STaff Ad Resource MANagement

(STARMAN)

Design Document

Robert Gerard Holohan

C00003778

Table Of Contents

Table Of Figures

Table of Tables

1 Introduction

With the regular movement of staff within functional areas in South-East Technological University Carlow Campus (SETUCC), the need to ensure people have timely access to IT resources becomes more important.

At present, there is a significant dependence on the provision of services from the existing IT support functions. Notwithstanding the existence of a formal support ticketing system, requests for the provision and removal of resource access do not always reach the right ears in a timely manner.

Enabling functional areas to have a level of self-sufficiency can lead to an improvement in end-user experiences. Ensuring people have the resources they need when they need them, and, in the world of GDPR, not having access to resources they do not need, is most important.

This system is a web application that allows specific end-users request or remove access to certain of their own IT resources from their managed staff. Without the need for specific security rights or skills, an end-user can use this system to control access to their IT resources from one central place.

The primary resources that are managed through this application are file storage.

This document contains technical details of the system and how it will function. Included are all relevant architectures and data storage solutions.

2 Application Architecture

This application consists of four distinct elements:

- The front-end application, facilitating the submission and monitoring of access control requests by authenticated users
- The back-office application that monitors and processes requests and notifies the end-user of successful outcomes
- An SQL database to store all system-related data
- A series of web services to support the operation of both the front-end and back office systems

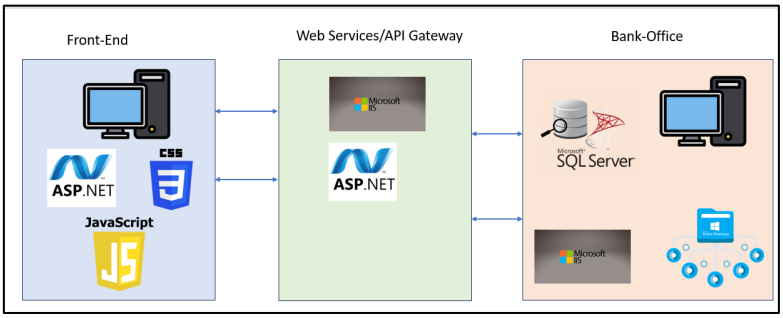


Figure 1: Technology Stack

2.1 Front-end application

The front-end application will be a web-based system, developed using Microsoft C# and the .NET ASPX tool sets.

The application will consist of one main form with an Ajax Tab Container to allow the extension of the perceived screen area. Each tab will support a different function of the system.

The web application will run on a Microsoft Internet Information Services (MIIS) platform. The site will use HTTPS protocols, supported by a commercial 2048-bit certificate.

The front-end will use a Microsoft SQL back-end database to store all relevant information. This database will not only store requests from the end-user but it will also be used to store system parameters, thus supporting the tailoring of the system without the need for software development skills. This will simplify elements of both the support and the growth of functionality of the system over time.

All communications between the front-end and the back-end database server will be made via a web service API. Requests will be secured using session tokens to increase security. All SQL statements will be parameterised to reduce the risk of SQL injection attacks on the system.

User interaction with the system will be through mouse-click selections from lists. There will be no free-text input fields on the forms, thus reducing the risk of security breaches.

Access to the website will be restricted to specific IP address ranges. The server will also only be available within the SETU network infrastructure (either actual or virtual machines).

2.2 Back-office application

The back-office application will be a console application developed using Microsoft C#. The application will run on a iterative loop, carrying out a series of tasks to include: -

- Add users to AD groups;
- Remove users from AD groups;
- Notify requesting end-user of successful outcome of each request;

The back-office application will run a series of parameterised SQL queries against the database, using the web service to execute same.

One query will run to identify incomplete requests. These requests will be processed in turn, with requests being submitted to the web service for completion. The returned result of the request will be analysed and the relevant database record updated to reflect successful completion or an explanation for the failure (based on the returned response).

Another query will identify completed requests for email notification. The query will group records by end-user. This will facilitate a summary email notification being sent in the event of more than one request existing.

The email request will be processed via the web service and a response returned. The returned result of the request will be analysed and the relevant database record updated to reflect successful completion or an explanation for the failure (based on the returned response).

All communications between the back-office application and the back-end database server will be made via a web service API. Requests will be secured using session tokens to increase security. All SQL statements will be parameterised to reduce the risk of SQL injection attacks on the system.

2.3 Web services

A set of secure web services will provide the underlying functionality to the front-end and back-office systems. The web services will be developed using Microsoft C# and will reside on a MIIS platform. The site will be secured using a commercial 2048-bit certificate. Access to the site will be restricted and access to each web service will be controlled using session tokens. When a user logs into the system, a unique session token will be generated. This token will be passed with each request from the front-end to the web service. Invalid session tokens will result in requests being ignored. No explanation will be given for requests that fail due to invalid session tokens.

The web service will interact with the SQL database server, the AD infrastructure and local mail server(s) as required.

By using a web service, there is no need for the end-user to have any direct access to either the SQL server or the AD infrastructure.

Access to the web services will be restricted to the application web server only via IP address restrictions through MIIS.

2.4 Back-end database server

A Microsoft SQL server will be used to host the application database. This database will store all data related to end-user requests. The database will also store an audit trail of all activities carried out in the system.

The database will store details of each end-user and the resources they own. These lists will be maintained by technical staff familiar with the operation of the system and the managed staff.

To support ease of maintenance and functionality improvements, system parameters will also be stored in the database where possible. A secured and restricted access interface will enable technical staff to maintain the underlying data sets as required.

3 Design Pattern

The application is designed to be a multi-user web-based tool with database and other services provided in the background. As such, a client-server style architecture would be most appropriate.

As there are 4 distinct different elements to this application, there are different elements of design to be considered. The web application used by the end-user is based on client-server designs. As an end-user interacts with the application, all processing takes place on the web server, with the results being displayed to the end-user via HTML.

The back-office application interacts with the web service platform to deliver functionality. In this case, the back-office application is a client and the web service acts in the role of server.

In both the front-end and back-office iterations, the database server acts as a server by processing SQL requests and delivering the resulting data to the requestor.

4 UI/UX Design

This application consists of two main components. The end-user front end and the back-office application.

The front-end has two main interfaces: -

- The end-user web interface where the operator interacts with the system through a series of web controls and buttons
- The administrator web interface where system administrators manage the underlying supporting data sets and system parameters

The back-office is a console application with a limited interface. As the application completes an operation, details will be displayed to the screen. These will include details of the iteration being completed and the success/failure of each request being processed.

4.1 End-user interface

As indicated above, the web front-end has two distinct parts. The first part, referred to from here on as the end-use interface, is the interface for all standard day-to-day usage of the system.

4.1.1 Main form

Once a user is authenticated to use the STARMAN system (Azure AD Single Sign-On and local AD group membership), the first screen they see displays a list of all users that are managed by the operator.

Figure 2 is a sample screenshot of what the form will look like.

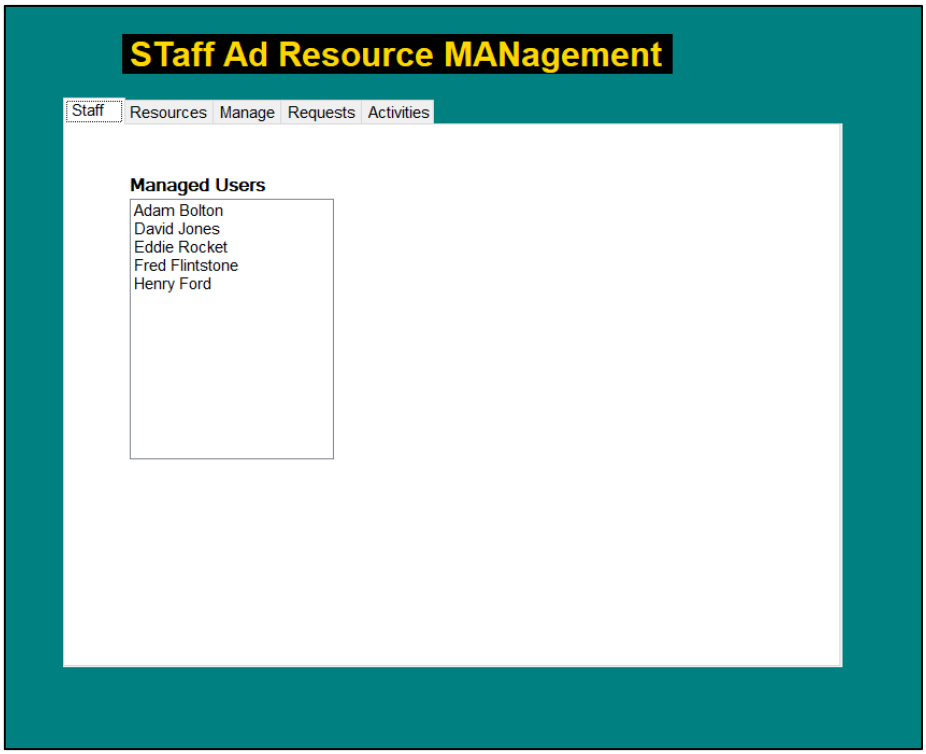


Figure 2: Main screen after login

4.1.2 View resources assigned to a user

When the operator clicks on a user in the list of Managed Users, a list of the resources assigned to the selected user will be displayed (Figure 3). This list is for reference purposes only and allows the operator to determine if they need to make changes to the user's assignments.

