

```
PopupClose()
```

**Click on BTN_WeeklyReport ( POPUP_Combobox ) (server)**

```
PageDisplay(PAGE_WeeklyReport)
```

# PAGE_Exam_Dashboard

## Procedures

**Local procedure LoadDateValue (server)**

```
PROCEDURE LoadDateValue()
gDate = EDT_ChangeDay
nStudents is int
nModules is int

// Hide Module information cell
CELL_ModuleInfo.Visible = False

// Clear Combo box and refill
COMBO_ModuleInfo.DeleteAll()
ListAdd(COMBO_ModuleInfo,"Select a module...")
ListSelectPlus(COMBO_ModuleInfo,0)

// Expected Students
QRY_DailyRpt_StudentsToWrite.ParamDateExam = gDate
HExecuteQuery(QRY_DailyRpt_StudentsToWrite)
FOR EACH QRY_DailyRpt_StudentsToWrite
    nStudents += 1
END

STC_StudentsToWriteValue = NumToString(nStudents)
```

```
// Modules to be written
QRY_DailyRpt_ModulesExpected.ParamDateExam = gDate
HExecuteQuery(QRY_DailyRpt_ModulesExpected)
FOR EACH QRY_DailyRpt_ModulesExpected
    nModules += 1

    // Fill combo box
    ListAdd(COMBO_ModuleInfo,QRY_DailyRpt_ModulesExpected.ModuleCode)
END

STC_ExpectedModulesValue = NumToString(nModules)

// Update Chart
arrTimes is array of DateTime = [gDate+070000000, gDate+080000000, gDate+090000000, ...
    gDate+100000000, gDate+110000000,gDate+120000000,gDate+130000000,gDate+140000000, ...
    gDate+150000000,gDate+160000000,gDate+170000000,gDate+180000000,gDate+190000000, ...
    gDate+200000000,gDate+210000000]

numOfSubmissions is int

grDeleteSeries(CHART_Uploads,1,grData)
FOR times = 1 TO arrTimes.Count()
    IF times = arrTimes.Count() THEN BREAK
    numOfSubmissions = 0
    QRY_DailyRpt_Chart.ParamUploadTimeMin = arrTimes[times]
    QRY_DailyRpt_Chart.ParamUploadTimeMax = arrTimes[times + 1]
    HExecuteQuery(QRY_DailyRpt_Chart)
    FOR EACH QRY_DailyRpt_Chart
        numOfSubmissions += 1
    END

    grAddData(CHART_Uploads,1,numOfSubmissions)
END
grDraw(CHART_Uploads)
```

# PAGE_Lecturer_Dashboard

## Code

**Global declarations of PAGE_Lecturer_Dashboard (server)**

```
PROCEDURE MyPage()
sADDModuleName is string
```

# PAGE_Lecturer_Dashboard

## Control code

**Click on IMG_Logo (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    IF YesNo("You are currently logged in. Returning to Home page will Log you out. Continue?") = Yes THEN
        // End session and logout
        gnUserID          = 0
        gsSession             = ""
        gsUser                = ""
        gsUserType            = ""

        HCloseConnection(gConnection)
        PageDisplay(PAGE_Login)
    END
END
```

**Initializing of Link_Logout (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    Link_Logout..Visible = True
END
```

**Click on Link_Logout (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    // End session and logout
    gnUserID          = 0
    gsSession             = ""
    gsUser                = ""
    gsUserType            = ""

    HCloseConnection(gConnection)
    PageDisplay(PAGE_Login)
END
```

**Initializing of STC_WelcomeUser (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    STC_WelcomeUser.Caption = "Welcome " + gsUser
ELSE
    STC_WelcomeUser.Caption = ""
END
```

### Initializing of MENU_Nav (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Initializing of MENU_Nav (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (browser)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Return from AJAX process after selecting the ZONE_Menu.MENU_Nav.Home menu ( MENU_Nav ) (browser)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (browser)

```
//Run the process defined in the template
ExecuteAncestor
```

### Return from AJAX process after selecting the ZONE_Menu.MENU_Nav.Home menu ( MENU_Nav ) (browser)

```
//Run the process defined in the template
ExecuteAncestor
```

### Initializing of LOOP_StaffModules (server)

```
sFileName is string

QRY_StaffModules.ParamStaffNumber = NumToString(gnUserID)
HExecuteQuery(QRY_StaffModules)
FOR EACH QRY_StaffModules

    sFileName = "\ExamFiles\" + QRY_StaffModules.ModuleCode + ".pdf"
    IF fFileExist(fDataDir() + sFileName) THEN
        sFileName = QRY_StaffModules.ModuleCode + ".pdf"
    ELSE
```

```
        sFileName                   = "None"
    END
    LooperAddLine(LOOP_StaffModules,QRY_StaffModules.ModuleCode,Right(QRY_StaffModules.DateExam, 2)+"-"+
    QRY_StaffModules.DateExam[[5 TO 6]]+"-"+Left(QRY_StaffModules.DateExam,4),sFileName)
END
```

### Initializing of BTN_AddModuleExam ( LOOP_StaffModules ) (server)

```
// Version 1
// Description
// Button that triggers a server action
```

### Click on BTN_AddModuleExam ( LOOP_StaffModules ) (server)

```
CELL_UploadExamPDF.Visible = True
sADDModuleName = ATT_Module.Value
```

### Click on BTN_SEND (onclick browser event) ( CELL_UploadExamPDF )

```
// Gray the buttons to avoid uploading new files during the upload
MySelf..Grayed        = True
UPL_Upload..Grayed    = True

// Hide the delete button for each file
FOR ALL ROW OF LOOP_Files
    ATT_Del = False
END

// Start the upload
UploadStart(UPL_Upload)
```

### Click on BTN_Delete (onclick browser event) ( LOOP_Files )

```
UploadDelete(UPL_Upload,LOOP_Files)
IF UPL_Upload..Count = 0 THEN
    LooperDeleteAll(LOOP_Files)
    STC_Drop_the_files_here..Visible = True
END
```

### Initializing of STC_Drop_the_files_here ( CELL_Upload ) (server)

```
MySelf..Y = 5
```

### Initializing of UPL_Upload ( CELL_UploadExamPDF ) (server)

```
// Version 1
// Description
// Upload files by simple DND
```

### Whenever modifying the list of files selected in UPL_Upload ( CELL_UploadExamPDF ) (browser)

```
nSize    is system int
sSize    is string

// If the control contains a file: starts the file upload
```

```
IF MySelf..Count = 1 THEN
    LooperDeleteAll(LOOP_Files)

    STC_Drop_the_files_here..Visible   = False
    nSize                              = UploadFileSize(MySelf, 1)
    sSize                              = LengthToString(nSize)
    LooperAddLine(LOOP_Files,MySelf[1],sSize,0,"",RGB(255,192,64))
END
```

**Progress of transfer of UPL_Upload ( CELL_UploadExamPDF ) (browser)**

```
sFile                is string    = MySelf[UploadCurrentFile(MySelf)]
// File currently uploaded
rGlobalProgress      is real      = UploadSizeSent(MySelf) / UploadSize(MySelf)
// Global upload
rFileProgress        is real      = UploadCurrentFileSizeSent(MySelf) / UploadCurrentFileSize(MySelf)

// Progress of progress bar (width of progress bar control set to 179px)
ATT_ProgBarValue[UploadCurrentFile(MySelf)]= Round(rFileProgress * 100,0) + " %"
ATT_ProgBarWidth[UploadCurrentFile(MySelf)]= rFileProgress * 100 * 179 / 100

// End of progress
IF Round(rFileProgress * 100) >= 100 THEN
    IF NOT StringEndsWith(sFile,".pdf",ccNormal) THEN
        ATT_ProgBarColor[UploadCurrentFile(MySelf)]= RGB(255, 0, 0)
        ATT_ProgBarValue[UploadCurrentFile(MySelf)]= "Failed"
    ELSE
        ATT_ProgBarColor[UploadCurrentFile(MySelf)]= RGB(76,175,80)
        ATT_ProgBarValue[UploadCurrentFile(MySelf)]= "Uploaded to server"
    END
END
```

**Receive files uploaded from UPL_Upload ( CELL_UploadExamPDF ) (server)**

```
// Insert the code for processing uploaded files
sFileName is string
// Copies the uploaded file into a specific directory
FOR i = 1 _TO_ MySelf..Count
    sFileName = StringFormat(MySelf[i].NameBrowserFile,ccUpCase)
    sFileName = StringFormat(Right(sFileName,4),ccLowCase)
    // Check if file is PDF format
     IF StringEndsWith(sFileName,".pdf",ccNormal) THEN
        // IF ex. ICT3715.pdf = ICT3715.pdf
        IF sFileName <> (sADDModuleName + ".pdf") THEN
            // Rename file
            sFileName = sADDModuleName + ".pdf"
        END
        WHEN EXCEPTION IN
            IF fFileExist(fDataDir() + fSep() + "ExamFiles" + fSep() + sFileName) THEN
                fDelete(fDataDir() + fSep() + "ExamFiles" + fSep() + sFileName)
            END
            UploadCopyFile(MySelf, fDataDir() + fSep() + "ExamFiles" , sFileName,i)
            ToastDisplay("Upload successful",toastShort,vaMiddle,haCenter)
        DO
            ToastDisplay("An error occured while uploading the document",toastShort,vaMiddle,haCenter)
            BREAK
        END
     END
END
// Reload looper
```

```
LooperDeleteAll(LOOP_StaffModules)
LooperDisplay(LOOP_StaffModules,taInit)

CELL_UploadExamPDF.Visible = False
```

**After reception of the files uploaded from UPL_Upload ( CELL_UploadExamPDF ) (browser)**

```
// Ungray the add and upload buttons
BTN_SEND..Grayed = False
MySelf..Grayed       = False
```

# PAGE_Student_Dashboard

## Code

**Global declarations of PAGE_Student_Dashboard (server)**

```
PROCEDURE MyPage(gbDownload is boolean = False)
sADDModuleName is string
sDownloadModCode is string,browser synchronized

gStartTime       is Time
gEndTime     is Time

gStartTime.Hour           = 9
gStartTime.Minute     = 0
gStartTime.Second     = 0
gStartTime.Millisecond    = 0
gEndTime.Hour             = 14
gEndTime.Minute           = 0
gEndTime.Second           = 0
gEndTime.Millisecond  = 0
```

# PAGE_Student_Dashboard

## Control code

**Click on IMG_Logo (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    IF YesNo("You are currently logged in. Returning to Home page will Log you out. Continue?") = Yes THEN
        // End session and logout
        gnUserID          = 0
        gsSession             = ""
        gsUser                = ""
        gsUserType            = ""

        HCloseConnection(gConnection)
        PageDisplay(PAGE_Login)
    END
END
```

**Initializing of Link_Logout (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    Link_Logout..Visible = True
END
```

**Click on Link_Logout (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    // End session and logout
    gnUserID          = 0
    gsSession             = ""
    gsUser                = ""
    gsUserType            = ""

    HCloseConnection(gConnection)
    PageDisplay(PAGE_Login)
END
```

### Initializing of STC_WelcomeUser (server) (PAGETPL_Session template)

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    STC_WelcomeUser.Caption = "Welcome " + gsUser
ELSE
    STC_WelcomeUser.Caption = ""
END
```

### Initializing of MENU_Nav (server) (PAGETPL_Session template)

```
MySelf.OPT_ExamView.Visible = False
MySelf.OPT_ExamScheduleUpload.Visible = False

IF gsUser = ExamDeptUser THEN
    MySelf.OPT_ExamScheduleUpload.Visible = True
END
IF gsUserType = "Lecturer" THEN
    MySelf.OPT_ExamView.Visible = True
END
```

### Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (server) (PAGETPL_Session template)

```
IF gsUser = ExamDeptUser THEN
    PageDisplay(PAGE_Exam_Dashboard)
END
IF gsUserType = "Lecturer" THEN
     PageDisplay(PAGE_Lecturer_Dashboard)
END
IF gsUserType = "Student" THEN
    PageDisplay(PAGE_Student_Dashboard)
END
```

### Initializing of BTN_SetDay (server)

```
// Version 1
// Description
// Button that triggers a server action
```

### Click on BTN_SetDay (server)

```
// Clear looper contents
LOOP_StudentExams.DeleteAll()
// Re initiate looper
LOOP_StudentExams.Display(taInit)
```

### Initializing of LOOP_StudentExams (server)

```
sFileName is string
nLineNumber is int = 0

// Load student exams
QRY_StudentExams.ParamStudentNumber = NumToString(gnUserID)
HExecuteQuery(QRY_StudentExams)
FOR EACH QRY_StudentExams
    nLineNumber += 1
    sFileName = "\ExamFiles\" + QRY_StudentExams.ModuleCode + ".pdf"
    IF fFileExist(fDataDir() + sFileName) THEN
        sFileName = QRY_StudentExams.ModuleCode + ".pdf"
    ELSE
        sFileName                 = "None"
    END
    LooperAddLine(LOOP_StudentExams,QRY_StudentExams.ModuleCode,Right(QRY_StudentExams.DateExam, 2)+"-"+
    QRY_StudentExams.DateExam[[5 TO 6]]+"-"+Left(QRY_StudentExams.DateExam,4),sFileName)

    IF QRY_StudentExams.DateExam = EDT_ChangeDay AND Now() >= gStartTime AND Now() < gEndTime THEN
        LOOP_StudentExams[nLineNumber].BTN_DownloadModuleExam.State  = Active
        // Check if user already initiated the download
        // Start time if not
        HReadSeekFirst(ExamOutput,StudentNumberModuleCode,[QRY_StudentExams.StudentNumber,QRY_StudentExams
        .ModuleCode])
        IF HFound(ExamOutput) = True THEN
            // Check if still in allocated time range
            ExamStartTime is Time = ExamOutput.StartTime.Time
            SubmitByTime is Time = ExamStartTime
            SubmitByTime.Hour += 2

            SubmitByTimeDate is DateTime = ExamOutput.StartTime
            SubmitByTimeDate.Hour += 2

            IF TimeDifference(TimeSys(),SubmitByTime) > 0 THEN
                LOOP_StudentExams[nLineNumber].BTN_UploadModuleExam.State    = Active
            ELSE
                ToastDisplay("Upload time has passed for " + ExamOutput.ModuleCode,toastShort,vaMiddle,
                haCenter)
                LOOP_StudentExams[nLineNumber].BTN_DownloadModuleExam.State  = Grayed
            END
        END
    END
END
```