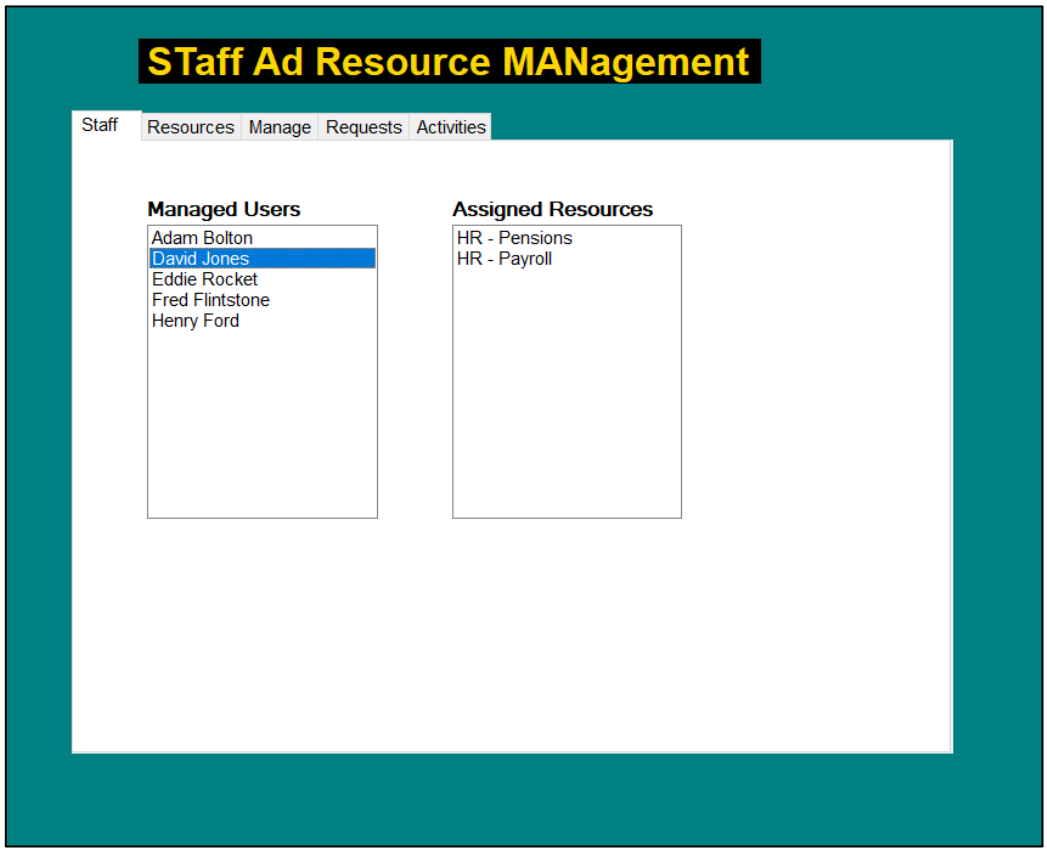


Figure 3: Display resources assigned to the selected user

4.1.3 Viewing managed resources

The ***Resources*** tab on the form bring up a list of resources that the operator manages (Figure 4). This list is in alphabetical order.

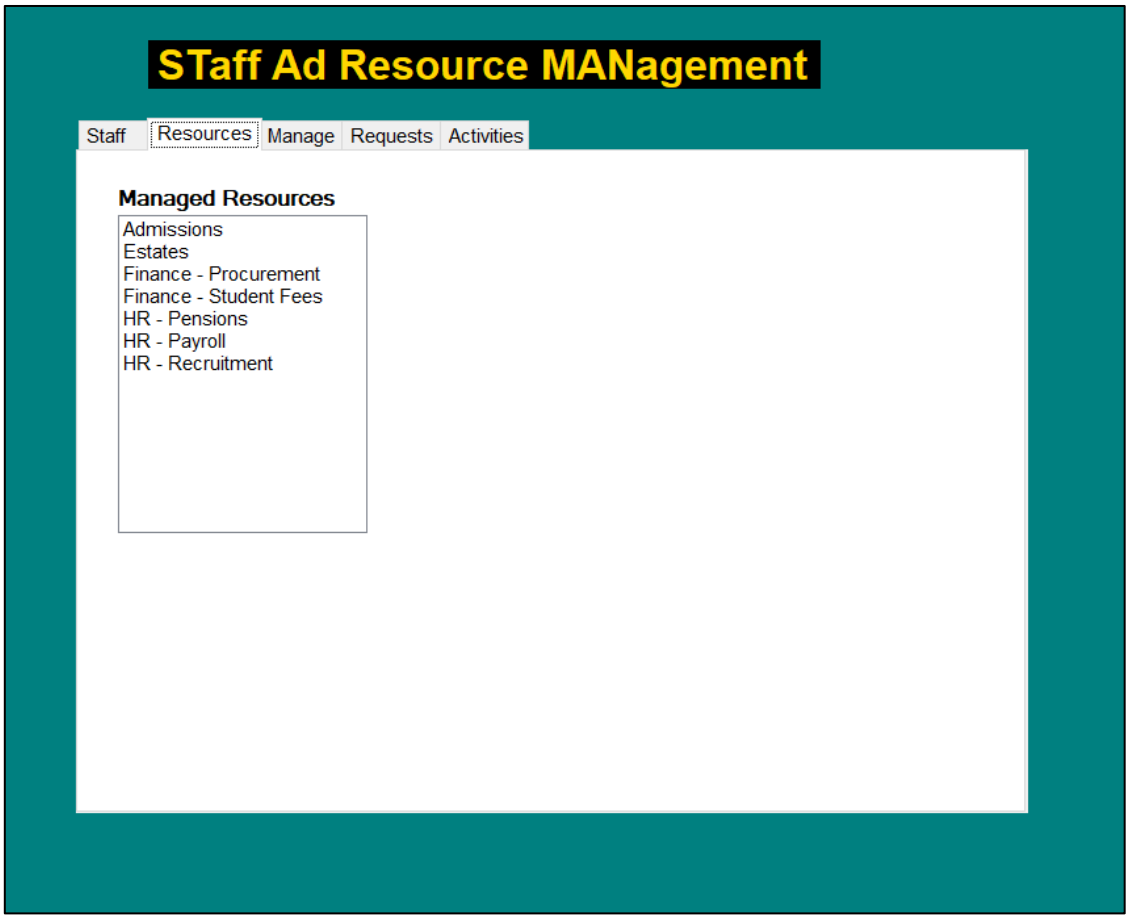


Figure 4: View managed resources

4.1.4 *Viewing who is assigned a resource*

When viewing the list of resources on the ***Resources*** tab, the operator can click on a specific resource to see who is assigned access (Figure 5). This is for reference purposes only.

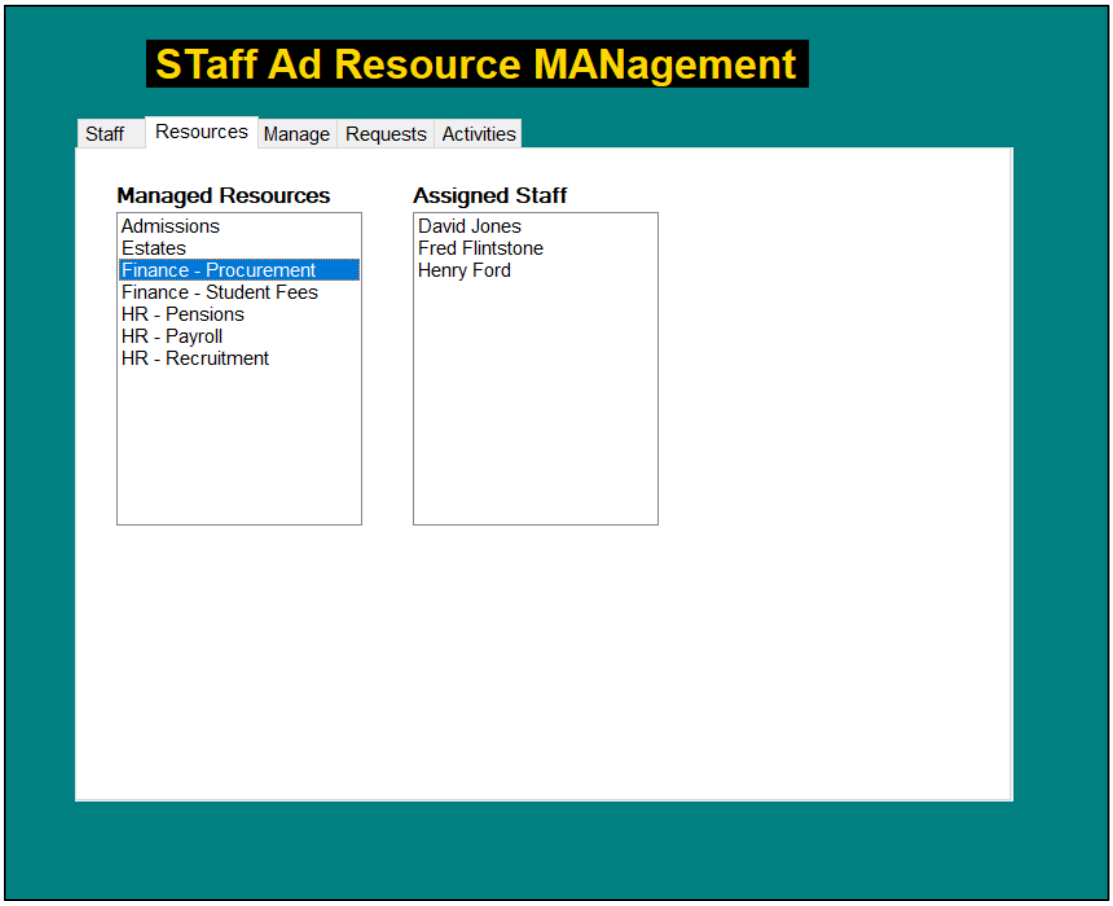


Figure 5: Show staff assigned to a resource

4.1.5 Assign resources to a user

Access to a resource for a user can be added or removed. Click on the ***Manage*** tab bring up the screen to do this (Figure 6).

To assign or revoke a resource, the operator selects a user in the list. Next, the operator selects one or more resources from the list (ctrl-click to select multiple resources).

The request is submitted by clicking the **Add** or **Remove** button as appropriate.

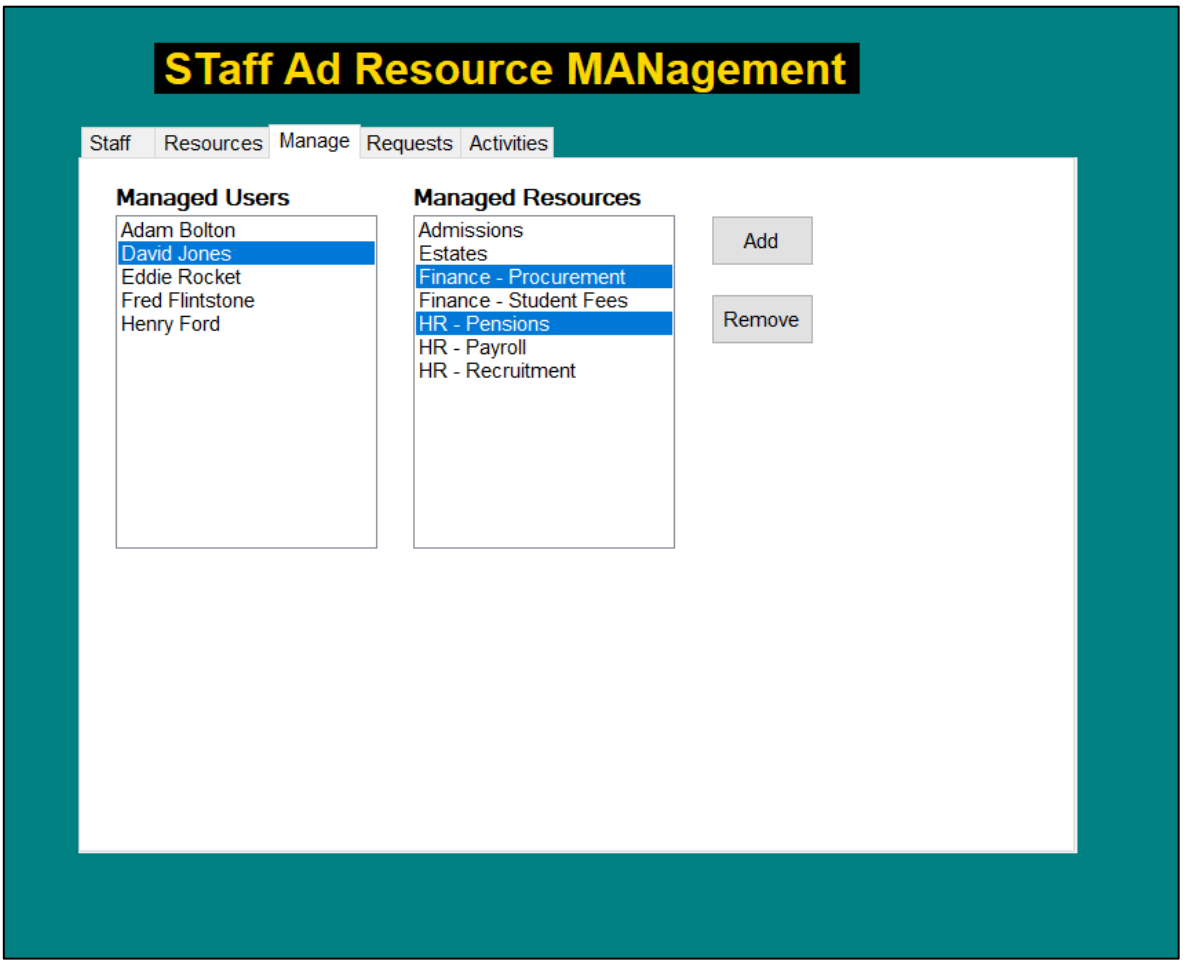


Figure 6: Assign one or more resources to a user

4.1.6 Reviewing requests

At any time, the operator can review a list of requests they have submitted to date.

Clicking on the ***Requests*** tab brings up a list of requests in date order (Figure 7). Details of the request and whether is has been completed or not are displayed.

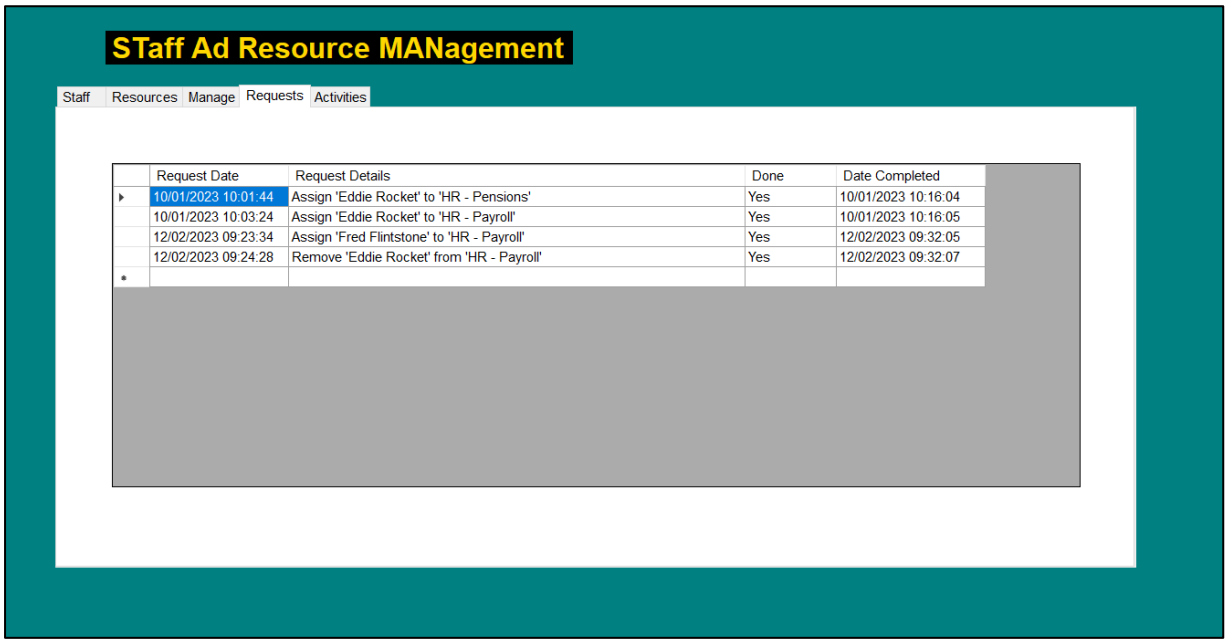


Figure 7: Review requests to date

4.1.7 Reviewing activities

At any time, the operator can view a list of their activities to date (Figure 8). This list is in date order and displays the date and time of the activity and a description of the activity itself.