**Initializing of BTN_DownloadModuleExam ( LOOP_StudentExams ) (server)**

```
// Version 1
// Description
// Button that triggers a server action
```

**Click on BTN_DownloadModuleExam ( LOOP_StudentExams ) (server)**

```
// Set module code for processing
sDownloadModCode = LOOP_StudentExams[LOOP_StudentExams.Select()].ATT_Module
PopupDisplay(POPUP_DownloadDialog,popupTopCenter)
```

**Initializing of BTN_UploadModuleExam ( LOOP_StudentExams ) (server)**

```
// Version 1
// Description
```

```
// Button that triggers a server action
```

### Click on BTN_UploadModuleExam ( LOOP_StudentExams ) (server)

```
LOOP_Files.DeleteAll()
CELL_UploadExamPDF.Visible     = True
sADDModuleName                 = ATT_Module.Value
```

### Click on BTN_SEND (onclick browser event) ( CELL_UploadExamPDF )

```
// Gray the buttons to avoid uploading new files during the upload
MySelf..Grayed        = True
UPL_Upload..Grayed    = True

// Hide the delete button for each file
FOR ALL ROW OF LOOP_Files
    ATT_Del = False
END

// Start the upload
UploadStart(UPL_Upload)
```

### Click on BTN_Delete (onclick browser event) ( LOOP_Files )

```
UploadDelete(UPL_Upload,LOOP_Files)
IF UPL_Upload..Count = 0 THEN
    LooperDeleteAll(LOOP_Files)
    STC_Drop_the_files_here..Visible = True
END
```

### Initializing of STC_Drop_the_files_here ( CELL_Upload ) (server)

```
MySelf..Y = 5
```

### Initializing of UPL_Upload ( CELL_UploadExamPDF ) (server)

```
// Version 1
// Description
// Upload files by simple DND
```

### Whenever modifying the list of files selected in UPL_Upload ( CELL_UploadExamPDF ) (browser)

```
nSize    is system int
sSize    is string

// If the control contains a file: starts the file upload
IF MySelf..Count = 1 THEN
    LooperDeleteAll(LOOP_Files)

    FOR i=1 _TO_ MySelf..Count
        STC_Drop_the_files_here..Visible   = False
        nSize                              = UploadFileSize(MySelf, i)
        sSize                              = LengthToString(nSize)
        LooperAddLine(LOOP_Files,MySelf[i],sSize,0,"",RGB(255,192,64))
    END
ELSE
```

```
    IF MySelf..Count > 1 THEN
        Error("Only one PDF file is allowed to be uploaded for this exam.")
    END
END
```

**Progress of transfer of UPL_Upload ( CELL_UploadExamPDF ) (browser)**

```
sFile                is string    = MySelf[UploadCurrentFile(MySelf)]
// File currently uploaded
rGlobalProgress      is real      = UploadSizeSent(MySelf) / UploadSize(MySelf)
// Global upload
rFileProgress        is real      = UploadCurrentFileSizeSent(MySelf) / UploadCurrentFileSize(MySelf)

// Progress of progress bar (width of progress bar control set to 179px)
ATT_ProgBarValue[UploadCurrentFile(MySelf)]= Round(rFileProgress * 100,0) + " %"
ATT_ProgBarWidth[UploadCurrentFile(MySelf)]= rFileProgress * 100 * 179 / 100

// End of progress
IF Round(rFileProgress * 100) >= 100 THEN
    IF NOT StringEndsWith(sFile,".pdf",ccNormal) THEN
        ATT_ProgBarColor[UploadCurrentFile(MySelf)]= RGB(255, 0, 0)
        ATT_ProgBarValue[UploadCurrentFile(MySelf)]= "Failed"
    ELSE
        ATT_ProgBarColor[UploadCurrentFile(MySelf)]= RGB(76,175,80)
        ATT_ProgBarValue[UploadCurrentFile(MySelf)]= "Uploaded to server"
    END
END
```

**Receive files uploaded from UPL_Upload ( CELL_UploadExamPDF ) (server)**

```
// Insert the code for processing uploaded files
sFileName is string

// Copies the uploaded file into a specific directory
FOR i = 1 _TO_ MySelf..Count
    QRY_StudentExams.ParamModuleCode   = sADDModuleName.Upper()
    QRY_StudentExams.ParamDateExam     = EDT_ChangeDay
    HExecuteQuery(QRY_StaffModules)
    HReadFirst(QRY_StaffModules)
    IF HFound(QRY_StaffModules) THEN
        // Check if student still in allocated time
        IF Now() >= gStartTime THEN
            // Check if user already initiated the download
            // Start time if not
            HReadSeekFirst(ExamOutput,StudentNumberModuleCode,[gnUserID,sADDModuleName.Upper()])
            IF HFound(ExamOutput) = True THEN
                // Check if still in allocated time range
                SubmitByTime is Time = ExamOutput.StartTime.Time
                SubmitByTime.Hour += 2
                IF TimeDifference(TimeSys(),SubmitByTime) > 0 THEN
                    // Create directory if it does not already exist
                    IF fDirectoryExist(fDataDir() + fSep() + "JRouter" + fSep() + sADDModuleName.Upper())
                     = False THEN
                        fMakeDir(fDataDir() + fSep() + "JRouter" + fSep() + sADDModuleName.Upper())
                    END

                    sFileName    = StringFormat(MySelf[i].NameBrowserFile,ccUpCase)
                    sFileName    = StringFormat(Right(sFileName,4),ccLowCase)
                    // Check if file is PDF format
                    IF StringEndsWith(sFileName,".pdf",ccNormal) THEN
```

```
                        //STUDNUM_MODCODE_EXAM_UploadTime.pdf
                        //38446632_ICT3715_EXAM_20221109-11:03:20.pdf
                        // Rename file to correct format
                        AnswerPaper  is string
                        dDate        is DateTime  = SysDateTime()
                        AnswerPaper = NumToString(gnUserID) + "_" + sADDModuleName.Upper() + "_EXAM_"
                        AnswerPaper += DateTimeToString(dDate,"YYYYMMDD_HH-MM-SS")

                        UploadCopyFile(MySelf, fDataDir() + fSep() + "JRouter" + fSep() + sADDModuleName.
                        Upper(), AnswerPaper,i)

                        // Update ExamOutput record
                        emailResult is string
                        ExamOutput.UploadTime     = SysDateTime()
                        ExamOutput.AnswerPaperPDF = AnswerPaper + ".pdf"
                        HSave(ExamOutput)

                        // Send email to user
                        emailResult = EmailStudentUploadResult(sADDModuleName.Upper())

                        CELL_UploadExamPDF.Visible = False
                        ToastDisplay("Upload successful" + CR + emailResult,toastShort,vaMiddle,haCenter,
                        LightGreen)
                    ELSE
                        CELL_UploadExamPDF.Visible = False
                        ToastDisplay(
                        "Upload failed due to format being incorrect. Please only use PDF file type.",
                        toastShort,vaMiddle,haCenter,DarkRed)
                    END
                ELSE
                    Error("Unfortunately upload time passed.",toastShort,vaMiddle,haCenter,DarkRed)
                END
            END
        END
    END
END
```

**After reception of the files uploaded from UPL_Upload ( CELL_UploadExamPDF ) (browser)**

```
// Ungray the add and upload buttons
BTN_SEND..Grayed  = False
MySelf..Grayed        = False
```

**Add a token in EDT_ChangeDay (browser)**

```
PROCEDURE AddToken (MyToken is Token)

//RETURN False to prevent from adding the token
RETURN True
```

**Click on a token of EDT_ChangeDay (browser)**

```
PROCEDURE ClickToken (MyToken is Token)
```

**Delete a token in EDT_ChangeDay (browser)**

```
PROCEDURE DeleteToken(MyToken is Token)

//RESULT False to prevent from deleting the token
RESULT True
```

### Initializing of BTN_No ( CELL_Zone1 ) (server)

```
// Version 1
// Description
// Button that triggers a server action
```

### Click on BTN_No (onclick browser event) ( CELL_Zone1 )

```
PopupClose(POPUP_DownloadDialog)
```

### Initializing of BTN_Yes ( CELL_Zone1 ) (server)

```
// Version 1
// Description
// Button that triggers a server action
```

### Click on BTN_Yes (onclick browser event) ( CELL_Zone1 )

```
PopupClose(POPUP_DownloadDialog)
```

### Click on BTN_Yes ( CELL_Zone1 ) (server)

```
InitiateDownload()
```

### Initializing of STC_Details ( CELL_Zone1 ) (server)

```
MySelf = "Once download is done, exam time will start. Are you sure you want to continue?"
```

# PAGE_Student_Dashboard

## Procedures

### Local procedure EmailStudentUploadResult (server)

```
PROCEDURE EmailStudentUploadResult(sModCode is string) : string

// Start the SMTP session
EmailSMTPSession is emailSMTPSession
EmailSMTPSession.ServerAddress = SERVERADDRESS
EmailSMTPSession.Name = SERVEREMAIL
```

```
EmailSMTPSession.Password = SERVERPASSWORD
EmailSMTPSession.Port = SERVERPORT

// Starts the SMTP session
IF EmailSMTPSession.StartSession() = False THEN
    // Failure starting the session
    RETURN "Failure starting the session"
END

QRY_StudentExamResults.ParamModuleCode = sModCode
QRY_StudentExamResults.ParamStudentNumber = gnUserID
HExecuteQuery(QRY_StudentExamResults)
HReadFirst(QRY_StudentExamResults)
IF HFound() = False THEN
    RETURN "Failure accessing the record"
END

eEmail is Email
eEmail.Sender = SERVEREMAIL
eEmail.Subject        = "Upload for " + sModCode + " confirmation email"
eEmail.Message        = "You have successfully uploaded " + sModCode + " exam to UNISA" + CR +
"Student No: " + QRY_StudentExamResults.StudentNumber + CR +
"Student Name: " + QRY_StudentExamResults.StudentName + CR +
"File uploaded: " + QRY_StudentExamResults.AnswerPaperPDF + CR +
"Transaction ID: " + QRY_StudentExamResults.TransactionID + CR +
"Start time: " + DateToString(QRY_StudentExamResults.StartTime,maskDateEmail) + CR +
"Upload time: " + DateToString(QRY_StudentExamResults.UploadTime,maskDateEmail) + CR + CR +
"You will receive your examination results after the examination period, after marking has been completed."

// Adds a recipient
Add(eEmail.Recipient, QRY_StudentExamResults.StudentEmail)

// Send in Outlook
//For information: a name found in the address book can be used as recipient
IF EmailSMTPSession.SendMessage(eEmail) = False THEN
    // Close the session
    EmailSMTPSession.CloseSession()
    RETURN "An error occurred while trying to send the email"
ELSE
    // Close the session
    EmailSMTPSession.CloseSession()
    RETURN "Email successfully sent"
END
```

### Local procedure InitiateDownload (server)

```
PROCEDURE InitiateDownload()
// Check if user already initiated the download
// Start time if not
HReadSeekFirst(ExamOutput,StudentNumberModuleCode,[NumToString(gnUserID),sDownloadModCode])
IF HFound(ExamOutput) <> True THEN
    HReset(ExamOutput)
    ExamOutput.StartTime      = SysDateTime()
    ExamOutput.StudentNumber  = NumToString(gnUserID)
    ExamOutput.ModuleCode     = sDownloadModCode

    // Set Unique TransactionID
    newTransactionID is string    = ""
    LOOP
        newTransactionID = ServerProcedures.TransactionID(ExamOutput.StudentNumber)
```

```
        QRY_ExamOutputByModuleCode.ParamTransactionID = newTransactionID
        HExecuteQuery(QRY_ExamOutputByModuleCode)
        IF HFound(QRY_ExamOutputByModuleCode) = True THEN
            CONTINUE
        ELSE
            ExamOutput.TransactionID = newTransactionID
            HAdd(ExamOutput)
            BREAK
        END
    END
END

// Start download
//The FileDisplay function with MIME setting
//By default, the file will be opened in the browser if it can open this file
//A download box will display when the name of the file to download is specified
FileDisplay(fDataDir() + fSep() + "ExamFiles" + fSep() + sDownloadModCode + ".pdf",
"application/byte-stream", fExtractPath(fDataDir() + fSep() + "ExamFiles" + fSep() + sDownloadModCode + ".
pdf", fFileName+fExtension))
```

# PAGE_ExamDept_UploadExamSchedule

## Code

**Global declarations of PAGE_ExamDept_UploadExamSchedule (server)**

```
PROCEDURE MyPage()
```

# PAGE_ExamDept_UploadExamSchedule

## Control code

**Click on IMG_Logo (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    IF YesNo("You are currently logged in. Returning to Home page will Log you out. Continue?") = Yes THEN
        // End session and logout
        gnUserID          = 0
        gsSession             = ""
        gsUser                = ""
        gsUserType            = ""

        HCloseConnection(gConnection)
        PageDisplay(PAGE_Login)
    END
END
```

**Initializing of Link_Logout (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    Link_Logout..Visible = True
END
```

**Click on Link_Logout (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    // End session and logout
    gnUserID          = 0
    gsSession             = ""
    gsUser                = ""
    gsUserType            = ""