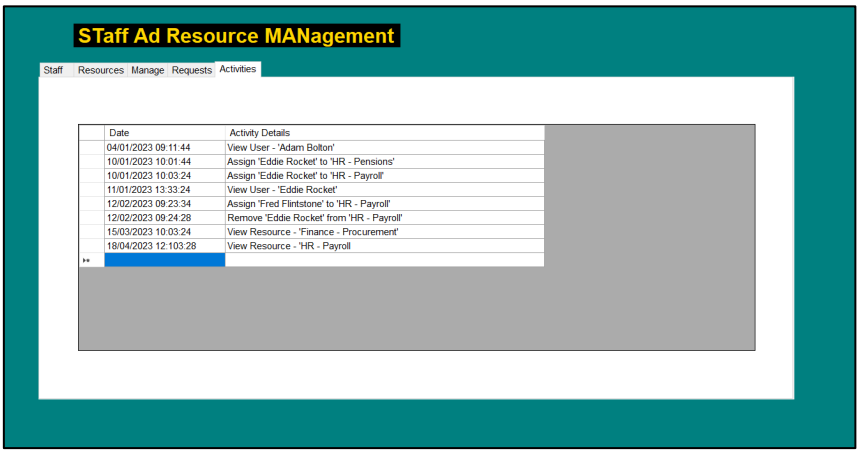


Figure 8: Review Activities to date

4.2 Administrator front-end

As outlined earlier, the software system will require a level of administration from time to time. The following are the provisional designs from the Administrator interface.

4.2.1 Staff administration

This screen has two main features. Firstly, it allows the administrator to add or remove operators from the system. Adding an operator is done by selecting a user from the tree populated from AD. Removing an operator involves selecting the operator from the list and clicking the **Remove** button.

Removing an operator will normally only happen when the operator change's role. In this case, the operator will be disabled within the system and any roles they manage will be retained.

Secondly, the administrator will be able to manage the list of staff managed by an operator. To do so, the administrator will select the relevant operator and then use the **Add/Remove Staff** buttons to carry out the relevant operations.

Figure 9 shows the Staff Administration interface for STARMAN.

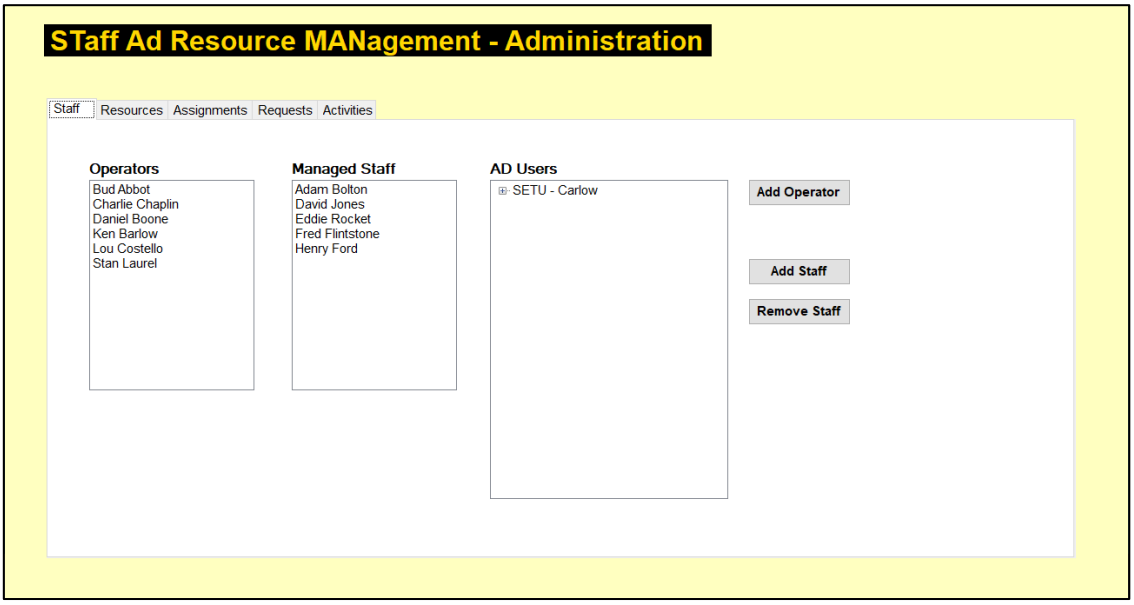


Figure 9: Staff administration interface

4.2.2 Resource administration interface

This interface allows the administrator to manage all resources within STARMAN. A list of existing resources is displayed and additional resources can be selected from the tree. To remove a resource, the administrator selects the resource from the ***Managed Resources*** list and clicks the **Remove Resource** button.

Figure 10 shows the Resource Administration interface.

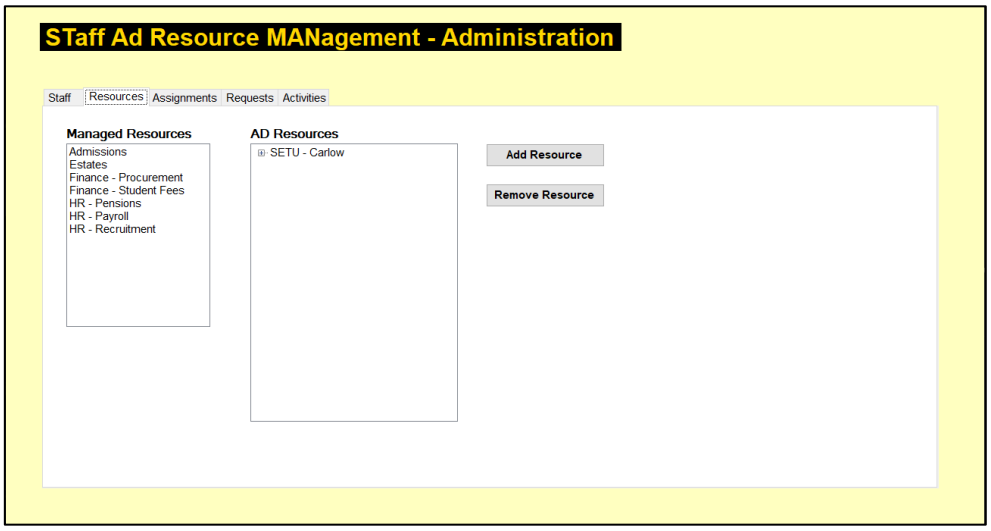


Figure 10: Resource Administration interface

4.2.3 Operator/Resource Assignment Administration interface

This interface provides the functionality to enable the operator to maintain the resources each operator manages (Figure 11). The administrator selects an operator from the list. The resources managed by the operator are listed. The administrator can then select a resource from the list and release if from the operator.

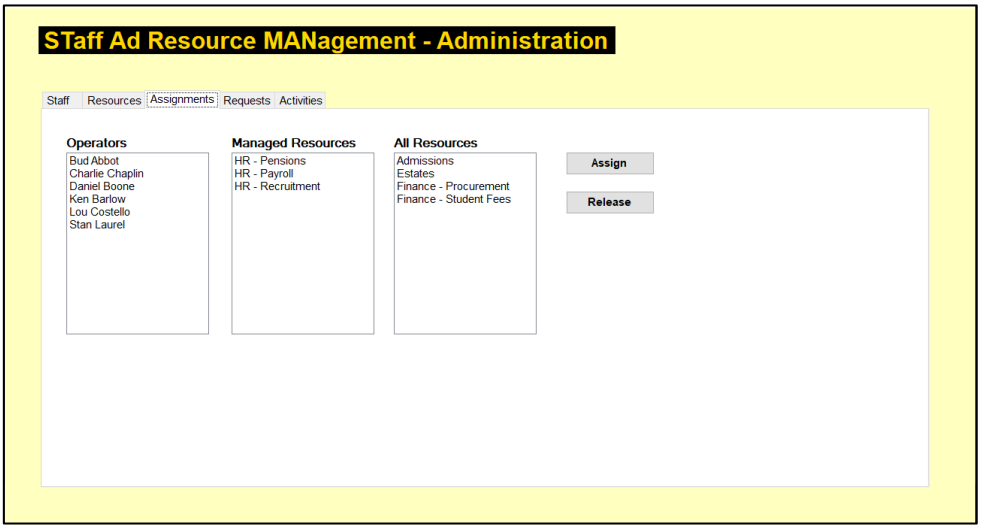


Figure 11: Operator/Resource Assignment

4.2.4 Review requests

This interface allows the administrator to review all requests made by operators. The list can be filtered by clicking on one or more operators from the list. Figure 12 shows the interface.

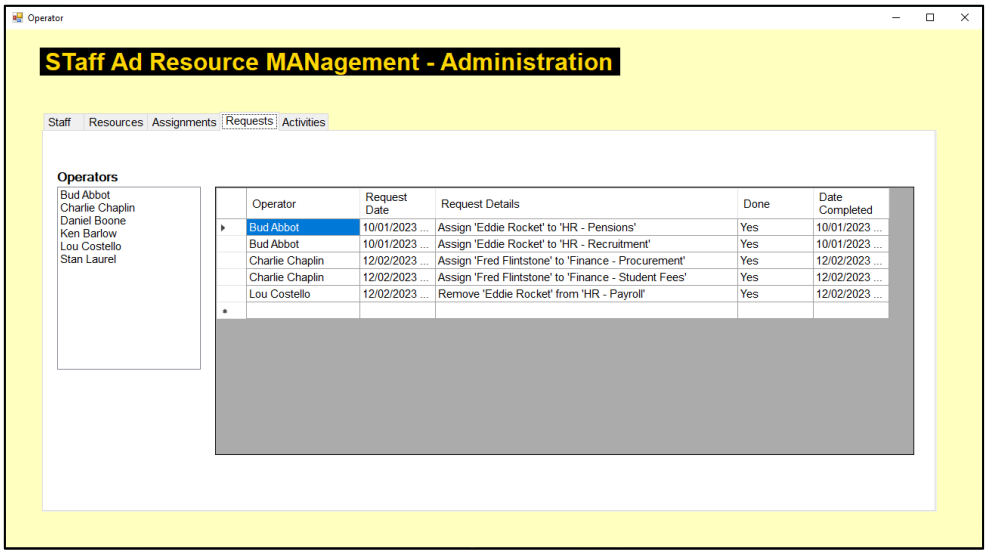


Figure 12: List of Requests submitted - filter by operator

4.2.5 Review activities

Using this interface, the administrator can review all activities completed within the system. The list can be filtered by selecting one or more Operators from the list. Figure 13 shows the interface.

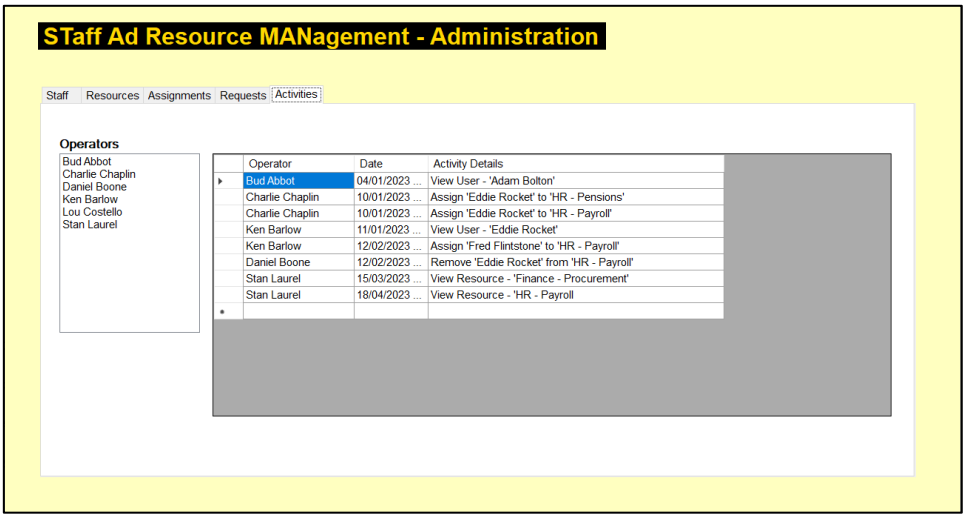
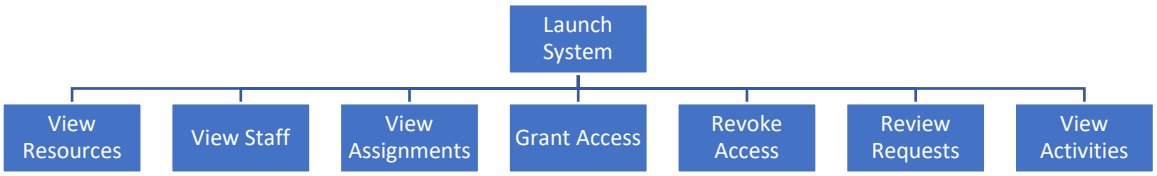


Figure 13: List of activities carried out by operators

5 System Flowchart

The application has a series of functions. This section of the document will outline each function by means of a flowchart.

5.1 End-user Flowchart



6 System Sequence Diagrams

Based on the Use Cases outlined in the Functional Specification for the application, the following are the System Sequence Diagrams (SSD) for the application.