    HCloseConnection(gConnection)
    PageDisplay(PAGE_Login)
END
```

**Initializing of STC_WelcomeUser (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    STC_WelcomeUser.Caption = "Welcome " + gsUser
ELSE
    STC_WelcomeUser.Caption = ""
END
```

### Initializing of MENU_Nav (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Initializing of MENU_Nav (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (browser)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Return from AJAX process after selecting the ZONE_Menu.MENU_Nav.Home menu ( MENU_Nav ) (browser)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (browser)

```
//Run the process defined in the template
ExecuteAncestor
```

### Return from AJAX process after selecting the ZONE_Menu.MENU_Nav.Home menu ( MENU_Nav ) (browser)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.OPT_ExamScheduleUpload menu option ( MENU_Nav ) (browser)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.OPT_ExamScheduleUpload menu option ( MENU_Nav ) (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.OPT_ExamScheduleUpload menu option ( MENU_Nav ) (server)

```
//Run the process defined in the template
ExecuteAncestor
```

**Return from AJAX process after selecting the ZONE_Menu.MENU_Nav.OPT_ExamScheduleUpload menu (**

```
//Run the process defined in the template
ExecuteAncestor
```

**Initializing of BTN_UploadExamSchedule (server)**

```
// Version 1
// Description
// Button that triggers a server action
```

**Click on BTN_UploadExamSchedule (server)**

```
PageDisplayDialog(PAGE_InfoWait,EDT_File_Location,"ExamSchedule")
```

**Initializing of EDT_File_Location ( CELL_NoName2 ) (server)**

```
// Version 1
// Description
// Edit control for plain single-line text
```

**Add a token in EDT_File_Location ( CELL_NoName2 ) (browser)**

```
PROCEDURE AjoutJeton (MonJeton is a Token)

//RENVOYER Faux pour interdire l'ajout du jeton
RESULT True
```

**Click on a token of EDT_File_Location ( CELL_NoName2 ) (browser)**

```
PROCEDURE ClicJeton (MonJeton is a Token)
```

**Delete a token in EDT_File_Location ( CELL_NoName2 ) (browser)**

```
PROCEDURE SuppressionJeton (MonJeton is a Token)

//RENVOYER Faux pour interdire la suppression du jeton
RESULT True
```

# PAGE_Student_Login

## Code

**Global declarations of PAGE_Student_Login (server)**

```
PROCEDURE MyPage()
```

# PAGE_Student_Login

## Control code

**Click on IMG_Logo (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    IF YesNo("You are currently logged in. Returning to Home page will Log you out. Continue?") = Yes THEN
        // End session and logout
        gnUserID          = 0
        gsSession             = ""
        gsUser                = ""
        gsUserType            = ""

        HCloseConnection(gConnection)
        PageDisplay(PAGE_Login)
    END
END
```

**Initializing of Link_Logout (server)**

```
//Run the process defined in the template
ExecuteAncestor
```

**Initializing of Link_Logout (server)**

```
//Run the process defined in the template
ExecuteAncestor
```

**Click on Link_Logout (onclick browser event)**

```
//Run the process defined in the template
ExecuteAncestor
```

**Click on Link_Logout (server)**

```
//Run the process defined in the template
ExecuteAncestor
```

**Click on Link_Logout (server)**

```
//Run the process defined in the template
ExecuteAncestor
```

**Return from AJAX process after clicking on Link_Logout (browser)**

```
//Run the process defined in the template
ExecuteAncestor
```

**Initializing of STC_WelcomeUser (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    STC_WelcomeUser.Caption = "Welcome " + gsUser
ELSE
    STC_WelcomeUser.Caption = ""
END
```

**Initializing of MENU_Nav (server)**

```
//Run the process defined in the template
ExecuteAncestor
```

**Initializing of MENU_Nav (server)**

```
//Run the process defined in the template
ExecuteAncestor
```

**Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (server) (PAGETPL_Session template)**

```
IF gsUser = ExamDeptUser THEN
    PageDisplay(PAGE_Exam_Dashboard)
END
IF gsUserType = "Lecturer" THEN
     PageDisplay(PAGE_Lecturer_Dashboard)
END
IF gsUserType = "Student" THEN
    PageDisplay(PAGE_Student_Dashboard)
END
```

**Click on BTN_CANCEL (onclick browser event) ( CELL_NoName1 )**

```
history.back();
```

**Click on BTN_OK (onclick browser event) ( CELL_NoName1 )**

```
IF EDT_USERNUMBER = "" THEN
    ReturnToCapture(EDT_USERNUMBER)
END
IF EDT_PASSWORD = "" THEN
    ReturnToCapture(EDT_PASSWORD)
END
```

**Click on BTN_OK ( CELL_NoName1 ) (server)**

```
bUserLogin is boolean
bUserLogin = Connection()
```

```
IF bUserLogin = False THEN
    EDT_PASSWORD                = ""
    STC_Error                       = "Invalid user or password"
    STC_Error..Visible              = True
ELSE
    PageDisplay(PAGE_Student_Dashboard)
END
```

**Add a token in EDT_PASSWORD ( CELL_NoName1 ) (browser)**

```
PROCEDURE AjoutJeton (MonJeton is a Token)

//RENVOYER Faux pour interdire l'ajout du jeton
RESULT True
```

**Click on a token of EDT_PASSWORD ( CELL_NoName1 ) (browser)**

```
PROCEDURE ClicJeton (MonJeton is a Token)
```

**Delete a token in EDT_PASSWORD ( CELL_NoName1 ) (browser)**

```
PROCEDURE SuppressionJeton (MonJeton is a Token)

//RENVOYER Faux pour interdire la suppression du jeton
RESULT True
```

**Add a token in EDT_USERNUMBER ( CELL_NoName1 ) (browser)**

```
PROCEDURE AjoutJeton (MonJeton is a Token)

//RENVOYER Faux pour interdire l'ajout du jeton
RESULT True
```

**Click on a token of EDT_USERNUMBER ( CELL_NoName1 ) (browser)**

```
PROCEDURE ClicJeton (MonJeton is a Token)
```

**Delete a token in EDT_USERNUMBER ( CELL_NoName1 ) (browser)**

```
PROCEDURE SuppressionJeton (MonJeton is a Token)

//RENVOYER Faux pour interdire la suppression du jeton
RESULT True
```

# PAGE_Student_Login

## Procedures

**Local procedure Connection (server)**

```
PROCEDURE Connection()
bOnUser is boolean = False
```

```
// Finds the user
HReadSeekFirst(StudentInfo,StudentNumber,EDT_USERNUMBER,hIdentical)
IF HFound(StudentInfo) THEN

    // Checks the password
    IF StudentInfo.StudentPassword = EDT_PASSWORD THEN

        // Generates a session
        gsUserType        = "Student"
        gnUserID      = StudentInfo.StudentNumber
        gsUser            = StudentInfo.StudentName

        GenerateSession()
        bOnUser = True
    END
END

RETURN bOnUser
```

**Local procedure GenerateSession (server)**

```
PROCEDURE GenerateSession()
// Initialize new connection
New_connection is Connection
// Connection parameters
New_connection..Provider       = hAccessHFClientServer
New_connection..User           = gsUser
New_connection..Password       = EDT_PASSWORD
New_connection..Server             = "localhost"
New_connection..Database       = "ICT3715_DB"
New_connection..CryptMethod        = hEncryptionNO

// Generates a session identifier
dtdhDateTimeCurrent is DateTime = Today() + TimeSys()
gsSession = dtdhDateTimeCurrent + TAB + gnUserID + "-" + gsUser

HChangeConnection("ExamOutput,ExamSetup,StudentModule",New_connection)

gConnection = New_connection

// Backup
CookieWrite("session", gsSession, 1)
```

# PAGE_Lecturer_Login

## Code

### Global declarations of PAGE_Lecturer_Login (server)

```
PROCEDURE MyPage()
```

# PAGE_Lecturer_Login

## Control code

### Click on IMG_Logo (server) (PAGETPL_Session template)

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    IF YesNo("You are currently logged in. Returning to Home page will Log you out. Continue?") = Yes THEN
        // End session and logout
        gnUserID          = 0
        gsSession             = ""
        gsUser                = ""
        gsUserType            = ""

        HCloseConnection(gConnection)
        PageDisplay(PAGE_Login)
    END
END
```

### Initializing of Link_Logout (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Initializing of Link_Logout (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Click on Link_Logout (onclick browser event)

```
//Run the process defined in the template
ExecuteAncestor
```

### Click on Link_Logout (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Click on Link_Logout (server)

```
//Run the process defined in the template
ExecuteAncestor
```

**Return from AJAX process after clicking on Link_Logout (browser)**

```
//Run the process defined in the template
ExecuteAncestor
```

**Initializing of STC_WelcomeUser (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    STC_WelcomeUser.Caption = "Welcome " + gsUser
ELSE
    STC_WelcomeUser.Caption = ""
END
```

**Initializing of MENU_Nav (server)**

```
//Run the process defined in the template
ExecuteAncestor
```

**Initializing of MENU_Nav (server)**

```
//Run the process defined in the template
ExecuteAncestor
```

**Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (server) (PAGETPL_Session template)**

```
IF gsUser = ExamDeptUser THEN
    PageDisplay(PAGE_Exam_Dashboard)
END
IF gsUserType = "Lecturer" THEN
     PageDisplay(PAGE_Lecturer_Dashboard)
END
IF gsUserType = "Student" THEN
    PageDisplay(PAGE_Student_Dashboard)
END
```

**Click on BTN_CANCEL (onclick browser event) ( CELL_NoName1 )**

```
history.back();
```

**Click on BTN_OK (onclick browser event) ( CELL_NoName1 )**

```
IF EDT_USERNUMBER = "" THEN
    ReturnToCapture(EDT_USERNUMBER)
END
IF EDT_PASSWORD = "" THEN
    ReturnToCapture(EDT_PASSWORD)
END
```

**Click on BTN_OK ( CELL_NoName1 ) (server)**

```
bUserLogin is boolean
bUserLogin = Connection()
```

```
IF bUserLogin = False THEN
    EDT_PASSWORD                = ""
    STC_Error                       = "Invalid user or password"
    STC_Error..Visible              = True
ELSE
    PageDisplay(PAGE_Lecturer_Dashboard)
END
```

**Add a token in EDT_PASSWORD ( CELL_NoName1 ) (browser)**

```
PROCEDURE AjoutJeton (MonJeton is a Token)

//RENVOYER Faux pour interdire l'ajout du jeton
RESULT True
```

**Click on a token of EDT_PASSWORD ( CELL_NoName1 ) (browser)**

```
PROCEDURE ClicJeton (MonJeton is a Token)
```

**Delete a token in EDT_PASSWORD ( CELL_NoName1 ) (browser)**

```
PROCEDURE SuppressionJeton (MonJeton is a Token)

//RENVOYER Faux pour interdire la suppression du jeton
RESULT True
```

**Add a token in EDT_USERNUMBER ( CELL_NoName1 ) (browser)**

```
PROCEDURE AjoutJeton (MonJeton is a Token)

//RENVOYER Faux pour interdire l'ajout du jeton
RESULT True
```

**Click on a token of EDT_USERNUMBER ( CELL_NoName1 ) (browser)**

```
PROCEDURE ClicJeton (MonJeton is a Token)
```

**Delete a token in EDT_USERNUMBER ( CELL_NoName1 ) (browser)**

```
PROCEDURE SuppressionJeton (MonJeton is a Token)

//RENVOYER Faux pour interdire la suppression du jeton
RESULT True
```

# PAGE_Lecturer_Login

## Procedures

**Local procedure Connection (server)**

```
PROCEDURE Connection()
bOnUser is boolean = False
```

```
// Finds the user
HReadSeekFirst(StaffInfo,StaffNumber,EDT_USERNUMBER,hIdentical)
IF HFound(StaffInfo) THEN

    // Checks the password
    IF StaffInfo.Password = EDT_PASSWORD THEN

        // Generates a session
        gsUserType        = "Lecturer"
        gnUserID      = StaffInfo.StaffNumber
        gsUser            = StaffInfo.Name

        GenerateSession()
        bOnUser = True
    END
END

RETURN bOnUser
```

**Local procedure GenerateSession (server)**

```
PROCEDURE GenerateSession()
// Initialize new connection
New_connection is Connection
// Connection parameters
New_connection..Provider       = hAccessHFClientServer
New_connection..User           = gsUser
New_connection..Password       = EDT_PASSWORD
New_connection..Server             = "localhost"
New_connection..Database       = "ICT3715_DB"
New_connection..CryptMethod        = hEncryptionNO

// Generates a session identifier
dtdhDateTimeCurrent is DateTime = Today() + TimeSys()
gsSession = dtdhDateTimeCurrent + TAB + gnUserID + "-" + gsUser

HChangeConnection("ExamSetup,ModuleLeader,StaffInfo",New_connection)

gConnection = New_connection

// Backup
CookieWrite("session", gsSession, 1)
```

# PAGE_Exam_Department_Login

## Code

**Global declarations of PAGE_Exam_Department_Login (server)**

```
PROCEDURE MyPage()
```

# PAGE_Exam_Department_Login

## Control code

**Click on IMG_Logo (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    IF YesNo("You are currently logged in. Returning to Home page will Log you out. Continue?") = Yes THEN
        // End session and logout
        gnUserID          = 0
        gsSession             = ""
        gsUser                = ""
        gsUserType            = ""

        HCloseConnection(gConnection)
        PageDisplay(PAGE_Login)
    END
END
```

**Initializing of Link_Logout (server)**

```
//Run the process defined in the template
ExecuteAncestor
```

**Initializing of Link_Logout (server)**

```
//Run the process defined in the template
ExecuteAncestor
```

**Click on Link_Logout (onclick browser event)**

```
//Run the process defined in the template
ExecuteAncestor
```

**Click on Link_Logout (server)**

```
//Run the process defined in the template
ExecuteAncestor
```

**Click on Link_Logout (server)**