6.1 Front-end Use Cases

6.1.1 View Resources

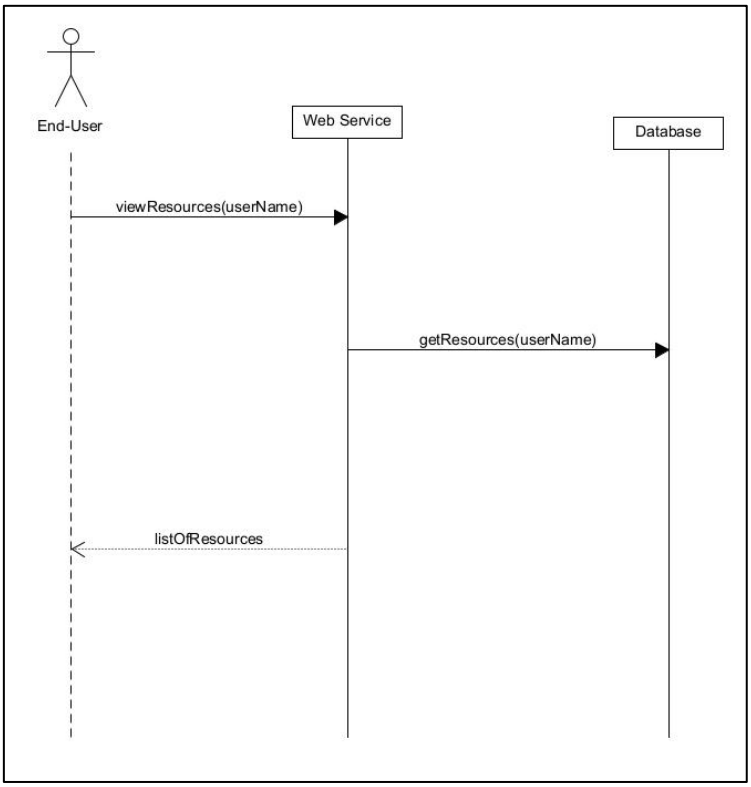


Figure 14: Sequence Diagram - View Resources

6.1.2 View Staff

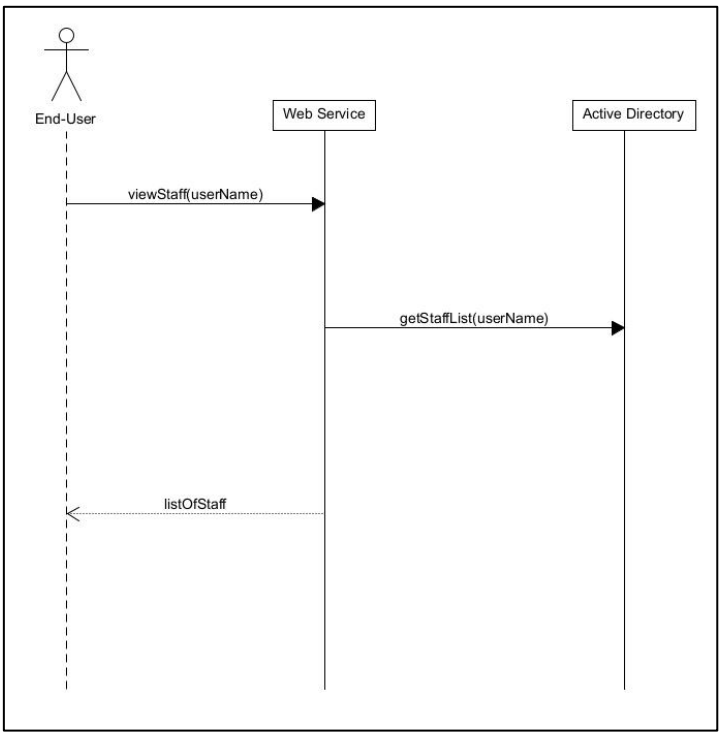


Figure 15: Sequence Diagram - View Staff

6.1.3 View Assignments

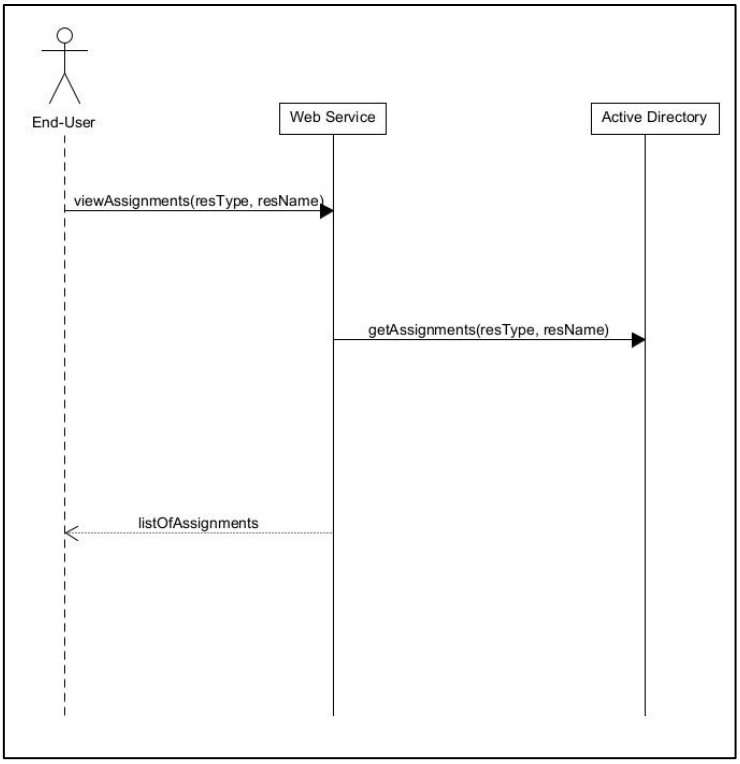


Figure 16: Sequence Diagram - View Assignments

6.1.4 Grant Access

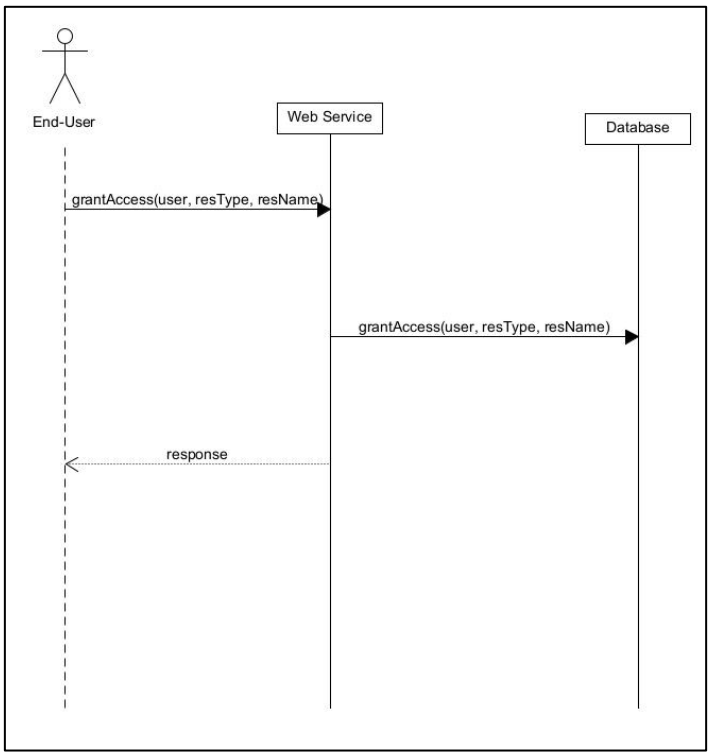


Figure 17: Sequence Diagram - Grant Access

6.1.5 Revoke Access

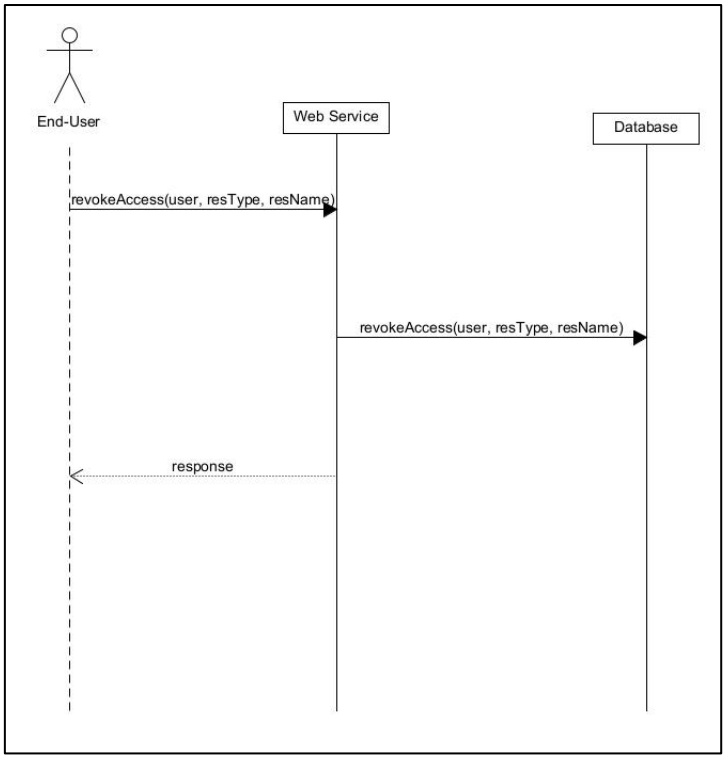


Figure 18: Sequence Diagram - Revoke Access

6.1.6 Review Requests

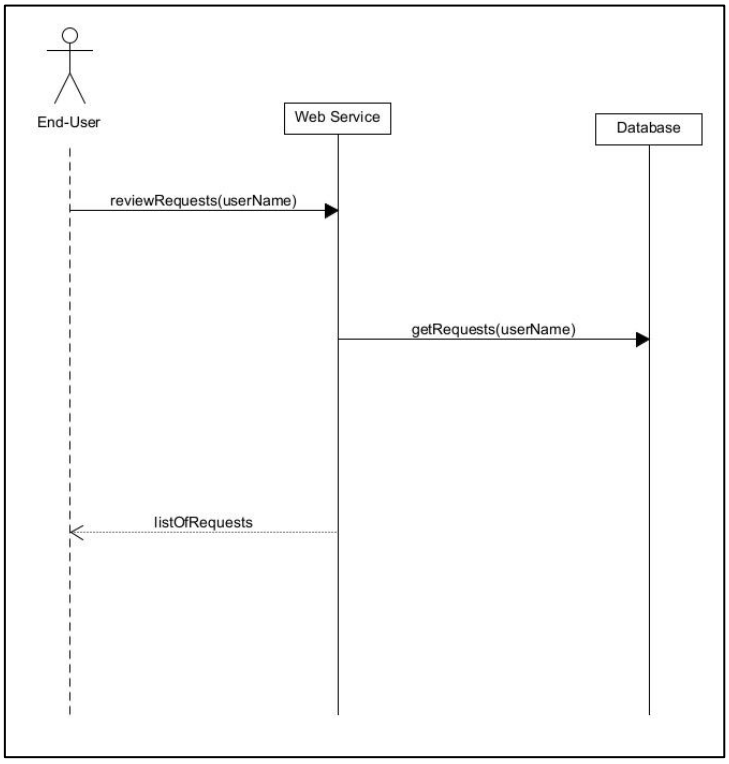


Figure 19: Sequence Diagram - Review Requests

6.1.7 View Activities