```
//Run the process defined in the template
ExecuteAncestor
```

**Return from AJAX process after clicking on Link_Logout (browser)**

```
//Run the process defined in the template
ExecuteAncestor
```

**Initializing of STC_WelcomeUser (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    STC_WelcomeUser.Caption = "Welcome " + gsUser
ELSE
    STC_WelcomeUser.Caption = ""
END
```

**Initializing of MENU_Nav (server)**

```
//Run the process defined in the template
ExecuteAncestor
```

**Initializing of MENU_Nav (server)**

```
//Run the process defined in the template
ExecuteAncestor
```

**Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (server) (PAGETPL_Session template)**

```
IF gsUser = ExamDeptUser THEN
    PageDisplay(PAGE_Exam_Dashboard)
END
IF gsUserType = "Lecturer" THEN
     PageDisplay(PAGE_Lecturer_Dashboard)
END
IF gsUserType = "Student" THEN
    PageDisplay(PAGE_Student_Dashboard)
END
```

**Click on BTN_CANCEL (onclick browser event) ( CELL_NoName1 )**

```
history.back();
```

**Click on BTN_OK (onclick browser event) ( CELL_NoName1 )**

```
IF EDT_USERNUMBER = "" THEN
    ReturnToCapture(EDT_USERNUMBER)
END
IF EDT_PASSWORD = "" THEN
    ReturnToCapture(EDT_PASSWORD)
END
```

**Click on BTN_OK ( CELL_NoName1 ) (server)**

```
bUserLogin is boolean
bUserLogin = Connection()
```

```
IF bUserLogin = False THEN
    EDT_PASSWORD                = ""
    STC_Error                       = "Invalid user or password"
    STC_Error..Visible              = True
ELSE
    PageDisplay(PAGE_Exam_Dashboard)
END
```

**Add a token in EDT_PASSWORD ( CELL_NoName1 ) (browser)**

```
PROCEDURE AjoutJeton (MonJeton is a Token)

//RENVOYER Faux pour interdire l'ajout du jeton
RESULT True
```

**Click on a token of EDT_PASSWORD ( CELL_NoName1 ) (browser)**

```
PROCEDURE ClicJeton (MonJeton is a Token)
```

**Delete a token in EDT_PASSWORD ( CELL_NoName1 ) (browser)**

```
PROCEDURE SuppressionJeton (MonJeton is a Token)

//RENVOYER Faux pour interdire la suppression du jeton
RESULT True
```

**Add a token in EDT_USERNUMBER ( CELL_NoName1 ) (browser)**

```
PROCEDURE AjoutJeton (MonJeton is a Token)

//RENVOYER Faux pour interdire l'ajout du jeton
RESULT True
```

**Click on a token of EDT_USERNUMBER ( CELL_NoName1 ) (browser)**

```
PROCEDURE ClicJeton (MonJeton is a Token)
```

**Delete a token in EDT_USERNUMBER ( CELL_NoName1 ) (browser)**

```
PROCEDURE SuppressionJeton (MonJeton is a Token)

//RENVOYER Faux pour interdire la suppression du jeton
RESULT True
```

# PAGE_Exam_Department_Login

## Procedures

**Local procedure Connection (server)**

```
PROCEDURE Connection()
```

```
bOnUser is boolean = False

// Finds the user
IF EDT_USERNUMBER = ExamDeptNumber _AND_ EDT_PASSWORD = ExamDeptPassword THEN
    // Generates a session
    gsUserType        = ExamDeptUser
    gnUserID      = ExamDeptNumber
    gsUser            = ExamDeptUser

    GenerateSession()
    bOnUser = True
END

RETURN bOnUser
```

**Local procedure GenerateSession (server)**

```
PROCEDURE GenerateSession()
// Initialize new connection
New_connection is Connection
// Connection parameters
New_connection..Provider       = hAccessHFClientServer
New_connection..User           = gsUser
New_connection..Password       = EDT_PASSWORD
New_connection..Server             = "localhost"
New_connection..Database       = "ICT3715_DB"
New_connection..CryptMethod        = hEncryptionNO

// Generates a session identifier
dtdhDateTimeCurrent is DateTime = Today() + TimeSys()
gsSession = dtdhDateTimeCurrent + TAB + gnUserID + "-" + gsUser

HChangeConnection("ModuleLeader,ExamSetup,ModuleInfo,StaffInfo,StudentModule",New_connection)

gConnection = New_connection

// Backup
CookieWrite("session", gsSession, 1)
```

# PAGE_YesNoDialog

## Code

**Global declarations of PAGE_YesNoDialog (server)**

```
PROCEDURE MyPage(sDetails is string)
```

# PAGE_YesNoDialog

## Control code

**Initializing of BTN_No ( CELL_Zone1 ) (server)**

```
// Version 1
// Description
// Button that triggers a server action
```

**Click on BTN_No ( CELL_Zone1 ) (server)**

```
PageCloseDialog(False)
PageDisplay(PAGE_Student_Dashboard,False)
```

**Initializing of BTN_Yes ( CELL_Zone1 ) (server)**

```
// Version 1
// Description
// Button that triggers a server action
```

**Click on BTN_Yes ( CELL_Zone1 ) (server)**

```
PageCloseDialog(True)
PageDisplay(PAGE_Student_Dashboard,True)
```

**Initializing of STC_Details ( CELL_Zone1 ) (server)**

```
MySelf = sDetails
```

# PAGE_Lecturer_Exam_Viewer

## Code

**Global declarations of PAGE_Lecturer_Exam_Viewer (server)**

```
PROCEDURE MyPage()
```

# PAGE_Lecturer_Exam_Viewer

## Control code

**Click on IMG_Logo (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    IF YesNo("You are currently logged in. Returning to Home page will Log you out. Continue?") = Yes THEN
        // End session and logout
        gnUserID          = 0
        gsSession             = ""
        gsUser                = ""
        gsUserType            = ""

        HCloseConnection(gConnection)
        PageDisplay(PAGE_Login)
    END
END
```

**Initializing of Link_Logout (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    Link_Logout..Visible = True
END
```

**Click on Link_Logout (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    // End session and logout
    gnUserID          = 0
    gsSession             = ""
    gsUser                = ""
    gsUserType            = ""

    HCloseConnection(gConnection)
    PageDisplay(PAGE_Login)
END
```

**Initializing of STC_WelcomeUser (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    STC_WelcomeUser.Caption = "Welcome " + gsUser
ELSE
    STC_WelcomeUser.Caption = ""
END
```

### Initializing of MENU_Nav (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Initializing of MENU_Nav (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (browser)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Return from AJAX process after selecting the ZONE_Menu.MENU_Nav.Home menu ( MENU_Nav ) (browser)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (browser)

```
//Run the process defined in the template
ExecuteAncestor
```

### Return from AJAX process after selecting the ZONE_Menu.MENU_Nav.Home menu ( MENU_Nav ) (browser)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.OPT_ExamView menu option ( MENU_Nav ) (browser)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.OPT_ExamView menu option ( MENU_Nav ) (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.OPT_ExamView menu option ( MENU_Nav ) (server)

```
//Run the process defined in the template
ExecuteAncestor
```

**Return from AJAX process after selecting the ZONE_Menu.MENU_Nav.OPT_ExamView menu ( MENU_Nav )**

```
//Run the process defined in the template
ExecuteAncestor
```

**Initializing of TREE_WrittenExams ( CELL_NoName1 ) (server)**

```
arrModules is array of strings

// Initialize the TreeView control
TreeAdd(TREE_WrittenExams, "Root", tvDefault, tvDefault, "")

QRY_StaffModules.ParamStaffNumber = NumToString(gnUserID)
HExecuteQuery(QRY_StaffModules)
FOR EACH QRY_StaffModules
    sFilename is string =  fSep() + "JRouter" + fSep() + QRY_StaffModules.ModuleCode
    IF fDirectoryExist(fDataDir() + sFilename) THEN
        // Modules
        TreeAdd(TREE_WrittenExams, "Root" + TAB + QRY_StaffModules.ModuleCode, tvDefault, tvDefault, "")
        ArrayAdd(arrModules,QRY_StaffModules.ModuleCode)
    END
END

// Get treeview count
nNBTreeCount is int = arrModules.Count()

// Run query for each module in treeview and get student exam if exists
IF nNBTreeCount > 0 THEN
    dropArrow is Image = "Drop-arrow.png"
    countVal is int = 1
    LOOP (nNBTreeCount)
        QRY_ExamOutputByModuleCode.ParamModuleCode = arrModules[countVal]
        HExecuteQuery(QRY_ExamOutputByModuleCode)
        FOR EACH QRY_ExamOutputByModuleCode
            // Student PDF files
            TreeAdd(TREE_WrittenExams, "Root" + TAB + QRY_ExamOutputByModuleCode.ModuleCode + TAB +
            QRY_ExamOutputByModuleCode.AnswerPaperPDF, dropArrow, dropArrow, "")
        END
        countVal++
    END
END

MySelf.Expand("Root")
```

**Click on TREE_WrittenExams ( CELL_NoName1 ) (server)**

```
sFullSelectPath is array of strings = StringSplit(MySelf.Select(),TAB)

/* Original code -->
sPath is string
sPath = "JRouter" + fSep() + sFullSelectPath[2] + fSep() + sFullSelectPath[3]
sPath = Replace(sPath,":","")
sPath = fDataDir() + fSep() + sPath

PageDisplay(PAGE_Lecturer_Exam_PDFViewer,sPath)
*/
```

```
// Code for testing day purposes

sPath is string
sPath = "Test folders and files" + fSep() + "TestExamSubmissionFile.pdf"
PageDisplay(PAGE_Lecturer_Exam_PDFViewer,sPath)
```

# PAGE_Lecturer_Exam_PDFViewer

## Code

**Global declarations of PAGE_Lecturer_Exam_PDFViewer (server)**

```
PROCEDURE MyPage(gsPath is string)
```

# PAGE_Lecturer_Exam_PDFViewer

## Control code

**Click on IMG_Logo (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    IF YesNo("You are currently logged in. Returning to Home page will Log you out. Continue?") = Yes THEN
        // End session and logout
        gnUserID          = 0
        gsSession             = ""
        gsUser                = ""
        gsUserType            = ""

        HCloseConnection(gConnection)
        PageDisplay(PAGE_Login)
    END
END
```

**Initializing of Link_Logout (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    Link_Logout..Visible = True
END
```

**Click on Link_Logout (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    // End session and logout
    gnUserID          = 0
    gsSession             = ""
    gsUser                = ""
    gsUserType            = ""

    HCloseConnection(gConnection)
    PageDisplay(PAGE_Login)
END
```

**Initializing of STC_WelcomeUser (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    STC_WelcomeUser.Caption = "Welcome " + gsUser
ELSE
    STC_WelcomeUser.Caption = ""
END
```

### Initializing of MENU_Nav (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Initializing of MENU_Nav (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (browser)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Return from AJAX process after selecting the ZONE_Menu.MENU_Nav.Home menu ( MENU_Nav ) (browser)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (browser)

```
//Run the process defined in the template
ExecuteAncestor
```

### Return from AJAX process after selecting the ZONE_Menu.MENU_Nav.Home menu ( MENU_Nav ) (browser)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.OPT_ExamView menu option ( MENU_Nav ) (browser)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.OPT_ExamView menu option ( MENU_Nav ) (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Select the ZONE_Menu.MENU_Nav.OPT_ExamView menu option ( MENU_Nav ) (server)

```
//Run the process defined in the template
ExecuteAncestor
```

### Return from AJAX process after selecting the ZONE_Menu.MENU_Nav.OPT_ExamView menu ( MENU_Nav )

```
//Run the process defined in the template
ExecuteAncestor
```

### Initializing of IFRM_PDF (server)

```
// Version 1
// Description
// Displays a PDF document in an Iframe control

// Define the path on disk of the PDF file to be displayed
// Displays the PDF via an intermediate page
// Purpose: Allow displaying a PDF file even if it cannot be directly accessed via a URL
ContextOpen(PAGE_Iframe_PDF,gsPath,Encode(HashFile(HA_HMAC_SHA_256,gsPath,"secret key"),encodeBASE64URL))
MyControl = PageAddress(PAGE_Iframe_PDF) + "?" + PAGE_Iframe_PDF.BTN_Retry.Alias
```

# PAGE_Iframe_PDF

## Code

**Global declarations of PAGE_Iframe_PDF (server)**

```
PROCEDURE MyPage(gsPDFPath="",gsHash="")

IF gsPDFPath = "" THEN gsPDFPath = PageParameter("PDF")
IF gsHash = "" THEN gsHash = PageParameter("HASH")
```

# PAGE_Iframe_PDF

## Control code

**Click on BTN_Retry (server)**

```
// Displays the PDF if the provided hash is verified
IF HashCheckFile(HA_HMAC_SHA_256,gsPDFPath,Decode(gsHash,encodeBASE64URL),"secret key") THEN
    FileDisplay(gsPDFPath,typeMimePDF)
END
```

# PAGE_Lecturer_SummaryReport

## Code

**Global declarations of PAGE_Lecturer_SummaryReport (server)**

```
PROCEDURE MyPage()
gChartNmbOfCategories is int
```

**Initializing of PAGE_Lecturer_SummaryReport (server)**

```
LoadSummary()
```

# PAGE_Lecturer_SummaryReport

## Control code

**Click on IMG_Logo (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    IF YesNo("You are currently logged in. Returning to Home page will Log you out. Continue?") = Yes THEN
        // End session and logout
        gnUserID          = 0
        gsSession             = ""
        gsUser                = ""
        gsUserType            = ""

        HCloseConnection(gConnection)
        PageDisplay(PAGE_Login)
    END
END
```

**Initializing of Link_Logout (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    Link_Logout..Visible = True
END
```

**Click on Link_Logout (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    // End session and logout
    gnUserID          = 0
    gsSession             = ""
    gsUser                = ""
    gsUserType            = ""
```

```
    HCloseConnection(gConnection)
    PageDisplay(PAGE_Login)
END
```

### Initializing of STC_WelcomeUser (server) (PAGETPL_Session template)

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    STC_WelcomeUser.Caption = "Welcome " + gsUser
ELSE
    STC_WelcomeUser.Caption = ""
END
```

### Initializing of MENU_Nav (server) (PAGETPL_Session template)

```
MySelf.OPT_ExamView.Visible = False
MySelf.OPT_ExamScheduleUpload.Visible = False

IF gsUser = ExamDeptUser THEN
    MySelf.OPT_ExamScheduleUpload.Visible = True
END
IF gsUserType = "Lecturer" THEN
    MySelf.OPT_ExamView.Visible = True
END
```

### Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (server) (PAGETPL_Session template)

```
IF gsUser = ExamDeptUser THEN
    PageDisplay(PAGE_Exam_Dashboard)
END
IF gsUserType = "Lecturer" THEN
     PageDisplay(PAGE_Lecturer_Dashboard)
END
IF gsUserType = "Student" THEN
    PageDisplay(PAGE_Student_Dashboard)
END
```

### Click on BTN_NoName1 (onclick browser event)

```
PopupDisplay(POPUP_Combobox,popupFixed+popupDiscardable)
```

### Initializing of BTN_SetDayCombo (server)

```
// Version 1
// Description
// Button that triggers a server action
```

### Click on BTN_SetDayCombo (server)

```
sComboValue is string
IF COMBO_LecturerStats.Select() <> 1 THEN
    indexCounter is int = 2
    uploadsCounter is int = 0
    // Clear Lecturer Modules Chart
```

```
    CHART_LecturerModules.DeleteAll()
    uploadsCounter = HNbRec(ExamOutput)

    sComboValue = ExtractStringBetween(COMBO_LecturerStats[COMBO_LecturerStats.Value],1,"(",")",
    FromBeginning)
    HReadSeekFirst(StaffInfo,StaffInfo.StaffNumber,sComboValue)
    IF HFound(StaffInfo) THEN
        STC_LecturerName                    = StaffInfo.Name
        STC_LecturerEmail                   = StaffInfo.Email

        TABLE_QRY_StaffModules_Summary.DeleteAll()
        QRY_StaffModules.ParamStaffNumber = StaffInfo.StaffNumber
        HExecuteQuery(QRY_StaffModules)
        FOR EACH QRY_StaffModules
            TableAddLine(TABLE_QRY_StaffModules_Summary,QRY_StaffModules.ModuleCode,DateToString(
            QRY_StaffModules.DateExam,"YYYY-MM-DD"),QRY_StaffModules.Description)

            // Fill Lecturer Modules Chart
            QRY_ExamModuleSubmissions.ParamModuleCode = QRY_StaffModules.ModuleCode
            HExecuteQuery(QRY_ExamModuleSubmissions)
            indexCounter += 1
            grCategoryLabel(CHART_LecturerModules,indexCounter,QRY_ExamModuleSubmissions.ModuleCode +
            " uploads")
            grAddData(CHART_LecturerModules,1,indexCounter,HNbRec(QRY_ExamModuleSubmissions))
            uploadsCounter -= HNbRec(QRY_ExamModuleSubmissions)
        END

        // Fill Lecturer Modules Chart
        grCategoryLabel(CHART_LecturerModules,1,"Other unique uploads")
        grAddData(CHART_LecturerModules,1,1,uploadsCounter)

        // Draw the Chart
        grDraw(CHART_LecturerModules)
        CELL_Statistics.Visible                 = True
    END
ELSE
    CELL_Statistics.Visible = False
END
```

**Add a token in COL_DateExam ( TABLE_QRY_StaffModules_Summary ) (browser)**

```
PROCEDURE AddToken (MyToken is Token)

//RETURN False to prevent from adding the token
RETURN True
```

**Click on a token of COL_DateExam ( TABLE_QRY_StaffModules_Summary ) (browser)**

```
PROCEDURE ClickToken (MyToken is Token)
```

**Delete a token in COL_DateExam ( TABLE_QRY_StaffModules_Summary ) (browser)**

```
PROCEDURE DeleteToken(MyToken is Token)

//RETURN False to prevent from deleting the token
RETURN True
```

**Add a token in COL_Description ( TABLE_QRY_StaffModules_Summary ) (browser)**

```
PROCEDURE AddToken (MyToken is Token)

//RETURN False to prevent from adding the token
RETURN True
```

### Click on a token of COL_Description ( TABLE_QRY_StaffModules_Summary ) (browser)

```
PROCEDURE ClickToken (MyToken is Token)
```

### Delete a token in COL_Description ( TABLE_QRY_StaffModules_Summary ) (browser)

```
PROCEDURE DeleteToken(MyToken is Token)

//RETURN False to prevent from deleting the token
RETURN True
```

### Add a token in COL_ModuleCode ( TABLE_QRY_StaffModules_Summary ) (browser)

```
PROCEDURE AddToken (MyToken is Token)

//RETURN False to prevent from adding the token
RETURN True
```

### Click on a token of COL_ModuleCode ( TABLE_QRY_StaffModules_Summary ) (browser)

```
PROCEDURE ClickToken (MyToken is Token)
```

### Delete a token in COL_ModuleCode ( TABLE_QRY_StaffModules_Summary ) (browser)

```
PROCEDURE DeleteToken(MyToken is Token)

//RETURN False to prevent from deleting the token
RETURN True
```

### Initializing of COMBO_LecturerStats (server)

```
// Version 1
// Description
// Combo box for selecting a continent

// Initialize Listbox value
ListSelectPlus(MySelf,0)
```

### Click on BTN_Daily_Report (onclick browser event) ( POPUP_Combobox )

```
// Functionality to implement as required
PopupClose()
```

### Click on BTN_Daily_Report ( POPUP_Combobox ) (server)

```
PageDisplay(PAGE_Exam_Dashboard)
```

### Click on BTN_LecturerSummaryReport (onclick browser event) ( POPUP_Combobox )

```
PopupClose()
```

**Click on BTN_LecturerSummaryReport ( POPUP_Combobox ) (server)**

```
PageDisplay(PAGE_Lecturer_SummaryReport)
```

**Click on BTN_StatisticalReport (onclick browser event) ( POPUP_Combobox )**

```
PopupClose()
```

**Click on BTN_StatisticalReport ( POPUP_Combobox ) (server)**

```
PageDisplay(PAGE_StatisticalReport)
```

**Click on BTN_WeeklyReport (onclick browser event) ( POPUP_Combobox )**

```
PopupClose()
```

**Click on BTN_WeeklyReport ( POPUP_Combobox ) (server)**

```
PageDisplay(PAGE_WeeklyReport)
```

# PAGE_Lecturer_SummaryReport

## Procedures

**Local procedure LoadSummary (server)**

```
PROCEDURE LoadSummary()
// Update Chart
grDeleteSeries(CHART_Uploads,1,grData)
numOfCurrentCategory  is int
sCurrentModule        is string    = ""
numOfSubmitions is int

HExecuteQuery(QRY_LecturerSummaryRpt_Chart)
FOR EACH QRY_LecturerSummaryRpt_Chart
    // New module code
    IF sCurrentModule = "" THEN
        // Set initializing values
        numOfCurrentCategory = 1
        sCurrentModule = QRY_LecturerSummaryRpt_Chart.ModuleCode
        numOfSubmitions = 1
    ELSE
        IF QRY_LecturerSummaryRpt_Chart.ModuleCode <> sCurrentModule THEN
            // Add collected data to graph
            grCategoryLabel(CHART_Uploads,numOfCurrentCategory,sCurrentModule)
            grAddData(CHART_Uploads,1,numOfSubmitions)
            numOfCurrentCategory += 1
```

```
            sCurrentModule = QRY_LecturerSummaryRpt_Chart.ModuleCode
            numOfSubmitions = 1
        ELSE
            numOfSubmitions += 1
        END
    END
END

// Add last data
grCategoryLabel(CHART_Uploads,numOfCurrentCategory,sCurrentModule)
grAddData(CHART_Uploads,1,numOfSubmitions)

// Draw chart
grDraw(CHART_Uploads)

// Set Number of categories in chart
gChartNmbOfCategories = numOfCurrentCategory

// Clear Combo box and refill
COMBO_LecturerStats.DeleteAll()
ListAdd(COMBO_LecturerStats,"Select a lecturer...")
ListSelectPlus(COMBO_LecturerStats,1)

// Fill combobox
sComboValue is string
FOR EACH StaffInfo
    sComboValue = StaffInfo.Name + " (" + StaffInfo.StaffNumber + ")"
    ListAdd(COMBO_LecturerStats,sComboValue)
END
```

# PAGE_WeeklyReport

## Code

**Global declarations of PAGE_WeeklyReport (server)**

```
PROCEDURE MyPage()
gDate is Date
```

**Initializing of PAGE_WeeklyReport (server)**

```
LoadDateValue()
```

# PAGE_WeeklyReport

## Control code

**Click on IMG_Logo (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    IF YesNo("You are currently logged in. Returning to Home page will Log you out. Continue?") = Yes THEN
        // End session and logout
        gnUserID          = 0
        gsSession             = ""
        gsUser                = ""
        gsUserType            = ""

        HCloseConnection(gConnection)
        PageDisplay(PAGE_Login)
    END
END
```

**Initializing of Link_Logout (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    Link_Logout..Visible = True
END
```

**Click on Link_Logout (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    // End session and logout
    gnUserID          = 0
    gsSession             = ""
    gsUser                = ""
    gsUserType            = ""
```

```
    HCloseConnection(gConnection)
    PageDisplay(PAGE_Login)
END
```

### Initializing of STC_WelcomeUser (server) (PAGETPL_Session template)

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    STC_WelcomeUser.Caption = "Welcome " + gsUser
ELSE
    STC_WelcomeUser.Caption = ""
END
```

### Initializing of MENU_Nav (server) (PAGETPL_Session template)

```
MySelf.OPT_ExamView.Visible = False
MySelf.OPT_ExamScheduleUpload.Visible = False

IF gsUser = ExamDeptUser THEN
    MySelf.OPT_ExamScheduleUpload.Visible = True
END
IF gsUserType = "Lecturer" THEN
    MySelf.OPT_ExamView.Visible = True
END
```

### Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (server) (PAGETPL_Session template)

```
IF gsUser = ExamDeptUser THEN
    PageDisplay(PAGE_Exam_Dashboard)
END
IF gsUserType = "Lecturer" THEN
     PageDisplay(PAGE_Lecturer_Dashboard)
END
IF gsUserType = "Student" THEN
    PageDisplay(PAGE_Student_Dashboard)
END
```

### Click on BTN_NoName1 (onclick browser event)

```
PopupDisplay(POPUP_Combobox,popupFixed+popupDiscardable)
```

### Initializing of BTN_SetDay (server)

```
// Version 1
// Description
// Button that triggers a server action
```

### Click on BTN_SetDay (server)

```
LoadDateValue()
```

### Initializing of BTN_SetDayCombo (server)

```
// Version 1
// Description
// Button that triggers a server action
```

### Click on BTN_SetDayCombo (server)

```
dMinDate is Date = gDate

// Set minimum date. 6 days less than current date
dMinDate.Day -= 6
// Set maximum to include current date
gDate.Day        += 1

IF COMBO_ModuleInfo.Select() <> 1 THEN
    QRY_StaffModules.ParamModuleCode = COMBO_ModuleInfo[COMBO_ModuleInfo.Value]
    HExecuteQuery(QRY_StaffModules)
    HReadFirst(QRY_StaffModules)
    IF HFound(QRY_StaffModules) THEN
        STC_LecturerName = QRY_StaffModules.LecturerName
        STC_LecturerEmail = QRY_StaffModules.LecturerEmail
        STC_ModuleCode = QRY_StaffModules.ModuleCode
        STC_ModuleDescription = QRY_StaffModules.Description
    END

    // Student uploads
    QRY_WeeklyRpt_UniqueUploads.ParamModuleCode = COMBO_ModuleInfo[COMBO_ModuleInfo.Value]
    HExecuteQuery(QRY_WeeklyRpt_UniqueUploads)
    nStudents is int = HNbRec(QRY_WeeklyRpt_UniqueUploads)

    STC_ModuleStudentSubmissions.Value = "Number of students who submitted " + COMBO_ModuleInfo[COMBO_
    ModuleInfo.Value] + " exam:"
    STC_ModuleStudentSubmissionsValue = NumToString(nStudents)
    CELL_ModuleInfo.Visible = True

ELSE
    CELL_ModuleInfo.Visible = False
END
```

### Initializing of COMBO_ModuleInfo (server)

```
// Version 1
// Description
// Combo box for selecting a continent

// Initialize Listbox value
ListSelectPlus(MySelf,0)
```

### Add a token in EDT_ChangeDay (browser)

```
PROCEDURE AddToken (MyToken is Token)

//RETURN False to prevent from adding the token
RETURN True
```

### Click on a token of EDT_ChangeDay (browser)

```
PROCEDURE ClickToken (MyToken is Token)
```

### Delete a token in EDT_ChangeDay (browser)

```
PROCEDURE DeleteToken(MyToken is Token)

//RESULT False to prevent from deleting the token
RESULT True
```

**Click on BTN_Daily_Report (onclick browser event) ( POPUP_Combobox )**

```
// Functionality to implement as required
PopupClose()
```