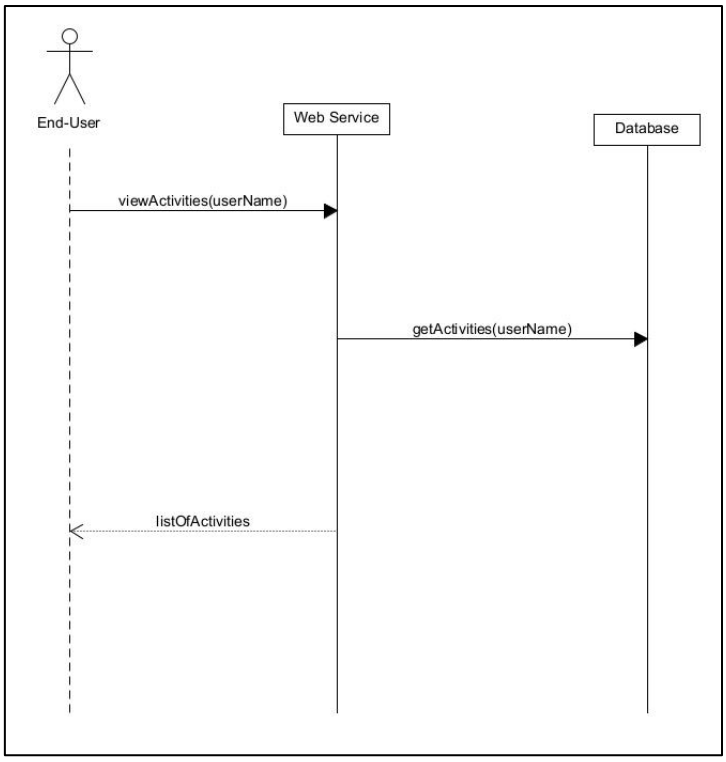


Figure 20: Sequence Diagram - View Activities

6.2 Administrator Use Cases

6.2.1 View Users

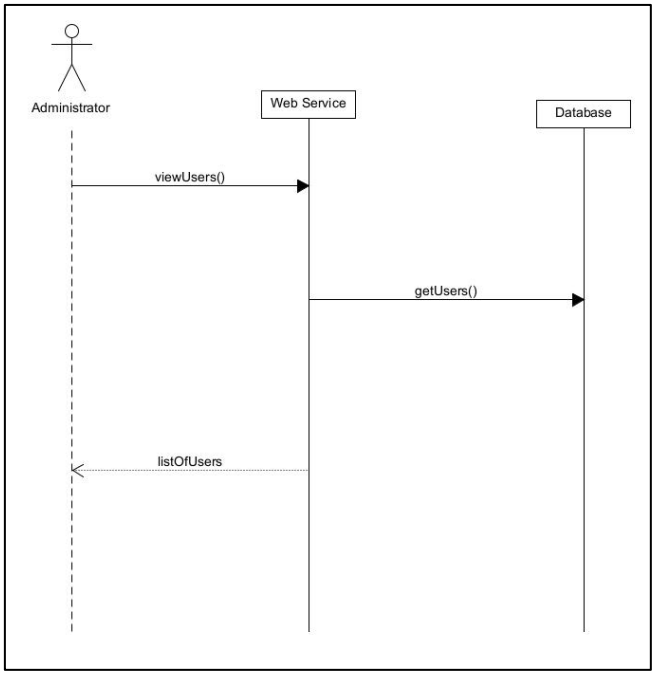


Figure 21: Administrator - View Users

6.2.2 Add User

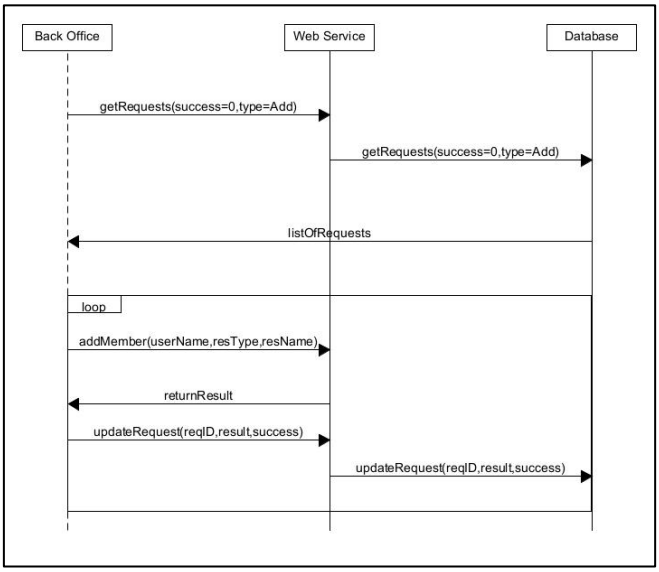


Figure 22: Administrator - Add User

6.2.3 Remove User

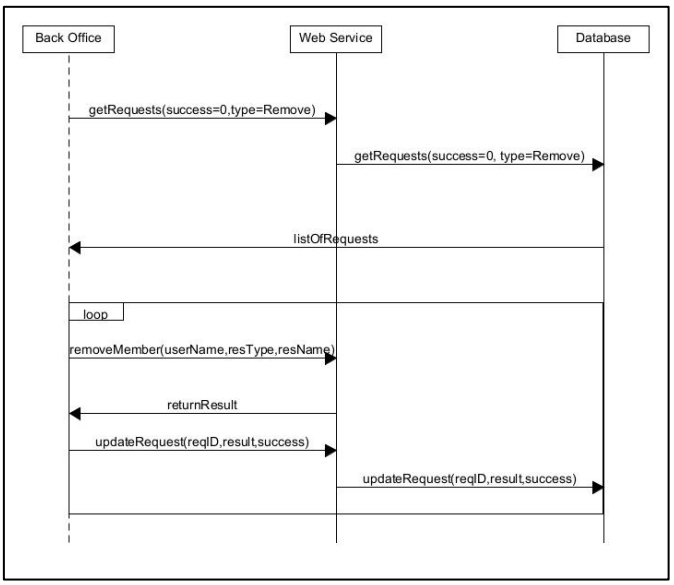


Figure 23: Administrator - Remove User

6.2.4 View Resources

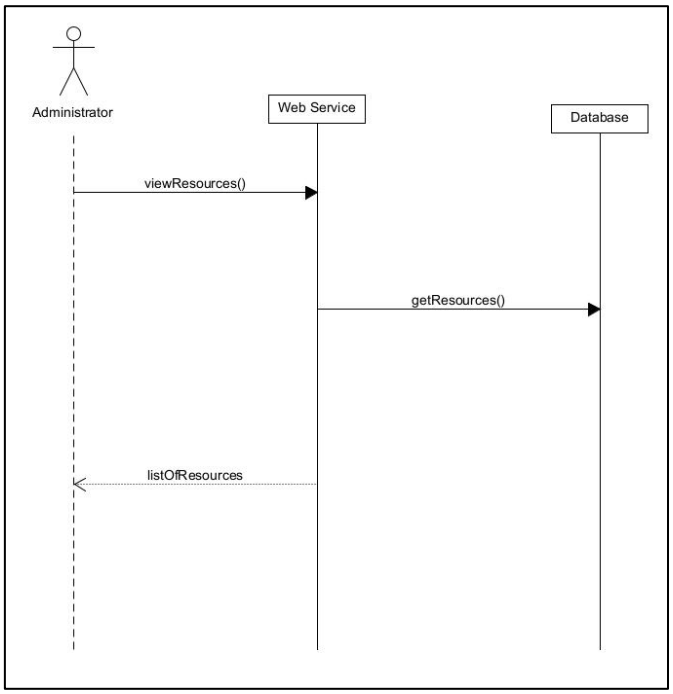


Figure 24: Administrator - View Resources

6.2.5 Add Resource

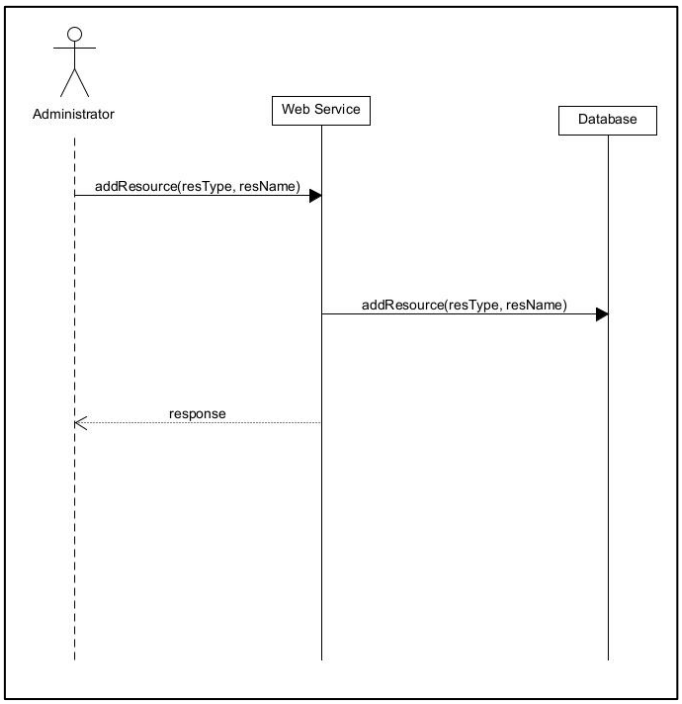


Figure 25: Administrator - Add Resource

6.2.6 Remove Resource

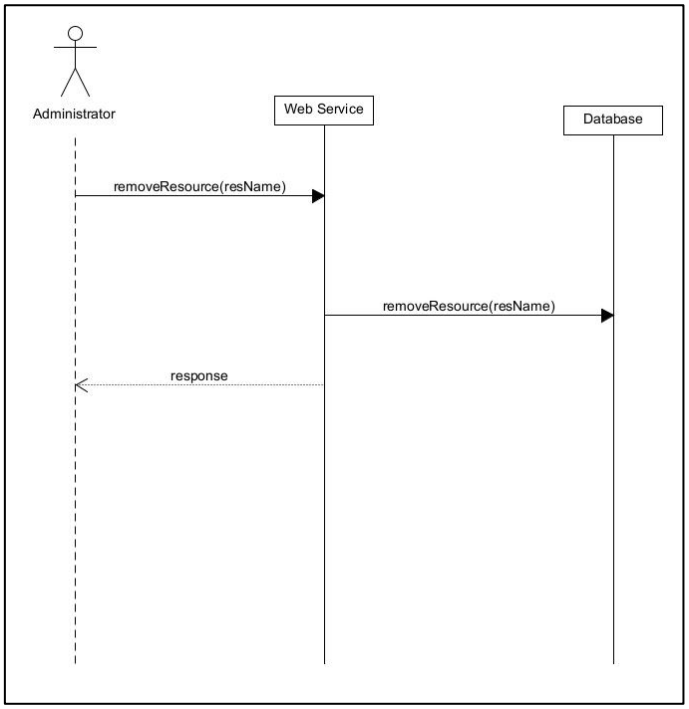


Figure 26: Administrator - Remove Resource