**Click on BTN_Daily_Report ( POPUP_Combobox ) (server)**

```
PageDisplay(PAGE_Exam_Dashboard)
```

**Click on BTN_LecturerSummaryReport (onclick browser event) ( POPUP_Combobox )**

```
PopupClose()
```

**Click on BTN_LecturerSummaryReport ( POPUP_Combobox ) (server)**

```
PageDisplay(PAGE_Lecturer_SummaryReport)
```

**Click on BTN_StatisticalReport (onclick browser event) ( POPUP_Combobox )**

```
PopupClose()
```

**Click on BTN_StatisticalReport ( POPUP_Combobox ) (server)**

```
PageDisplay(PAGE_StatisticalReport)
```

**Click on BTN_WeeklyReport (onclick browser event) ( POPUP_Combobox )**

```
PopupClose()
```

**Click on BTN_WeeklyReport ( POPUP_Combobox ) (server)**

```
PageDisplay(PAGE_WeeklyReport)
```

# PAGE_WeeklyReport

## Procedures

**Local procedure LoadDateValue (server)**

```
PROCEDURE LoadDateValue()
```

```
gDate = EDT_ChangeDay
dMinDate is Date = gDate
nUniqueUploads    is int
nModulesCompleted is int = 0

// Set minimum date. 6 days less than current date
dMinDate.Day -= 6
// Set maximum to include current date
gDate.Day        += 1

// Hide Module information cell
CELL_ModuleInfo.Visible = False

// Clear Combo box and refill
COMBO_ModuleInfo.DeleteAll()
ListAdd(COMBO_ModuleInfo,"Select a module...")
ListSelectPlus(COMBO_ModuleInfo,1)

// Students who wrote
QRY_WeeklyRpt_UniqueUploads.ParamUploadTimeMin = dMinDate
QRY_WeeklyRpt_UniqueUploads.ParamUploadTimeMax = gDate
HExecuteQuery(QRY_WeeklyRpt_UniqueUploads)
nUniqueUploads = HNbRec(QRY_WeeklyRpt_UniqueUploads)

STC_TotalUniqueUploadsValue = NumToString(nUniqueUploads)

// Modules Completed
CurrentModuleCode is string = ""
QRY_WeeklyRpt_ModulesCompleted.ParamUploadTimeMin   = dMinDate
QRY_WeeklyRpt_ModulesCompleted.ParamUploadTimeMax   = gDate
HExecuteQuery(QRY_WeeklyRpt_ModulesCompleted)
FOR EACH QRY_WeeklyRpt_ModulesCompleted
    IF QRY_WeeklyRpt_ModulesCompleted.ModuleCode <> CurrentModuleCode THEN
        CurrentModuleCode = QRY_WeeklyRpt_ModulesCompleted.ModuleCode
        // Fill combobox
        ListAdd(COMBO_ModuleInfo,CurrentModuleCode)
        nModulesCompleted += 1
    END
END

STC_ModulesCompletedValue = NumToString(nModulesCompleted)

// Update Chart
grDeleteSeries(CHART_Uploads,1,grData)
numOfSubmissions is int
numOfDateCategory is int = 1
arrUploadDates is array of strings
tempDate is string

// Get all the dates where uploads took place and add them to the array
QRY_WeeklyRpt_Chart.ParamUploadTimeMax = gDate
QRY_WeeklyRpt_Chart.ParamUploadTimeMin = dMinDate
HExecuteQuery(QRY_WeeklyRpt_Chart)
FOR EACH QRY_WeeklyRpt_Chart
    tempDate = DateToString(QRY_WeeklyRpt_Chart.UploadTime, "YYYY-MM-DD")
    IF ArraySeek(arrUploadDates, asLinearFirst, tempDate) = -1 THEN
        ArrayAdd(arrUploadDates,tempDate)
    END
END

// For each date in the array, get the number of uploads on that day
```

```
FOR arrCount = 1 TO arrUploadDates.Count()
    QRY_WeeklyRpt_UniqueUploadsByDate.ParamUploadTime = StringToDate(arrUploadDates[arrCount],"YYYY/MM/DD"
    )
    HExecuteQuery(QRY_WeeklyRpt_UniqueUploadsByDate)
    numOfSubmissions = HNbRec(QRY_WeeklyRpt_UniqueUploadsByDate)

    // Add collected data to graph
    grCategoryLabel(CHART_Uploads,numOfDateCategory,arrUploadDates[arrCount])
    grAddData(CHART_Uploads,1,numOfSubmissions)
    numOfDateCategory += 1
END

// Draw chart
grDraw(CHART_Uploads)
```

# PAGE_StatisticalReport

## Code

**Global declarations of PAGE_StatisticalReport (server)**

```
PROCEDURE MyPage()
```

**Initializing of PAGE_StatisticalReport (server)**

```
RunStatistics()
```

# PAGE_StatisticalReport

## Control code

**Click on IMG_Logo (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    IF YesNo("You are currently logged in. Returning to Home page will Log you out. Continue?") = Yes THEN
        // End session and logout
        gnUserID          = 0
        gsSession             = ""
        gsUser                = ""
        gsUserType            = ""

        HCloseConnection(gConnection)
        PageDisplay(PAGE_Login)
    END
END
```

**Initializing of Link_Logout (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    Link_Logout..Visible = True
END
```

**Click on Link_Logout (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    // End session and logout
    gnUserID          = 0
    gsSession             = ""
    gsUser                = ""
    gsUserType            = ""
```

```
    HCloseConnection(gConnection)
    PageDisplay(PAGE_Login)
END
```

**Initializing of STC_WelcomeUser (server) (PAGETPL_Session template)**

```
IF Length(gnUserID) = 7 OR Length(gnUserID) = 8 THEN
    STC_WelcomeUser.Caption = "Welcome " + gsUser
ELSE
    STC_WelcomeUser.Caption = ""
END
```

**Initializing of MENU_Nav (server) (PAGETPL_Session template)**

```
MySelf.OPT_ExamView.Visible = False
MySelf.OPT_ExamScheduleUpload.Visible = False

IF gsUser = ExamDeptUser THEN
    MySelf.OPT_ExamScheduleUpload.Visible = True
END
IF gsUserType = "Lecturer" THEN
    MySelf.OPT_ExamView.Visible = True
END
```

**Select the ZONE_Menu.MENU_Nav.Home menu option ( MENU_Nav ) (server) (PAGETPL_Session template)**

```
IF gsUser = ExamDeptUser THEN
    PageDisplay(PAGE_Exam_Dashboard)
END
IF gsUserType = "Lecturer" THEN
     PageDisplay(PAGE_Lecturer_Dashboard)
END
IF gsUserType = "Student" THEN
    PageDisplay(PAGE_Student_Dashboard)
END
```

**Click on BTN_NoName1 (onclick browser event)**

```
PopupDisplay(POPUP_Combobox,popupFixed+popupDiscardable)
```

**Click on BTN_Daily_Report (onclick browser event)  ( POPUP_Combobox )**

```
// Functionality to implement as required
PopupClose()
```

**Click on BTN_Daily_Report ( POPUP_Combobox ) (server)**

```
PageDisplay(PAGE_Exam_Dashboard)
```

**Click on BTN_LecturerSummaryReport (onclick browser event)  ( POPUP_Combobox )**

```
PopupClose()
```

**Click on BTN_LecturerSummaryReport ( POPUP_Combobox ) (server)**

```
PageDisplay(PAGE_Lecturer_SummaryReport)
```

**Click on BTN_StatisticalReport (onclick browser event) ( POPUP_Combobox )**

```
PopupClose()
```

**Click on BTN_StatisticalReport ( POPUP_Combobox ) (server)**

```
PageDisplay(PAGE_StatisticalReport)
```

**Click on BTN_WeeklyReport (onclick browser event) ( POPUP_Combobox )**

```
PopupClose()
```

**Click on BTN_WeeklyReport ( POPUP_Combobox ) (server)**

```
PageDisplay(PAGE_WeeklyReport)
```

# PAGE_StatisticalReport

## Procedures

**Local procedure RunStatistics (server)**

```
// Summary: Calculate the average writing time per module to date and add it to the Chart
// Syntax:
//RunStatistics ()
//
// Parameters:
//  None
// Example:
// <Specify a usage example>
//
// Return value:
PROCEDURE RunStatistics()
nModuleCounter    is int       = 1

// Clear charts
CHART_ModuleWritingTimes.DeleteAll()

FOR EACH ModuleInfo
    StartTime, UploadTime, tDifference are DateTimes
    WriteTimes is array of reals
    timeInMin         is real
    nAverage     is real
    nMax     is real = 0
    nMin     is real = 0

    // Get all the ExamOutput records with module code as parameter
    QRY_ExamOutputByModuleCode.ParamModuleCode = ModuleInfo.ModuleCode
```

```
    HExecuteQuery(QRY_ExamOutputByModuleCode)
    FOR EACH QRY_ExamOutputByModuleCode
        StartTime = QRY_ExamOutputByModuleCode.StartTime
        UploadTime = QRY_ExamOutputByModuleCode.UploadTime
        tDifference = DateTimeDifference(StartTime,UploadTime)
        timeInMin = tDifference..Hour*60 + tDifference..Minute + tDifference..Second/60
        ArrayAdd(WriteTimes,timeInMin)

        // Check if time new max
        IF nMin = 0 AND nMax = 0 THEN
            nMin = timeInMin
            nMax = timeInMin
        ELSE
            IF timeInMin > nMax THEN
                nMax = timeInMin
            END
            IF timeInMin < nMin THEN
                nMin = timeInMin
            END
        END
    END

    // Calculate average
    nAverage = Mean(WriteTimes)

    // Add Average to Averages Chart
    grCategoryLabel(CHART_ModuleWritingTimes,nModuleCounter,ModuleInfo.ModuleCode)
    grAddData(CHART_ModuleWritingTimes,1,nModuleCounter,nAverage)

    // Add Minimum time to Chart
    grAddData(CHART_ModuleWritingTimes,2,nModuleCounter,nMax)
    // Add Maximum time to Chart
    grAddData(CHART_ModuleWritingTimes,3,nModuleCounter,nMin)

    nModuleCounter += 1
END

// Set legend properties
grLegend(CHART_ModuleWritingTimes, grAtTop)
grSeriesLabel(CHART_ModuleWritingTimes, 1, "Average time written")
grSeriesLabel(CHART_ModuleWritingTimes, 2, "Max time written")
grSeriesLabel(CHART_ModuleWritingTimes, 3, "Min time written")

grSeriesPointType(CHART_ModuleWritingTimes,1,grPointCircle)
grSeriesPointType(CHART_ModuleWritingTimes,2,grPointDiamond)
grSeriesPointType(CHART_ModuleWritingTimes,3,grPointTriangle)

// Build chart
grDraw(CHART_ModuleWritingTimes)
```

*Part 3*

# Query

# QRY_StaffModules

## Code

**SQL code of QRY_StaffModules**

```
SELECT
    ModuleLeader.ModuleCode AS ModuleCode,
    ExamSetup.DateExam AS DateExam,
    ModuleLeader.StaffNumber AS StaffNumber,
    ModuleInfo.Description AS Description,
    StaffInfo.Name AS LecturerName,
    StaffInfo.Email AS LecturerEmail
FROM
    ModuleLeader,
    ExamSetup,
    ModuleInfo,
    StaffInfo
WHERE
    ModuleLeader.ModuleCode = ExamSetup.ModuleCode
    AND
    ModuleInfo.ModuleCode = ModuleLeader.ModuleCode
    AND
    StaffInfo.StaffNumber = ModuleLeader.StaffNumber
    AND
    (
        ModuleLeader.StaffNumber = {ParamStaffNumber} AND
        ModuleLeader.ModuleCode = {ParamModuleCode}
    )
```

# QRY_DailyRpt_StudentsToWrite

## Code

**SQL code of QRY_DailyRpt_StudentsToWrite**

```
SELECT
    ExamSetup.DateExam AS DateExam,
    ExamSetup.ModuleCode AS ModuleCode,
    StudentModule.ModuleCode AS ModuleCode_St
FROM
    ExamSetup,
    ModuleInfo,
    StudentModule
WHERE
    ModuleInfo.ModuleCode = StudentModule.ModuleCode
    AND      ExamSetup.ModuleCode = ModuleInfo.ModuleCode
    AND
    (
        ExamSetup.DateExam = {ParamDateExam}
        AND  ExamSetup.ModuleCode = {ParamModuleCode}
    )
```

# QRY_DailyRpt_ModulesExpected

## Code

**SQL code of QRY_DailyRpt_ModulesExpected**

```
SELECT
    ExamSetup.DateExam AS DateExam,
    ExamSetup.ModuleCode AS ModuleCode
FROM
    ExamSetup
WHERE
    ExamSetup.DateExam = {ParamDateExam}
```

# QRY_DailyRpt_Chart

## Code

**SQL code of QRY_DailyRpt_Chart**

```
SELECT
    ExamOutput.UploadTime AS UploadTime,
    ExamOutput.ModuleCode AS ModuleCode
FROM
    ExamOutput
WHERE
    ExamOutput.UploadTime >= {ParamUploadTimeMin}
    AND ExamOutput.UploadTime < {ParamUploadTimeMax}
ORDER BY
    UploadTime ASC
```

# QRY_WeeklyRpt_Chart

## Code

**SQL code of QRY_WeeklyRpt_Chart**

```
SELECT
    ExamOutput.UploadTime AS UploadTime,
    ExamOutput.ModuleCode AS ModuleCode
FROM
    ExamOutput
WHERE
    ExamOutput.UploadTime BETWEEN {ParamUploadTimeMin} AND {ParamUploadTimeMax}
ORDER BY
    UploadTime ASC
```

# QRY_WeeklyRpt_ModulesCompleted

## Code

**SQL code of QRY_WeeklyRpt_ModulesCompleted**

```
SELECT
    ExamOutput.ModuleCode AS ModuleCode
FROM
    ExamOutput
WHERE
    ExamOutput.UploadTime >= {ParamUploadTimeMin} AND
    ExamOutput.UploadTime <= {ParamUploadTimeMax}
ORDER BY
    ModuleCode ASC
```

# QRY_WeeklyRpt_UniqueUploads

## Code

**SQL code of QRY_WeeklyRpt_UniqueUploads**

```
SELECT
    ExamOutput.ModuleCode AS ModuleCode
FROM
    ExamOutput
WHERE
    (
        ExamOutput.UploadTime >= {ParamUploadTimeMin}
        AND ExamOutput.UploadTime <= {ParamUploadTimeMax}
        AND ExamOutput.ModuleCode = {ParamModuleCode}
    )
ORDER BY
    ModuleCode ASC
```

# QRY_WeeklyRpt_UniqueUploadsByDate

## Code

**SQL code of QRY_WeeklyRpt_UniqueUploadsByDate**

```
SELECT
    ExamOutput.UploadTime AS UploadTime
FROM
    ExamOutput
WHERE
    ExamOutput.UploadTime LIKE {ParamUploadTime}%
```

# QRY_LecturerSummaryRpt_Chart

## Code

**SQL code of QRY_LecturerSummaryRpt_Chart**

```
SELECT
    ExamOutput.ModuleCode AS ModuleCode
FROM
    ExamOutput
ORDER BY
    ModuleCode ASC
```

# QRY_ExamOutputByModuleCode

## Code

**SQL code of QRY_ExamOutputByModuleCode**

```
SELECT
    ExamOutput.TransactionID AS TransactionID,
    ExamOutput.ModuleCode AS ModuleCode,
    ExamOutput.StartTime AS StartTime,
    ExamOutput.UploadTime AS UploadTime,
    ExamOutput.AnswerPaperPDF AS AnswerPaperPDF,
    ExamOutput.StudentNumber AS StudentNumber
FROM
    ExamOutput
WHERE
    ExamOutput.ModuleCode = {ParamModuleCode}
    AND ExamOutput.TransactionID = {ParamTransactionID}
```

# QRY_ExamModuleSubmissions

## Code

**SQL code of QRY_ExamModuleSubmissions**

```
SELECT
    ExamOutput.TransactionID AS TransactionID,
    ExamOutput.StartTime AS StartTime,
    ExamOutput.UploadTime AS UploadTime,
    ExamOutput.AnswerPaperPDF AS AnswerPaperPDF,
    ExamOutput.StudentNumber AS StudentNumber,
    ExamOutput.ModuleCode AS ModuleCode
FROM
    ExamOutput
WHERE
    ExamOutput.ModuleCode = {ParamModuleCode}
```

# QRY_StudentExams

## Code

**SQL code of QRY_StudentExams**

```
SELECT
    StudentModule.StudentNumber AS StudentNumber,
    ExamSetup.ModuleCode AS ModuleCode,
    ExamSetup.DateExam AS DateExam,
    ExamSetup.ExamPaperPDF AS ExamPaperPDF
FROM
    ExamSetup,
    ModuleInfo,
    StudentModule
WHERE
    ModuleInfo.ModuleCode = StudentModule.ModuleCode
    AND     ExamSetup.ModuleCode = ModuleInfo.ModuleCode
    AND
    (
        StudentModule.StudentNumber = {ParamStudentNumber}
        AND ExamSetup.ModuleCode = {ParamModuleCode}
        AND ExamSetup.DateExam = {ParamDateExam}
    )
ORDER BY
    ModuleCode ASC
```

# QRY_StudentExamResults

## Code

**SQL code of QRY_StudentExamResults**

```
SELECT
    ExamOutput.StudentNumber AS StudentNumber,
    ExamOutput.ModuleCode AS ModuleCode,
    ExamOutput.StartTime AS StartTime,
    ExamOutput.UploadTime AS UploadTime,
    ExamOutput.AnswerPaperPDF AS AnswerPaperPDF,
    StudentInfo.StudentName AS StudentName,
    StudentInfo.StudentEmail AS StudentEmail,
    ExamOutput.TransactionID AS TransactionID
FROM
    StudentInfo,
    StudentModule,
    ExamOutput
WHERE
    ExamOutput.StudentNumberModuleCode = StudentModule.StudentNumberModuleCode
    AND     StudentInfo.StudentNumber = StudentModule.StudentNumber
    AND
    (
        ExamOutput.ModuleCode = {ParamModuleCode}
        AND ExamOutput.StudentNumber = {ParamStudentNumber}
    )
```

*Part 4*

# Set of procedures

# ServerProcedures

## Code

### Global procedure ExamAnswerPaperPDFNaming (server)

```
// Summary: <specify the procedure action>
// Syntax:
//ExamAnswerPaperPDFNaming (<DataFile> is data source object)
//
// Parameters:
//  DataFile (data source object):
// Example:
// <Specify a usage example>
//
PROCEDURE ExamAnswerPaperPDFNaming(DataFile is Data Source)
AnswerPaper  is string
sDate        is string
sTime        is string

FOR EACH DataFile
    sDate                = DateToString(DataFile.DateExam,"YYYYMMDD")
    sTime                = TimeToString(DataFile.UploadTime,"HH:MM:SS")
    AnswerPaper          = DataFile.StudentNumber + "_" + DataFile.ModuleCode + "_EXAM_" + sDate + "-" +
    sTime + ".pdf"
    DataFile.AnswerPaperPDF    = AnswerPaper
    HSave(DataFile)
END
```

### Global procedure PDFExamBuilder (server)

```
PROCEDURE PDFExamBuilder()
// For each exam in exam output, create a pdf for test purposes

// Create exam file
pdfDoc is pdfDocument =
"E:\ICT3715_Project\Online examination file submission system\Exe\Test folders and
files\TestExamSubmissionFile.pdf"

sFile is string
FOR EACH ModuleInfo

    // Create directory
    IF fDirectoryExist(fDataDir() + fSep + "JRouter" + fSep + ModuleInfo.ModuleCode) = False THEN
        fMakeDir(fDataDir() + fSep + "JRouter" + fSep + ModuleInfo.ModuleCode)
    END

    QRY_ExamOutputByModuleCode.ParamModuleCode = ModuleInfo.ModuleCode
    HExecuteQuery(QRY_ExamOutputByModuleCode)
    FOR EACH QRY_ExamOutputByModuleCode
        sFile = fDataDir() + fSep + "JRouter" + fSep + ModuleInfo.ModuleCode + fSep + Replace(
        QRY_ExamOutputByModuleCode.AnswerPaperPDF,":","")
        IF fFileExist(sFile) = False THEN
            pdfDoc.Save(sFile)
```

```
        END
    END
END
```

### Global procedure PDFExamFileFolderBuilder (server)

```
PROCEDURE PDFExamFileFolderBuilder()
//For each module, create a exam pdf and add a module file to JRouter file

FOR EACH ModuleInfo
    // Create exam file
    pdfDoc is pdfDocument =
    "E:\ICT3715_Project\Online examination file submission system\Exe\Test folders and
    files\TestExamFile.pdf"


    pdfDoc.Save("E:\ICT3715_Project\Online examination file submission system\Exe\ExamFiles\" + ModuleInfo
    .ModuleCode + ".pdf")
END
```

### Global procedure SetPassword (server)

```
// Summary: <specify the procedure action>
// Syntax:
//[ <Result> = ] SetPassword ()
//
// Parameters:
//   None
// Example:
// <Specify a usage example>
//
// Return value:
PROCEDURE SetPassword() : string
password     is string     = ""
AlphaListUC      is string     = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
AlphaListLC      is string     = "abcdefghijklmnopqrstuvwxyz"
RandomChar is int

// First 3 characters lowercase
InitRandom()
LOOP (3)
    RandomChar = Random(1,26)
    password += AlphaListLC[[RandomChar]]
END

// Second 2 characters number between 10 and 99
InitRandom()
nRandomNumber is int
nRandomNumber = Random(10, 99)

password += NumToString(nRandomNumber)

// Last 3 characters Upper case
InitRandom()
LOOP (3)
    RandomChar = Random(1,26)
    password += AlphaListUC[[RandomChar]]
END
```

```
RETURN password
```

### Global procedure TransactionID (server)

```
// Summary: <specify the procedure action>
// Syntax:
//[ <Result> = ] TransactionID (<sStudentNumber> is string)
//
// Parameters:
//   sStudentNumber (ANSI string):
// Example:
// <Specify a usage example>
//
PROCEDURE TransactionID(sStudentNumber is string) : string
// Set TransactionID
AlphaList        is string    = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
TransactionID    is string    = ""

InitRandom()
LOOP (4)
    RandomChar is int = Random(1,54)
    TransactionID += AlphaList[[RandomChar]]
END
TransactionID += sStudentNumber[[1 TO 4]]
TransactionID += "-"

//   Initialize random number generator
InitRandom()
nRandomNumber is int
nRandomNumber = Random(100, 999)

TransactionID += NumToString(nRandomNumber)
TransactionID += NumToString(modulo(nRandomNumber,7))

RETURN TransactionID
```

### Global procedure UploadExamOutput (server)

```
// Summary: <specify the procedure action>
// Syntax:
// UploadExamOutput ()
//
// Parameters:
//   None
// Return value:
//   None
//
// Example:
// <Specify a usage example>
//
PROCEDURE UploadExamOutput(sPath)
// Import ExamOutput Data
// Variable declaration
nFileNum             is int
sLineRead                is string
arrLineArray         is array of strings

sTransactionID           is string
sDateAndTime         is string
```

```
tStartTime                is DateTime
tUploadTime               is DateTime
sAnswerPaperPDf           is string
sStudentNumber            is string
sModuleCode               is string

nFileNum = fOpen (sPath, foRead)
IF nFileNum <> -1 THEN
    // Read the 1st line
    sLineRead = fReadLine(nFileNum)

    // Check the end of file
    IF sLineRead <> EOT THEN
        // Skip first line
        sLineRead = fReadLine(nFileNum)
    END
    WHILE sLineRead <> EOT
        // Process the line
        // Put line values in a string array
        arrLineArray = StringSplit(sLineRead,";")

        // Retrieving the values of the array
        sTransactionID                      = arrLineArray[1]

        // Check if Record exists
        HReadSeekFirst(ExamOutput,TransactionID,sTransactionID)
        IF HFound() = True THEN   // Exists continue to new record
            sLineRead = fReadLine(nFileNum)
            CONTINUE
        END

        sAnswerPaperPDf                     = arrLineArray[16]
        sStudentNumber                      = arrLineArray[2]
        sModuleCode                         = arrLineArray[6]

        // Get and set StartTime
        sDateAndTime                        = arrLineArray[13]        // Remove characters : and -

        sDateAndTime                        = Replace(sDateAndTime,"-","")
        sDateAndTime                        = Replace(sDateAndTime,":","")
        tStartTime                          = sDateAndTime

        // Get and set UploadTime
        sDateAndTime                        = arrLineArray[14]
        // Remove characters : and -
        sDateAndTime                        = Replace(sDateAndTime,"-","")
        sDateAndTime                        = Replace(sDateAndTime,":","")
        tUploadTime                         = sDateAndTime

        // Add to ExamOutput
        ExamOutput.TransactionID            = sTransactionID
        ExamOutput.StartTime                = tStartTime
        ExamOutput.UploadTime               = tUploadTime
        ExamOutput.AnswerPaperPDF           = sAnswerPaperPDf
        ExamOutput.StudentNumber            = sStudentNumber
        ExamOutput.ModuleCode               = sModuleCode
        HAdd(ExamOutput)

        // Read the next line
        sLineRead = fReadLine(nFileNum)
    END
```

```
    fClose(nFileNum)
END
```

### Global procedure UploadExamSchedule (server)

```
PROCEDURE UploadExamSchedule(sPath)
// Import StaffInfo Data
// Variable declaration
nFileNum              is int
sLineRead                 is string
arrLineArray          is array of strings

sModuleCode               is string
sDate                     is string
tDateExam                 is Date
sExamPaperPDF             is string

sNotSaved is array of strings

nFileNum = fOpen (sPath, foRead)
IF nFileNum <> -1 THEN
    // Read the 1st line
    sLineRead = fReadLine(nFileNum)

    // Check the end of file
    IF sLineRead <> EOT THEN
        // Skip first line
        sLineRead = fReadLine(nFileNum)
    END
    WHILE sLineRead <> EOT
        // Process the line
        // Put line values in a string array
        arrLineArray                          = StringSplit(sLineRead,";")

        // Retrieving the values of the array
        sModuleCode                           = arrLineArray[1]
        // Make sure ModCode length is 7 and uppercase
        sModuleCode = StringFormat(sModuleCode,ccUpCase)
        IF Length(sModuleCode) <> 7 THEN
            sLineRead = fReadLine(nFileNum)
            ArrayAdd(sNotSaved,sModuleCode)
            CONTINUE
        END

        // Get and set StartTime
        sDate                                 = arrLineArray[2]     // Remove character -
        sDate                                 = Replace(sDate,"/","")
        tDateExam                             = sDate

        // Else get rest of data
        sExamPaperPDF                         = arrLineArray[3]
        // Validate ExamPaperPDF
        IF sExamPaperPDF <> (sModuleCode + ".pdf") THEN
            sExamPaperPDF = sModuleCode + ".pdf"
        END

        // Modify or add to ExamSetup
        // HSave does both
        ExamSetup.ModuleCode                  = sModuleCode
```

```
        ExamSetup.DateExam                          = tDateExam
        ExamSetup.ExamPaperPDF                      = sExamPaperPDF
        HSave(ExamSetup)

        // Read the next line
        sLineRead = fReadLine(nFileNum)
    END
    fClose(nFileNum)
END

// If any lines was not saved, display to user.
IF sNotSaved.Count() > 0 THEN
    sModules is string = "The following modules were not saved:" + CR
    FOR i = 1 TO sNotSaved.Count()
        sModules += CR + sNotSaved[i]
    END
    sModules += CR + CR + "Please review data document for errors."

    Error(sModules)
END
```

**Global procedure UploadExamSetup (server)**

```
// Summary: <specify the procedure action>
// Syntax:
// UploadExamSetup ()
//
// Parameters:
//  None
// Return value:
//  None
//
// Example:
// <Specify a usage example>
//
PROCEDURE UploadExamSetup(sPath)
// Import StaffInfo Data
// Variable declaration
nFileNum              is int
sLineRead                 is string
arrLineArray          is array of strings

sModuleCode           is string
sDate                 is string
tDateExam             is Date
sExamPaperPDF         is string

nFileNum = fOpen (sPath, foRead)
IF nFileNum <> -1 THEN
    // Read the 1st line
    sLineRead = fReadLine(nFileNum)

    // Check the end of file
    IF sLineRead <> EOT THEN
        // Skip first line
        sLineRead = fReadLine(nFileNum)
    END
    WHILE sLineRead <> EOT
        // Process the line
        // Put line values in a string array
        arrLineArray                        = StringSplit(sLineRead,";")
```

```
        // Retrieving the values of the array
        sModuleCode                             = arrLineArray[6]

        // Check if Record exists
        HReadSeekFirst(ExamSetup,ModuleCode,sModuleCode)
        IF HFound() = True THEN   // Exists continue to new record
            sLineRead = fReadLine(nFileNum)
            CONTINUE
        END

        // Get and set StartTime
        sDate                                   = arrLineArray[13]        // Remove character -
        sDate                                   = Replace(sDate,"-","")
        tDateExam                               = sDate

        // Else get rest of data
        sExamPaperPDF                               = arrLineArray[15]

        // Add to ExamSetup
        ExamSetup.ModuleCode                    = sModuleCode
        ExamSetup.DateExam                      = tDateExam
        ExamSetup.ExamPaperPDF                      = sExamPaperPDF
        HAdd(ExamSetup)

        // Read the next line
        sLineRead = fReadLine(nFileNum)
    END
    fClose(nFileNum)
END
```

**Global procedure UploadModuleInfo (server)**

```
// Summary: <specify the procedure action>
// Syntax:
// UploadModuleInfo ()
//
// Parameters:
//   None
// Return value:
//   None
//
// Example:
// <Specify a usage example>
//
PROCEDURE UploadModuleInfo(sPath)
// Import StaffInfo Data
// Variable declaration
nFileNum              is int
sLineRead                 is string
arrLineArray          is array of strings

sModuleCode               is string
sDescription          is string

nFileNum = fOpen (sPath, foRead)
IF nFileNum <> -1 THEN
    // Read the 1st line
    sLineRead = fReadLine(nFileNum)
```

```
    // Check the end of file
    IF sLineRead <> EOT THEN
        // Skip first line
        sLineRead = fReadLine(nFileNum)
    END
    WHILE sLineRead <> EOT
        // Process the line
        // Put line values in a string array
        arrLineArray                        = StringSplit(sLineRead,";")

        // Retrieving the values of the array
        sModuleCode                         = arrLineArray[6]

        // Check if Record exists
        HReadSeekFirst(ModuleInfo,ModuleCode,sModuleCode)
        IF HFound() = True THEN   // Exists continue to new record
            sLineRead = fReadLine(nFileNum)
            CONTINUE
        END

        // Else get rest of data
        sDescription                        = arrLineArray[7]

        // Add to ModuleInfo
        ModuleInfo.ModuleCode               = sModuleCode
        ModuleInfo.Description                  = sDescription
        HAdd(ModuleInfo)

        // Read the next line
        sLineRead = fReadLine(nFileNum)
    END
    fClose(nFileNum)
END
```

**Global procedure UploadModuleLeader (server)**

```
// Summary: <specify the procedure action>
// Syntax:
// UploadModuleLeader ()
//
// Parameters:
//  None
// Return value:
//  None
//
// Example:
// <Specify a usage example>
//
PROCEDURE UploadModuleLeader(sPath)
// Import StaffInfo Data
// Variable declaration
nFileNum            is int
sLineRead               is string
arrLineArray        is array of strings

sStaffNumber        is string
sModuleCode             is string
```

```
nFileNum = fOpen (sPath, foRead)
IF nFileNum <> -1 THEN
    // Read the 1st line
    sLineRead = fReadLine(nFileNum)

    // Check the end of file
    IF sLineRead <> EOT THEN
        // Skip first line
        sLineRead = fReadLine(nFileNum)
    END
    WHILE sLineRead <> EOT
        // Process the line
        // Put line values in a string array
        arrLineArray                        = StringSplit(sLineRead,";")

        // Retrieving the values of the array
        sModuleCode                           = arrLineArray[6]
        sStaffNumber                        = arrLineArray[9]

        // Check if Record exists
        HReadSeekFirst(ModuleLeader,ModuleCodeStaffNumber,sModuleCode+sStaffNumber)
        IF HFound() = True THEN   // Exists continue to new record
            sLineRead = fReadLine(nFileNum)
            CONTINUE
        END

        // Add to ModuleLeader
        ModuleLeader.StaffNumber            = sStaffNumber
        ModuleLeader.ModuleCode             = sModuleCode
        HAdd(ModuleLeader)

        // Read the next line
        sLineRead = fReadLine(nFileNum)
    END
    fClose(nFileNum)
END
```

### Global procedure UploadStaffInfo (server)

```
// Summary: <specify the procedure action>
// Syntax:
//UploadStaffInfo (<sPath>)
//
// Parameters:
//   sPath:
// Example:
// <Specify a usage example>
//
PROCEDURE UploadStaffInfo(sPath)
// Import StaffInfo Data
// Variable declaration
nFileNum            is int
sLineRead               is string
arrLineArray        is array of strings

sStaffNumber        is string
sName                   is string
sEmail                  is string
sPassword               is string
```

```
nFileNum = fOpen (sPath, foRead)
IF nFileNum <> -1 THEN
    // Read the 1st line
    sLineRead = fReadLine(nFileNum)

    // Check the end of file
    IF sLineRead <> EOT THEN
        // Skip first line
        sLineRead = fReadLine(nFileNum)
    END
    WHILE sLineRead <> EOT
        // Process the line
        // Put line values in a string array
        arrLineArray = StringSplit(sLineRead,";")

        // Retrieving the values of the array
        sStaffNumber                    = arrLineArray[9]

        // Check if Record exists
        HReadSeekFirst(StaffInfo,StaffNumber,sStaffNumber)
        IF HFound() = True THEN   // Exists continue to new record
            sLineRead = fReadLine(nFileNum)
            CONTINUE
        END

        // Else get rest of data
        sName                           = arrLineArray[10]
        sEmail                          = arrLineArray[11]
        sPassword                       = arrLineArray[12]

        // Add to StaffInfo
        StaffInfo.StaffNumber           = sStaffNumber
        StaffInfo.Name                  = sName
        StaffInfo.Email                 = sEmail
        StaffInfo.Password              = sPassword
        HAdd(StaffInfo)

        // Read the next line
        sLineRead = fReadLine(nFileNum)
    END
    fClose(nFileNum)
END
```

### Global procedure UploadStudentInfo (server)

```
// Summary: <specify the procedure action>
// Syntax:
//UploadStudentInfo (<sPath>)
//
// Parameters:
//  sPath:
// Example:
// <Specify a usage example>
//
PROCEDURE UploadStudentInfo(sPath)
// Import StaffInfo Data
// Variable declaration
nFileNum            is int
sLineRead               is string
arrLineArray        is array of strings
```

```
sStudentNumber        is string
sStudentName      is string
sStudentPassword  is string
sStudentEmail         is string

nFileNum = fOpen (sPath, foRead)
IF nFileNum <> -1 THEN
    // Read the 1st line
    sLineRead = fReadLine(nFileNum)

    // Check the end of file
    IF sLineRead <> EOT THEN
        // Skip first line
        sLineRead = fReadLine(nFileNum)
    END
    WHILE sLineRead <> EOT
        // Process the line
        // Put line values in a string array
        arrLineArray                    = StringSplit(sLineRead,";")

        // Retrieving the values of the array
        sStudentNumber                  = arrLineArray[2]

        // Check if Record exists
        HReadSeekFirst(StudentInfo,StudentNumber,sStudentNumber)
        IF HFound() = True THEN   // Exists continue to new record
            sLineRead = fReadLine(nFileNum)
            CONTINUE
        END

        // Else get rest of data
        sStudentName                    = arrLineArray[3]
        sStudentPassword                = arrLineArray[5]
        sStudentEmail                       = arrLineArray[4]

        // Add to StudentInfo
        StudentInfo.StudentNumber           = sStudentNumber
        StudentInfo.StudentName             = sStudentName
        StudentInfo.StudentPassword         = sStudentPassword
        StudentInfo.StudentEmail            = sStudentEmail
        HAdd(StudentInfo)

        // Read the next line
        sLineRead = fReadLine(nFileNum)
    END
    fClose(nFileNum)
END
```

### Global procedure UploadStudentModule (server)

```
// Summary: <specify the procedure action>
// Syntax:
// UploadStudentModule ()
//
// Parameters:
//   None
// Return value:
//   None
//
// Example:
```

```
// <Specify a usage example>
//
PROCEDURE UploadStudentModule(sPath)
// Import StaffInfo Data
// Variable declaration
nFileNum              is int
sLineRead                 is string
arrLineArray          is array of strings

sModuleCode           is string
sStudentNumber        is string

nFileNum = fOpen (sPath, foRead)
IF nFileNum <> -1 THEN
    // Read the 1st line
    sLineRead = fReadLine(nFileNum)

    // Check the end of file
    IF sLineRead <> EOT THEN
        // Skip first line
        sLineRead = fReadLine(nFileNum)
    END
    WHILE sLineRead <> EOT
        // Process the line
        // Put line values in a string array
        arrLineArray                          = StringSplit(sLineRead,";")

        // Retrieving the values of the array
        sStudentNumber                        = arrLineArray[2]
        sModuleCode                           = arrLineArray[6]

        // Check if Record exists
        HReadSeekFirst(StudentModule,StudentNumberModuleCode,sStudentNumber+sModuleCode)
        IF HFound() = True THEN   // Exists continue to new record
            sLineRead = fReadLine(nFileNum)
            CONTINUE
        END

        // Add to StudentModule
        StudentModule.StudentNumber           = sStudentNumber
        StudentModule.ModuleCode              = sModuleCode
        HAdd(StudentModule)

        // Read the next line
        sLineRead = fReadLine(nFileNum)
    END
    fClose(nFileNum)
END
```

# AutomaticServerProcedures

## Code

### Global procedure DayAfterExam (server)

```
// Automatic procedure:
// The procedure is automatically run, after the project initialization code
// It will be repeated in a loop
//

/*
Automatic Procedures
https://doc.windev.com/en-US/?1000019455
• Get Day
• Check if its day after an exam
• Notify lecturer if exams found and exam 1 day after
• Run procedure on JRouter Module file and delete ekstra files, only keep new submissions
*/
PROCEDURE DayAfterExam()
sFile, sDirPath, sFileList is string
dDate is Date = Today()
dDate.Day -= 1

FOR EACH ModuleInfo
    // Check if its day after exam date
    HReadSeekFirst(ExamSetup,ExamSetup.DateExam,dDate)
    IF HFound() = False THEN
        CONTINUE
    END

    // Create directory
    sDirPath = fDataDir() + fSep + "JRouter" + fSep + ModuleInfo.ModuleCode
    IF fDirectoryExist(sDirPath) = False THEN
        fMakeDir(sDirPath)
    END

    sFileList = fListFile(sDirPath + fSep() + "*.pdf")

    QRY_ExamOutputByModuleCode.ParamModuleCode = ModuleInfo.ModuleCode
    HExecuteQuery(QRY_ExamOutputByModuleCode)
    FOR EACH QRY_ExamOutputByModuleCode
        // For each file in folder
        FOR EACH STRING sFile OF sFileList SEPARATED BY CR
            // If file starts with saved file details "STUDNUM_MODCODE_EXAM_"
            IF sFile.StartsWith(QRY_ExamOutputByModuleCode.StudentNumber + "_" +
            QRY_ExamOutputByModuleCode.ModuleCode + "_EXAM_") = True THEN
                // If file is not equal to stored pdf file name, delete
                IF QRY_ExamOutputByModuleCode.AnswerPaperPDF <> sFile THEN
                    fDelete(sDirPath + fSep() + sFile)
                END
            END
        END
    END
    // Email Lecturer to check
    EmailLecturerDayAfter(ModuleInfo.ModuleCode)
```

```
END
```

### Global procedure EmailLecturerDayAfter (server)

```
PROCEDURE EmailLecturerDayAfter(sModCode is string) : string

// Start the SMTP session
EmailSMTPSession is emailSMTPSession
EmailSMTPSession.ServerAddress = SERVERADDRESS
EmailSMTPSession.Name = SERVEREMAIL
EmailSMTPSession.Password = SERVERPASSWORD
EmailSMTPSession.Port = SERVERPORT

// Starts the SMTP session
IF EmailSMTPSession.StartSession() = False THEN
    // Failure starting the session
    RETURN "Failure starting the session"
END

// Get module code leader email details
QRY_StaffModules.ParamModuleCode = sModCode
HExecuteQuery(QRY_StaffModules)
HReadFirst(QRY_StaffModules)
IF HFound(QRY_StaffModules) = False THEN
    RETURN "Failure getting lecturer information"
END

eEmail is Email
eEmail.Sender = SERVEREMAIL
eEmail.Subject        = "Examination for " + sModCode + " results"
eEmail.Message        = "Examination for " + sModCode + " has completed." + CR +
"Please access the UNISA portal to view the submitted exams"

// Adds a recipient
Add(eEmail.Recipient, QRY_StaffModules.LecturerEmail)
Add(eEmail.Recipient, SERVERTESTEMAIL)

// Send in Outlook
//For information: a name found in the address book can be used as recipient
IF EmailSMTPSession.SendMessage(eEmail) = False THEN
    // Close the session
    EmailSMTPSession.CloseSession()
    RETURN "An error occurred while trying to send the email"
ELSE
    // Close the session
    EmailSMTPSession.CloseSession()
    RETURN "Email successfully sent"
END
```

*Part 5*

# Table of contents

# Project Online examination file submission