6.2.7 Create Matches

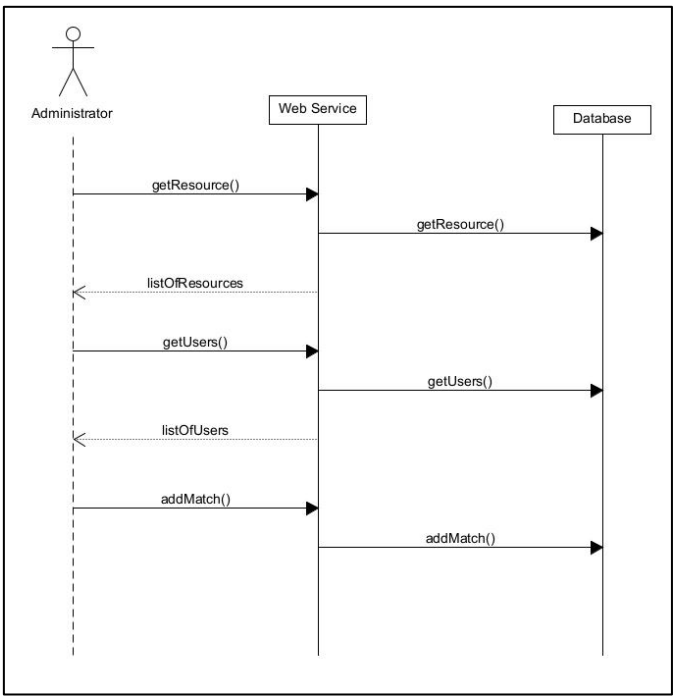


Figure 27: Administrator - Add Matches

6.2.8 Delete Matches

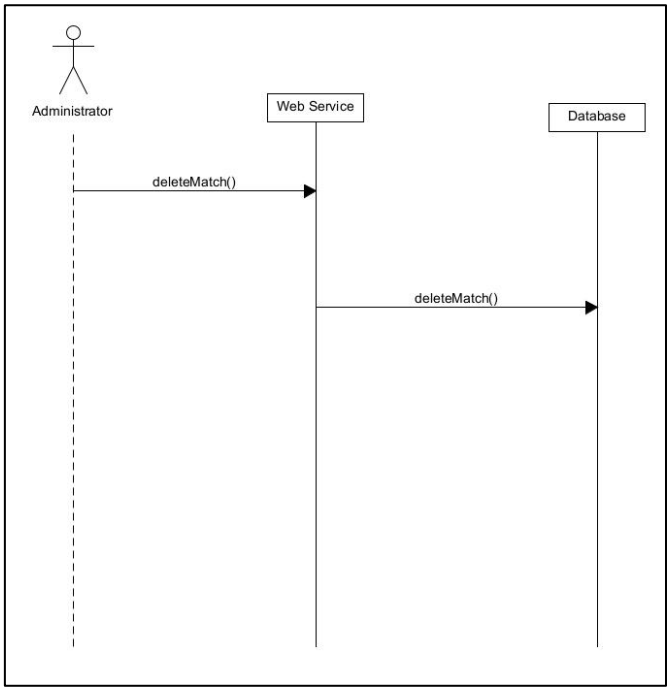


Figure 28: Administrator - Delete Matches

6.2.9 View Matches

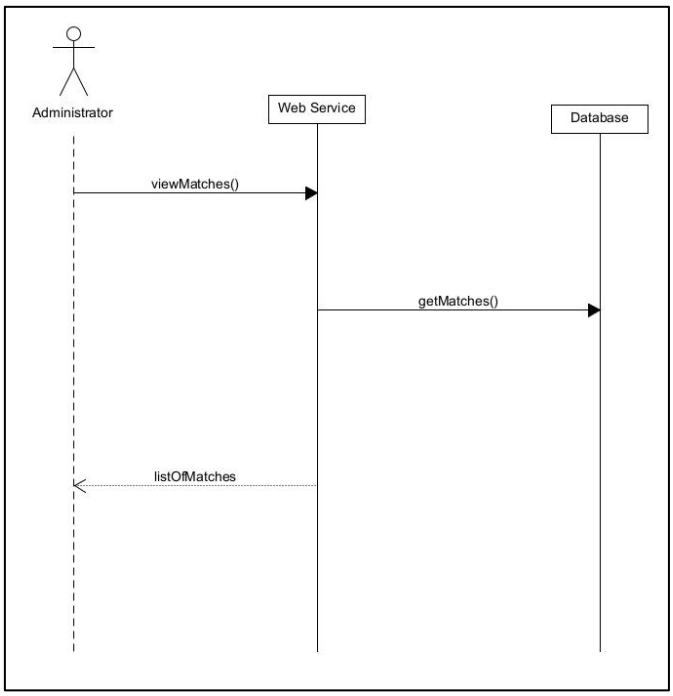


Figure 29: Administrator - View Matches

6.2.10 Review Requests

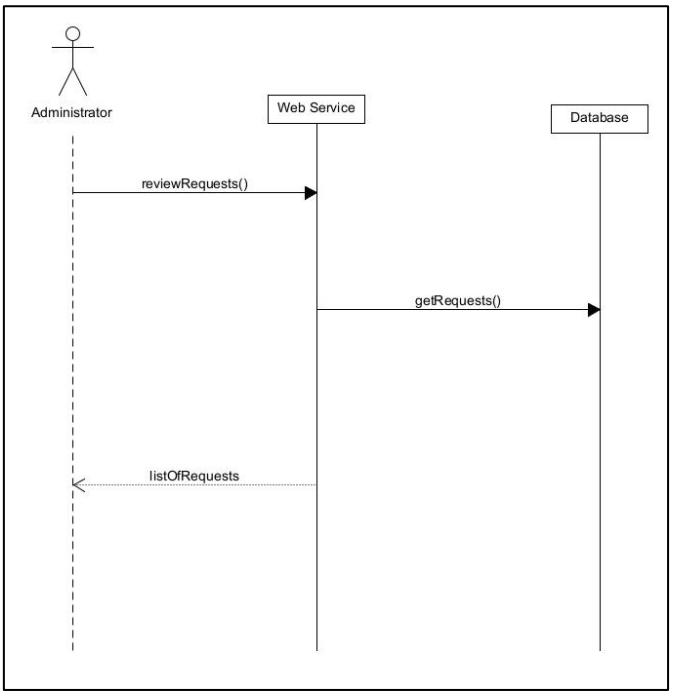


Figure 30: Administrator - Review Requests

6.2.11 View Activities

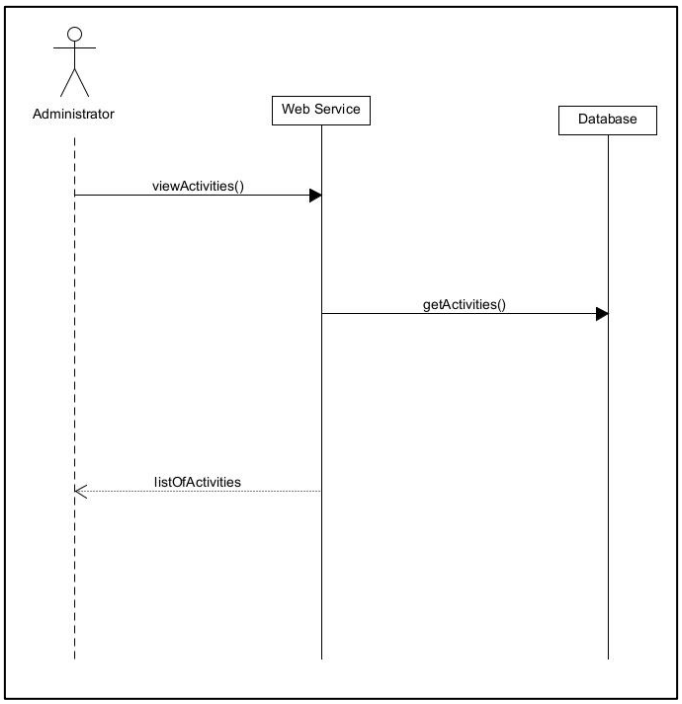


Figure 31: Administrator - View Activities

6.3 Back-office Use Cases

6.3.1 Add Member

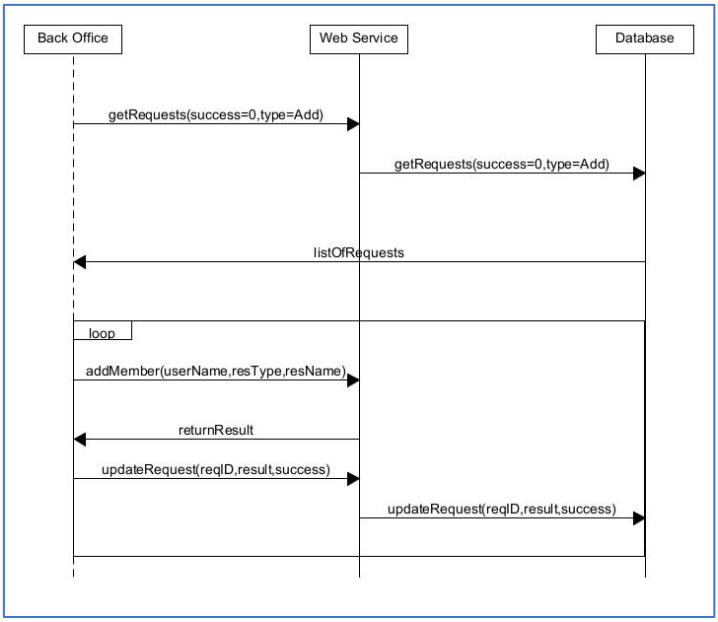


Figure 32: Sequence Diagram - Add Member

6.3.2 Remove Member

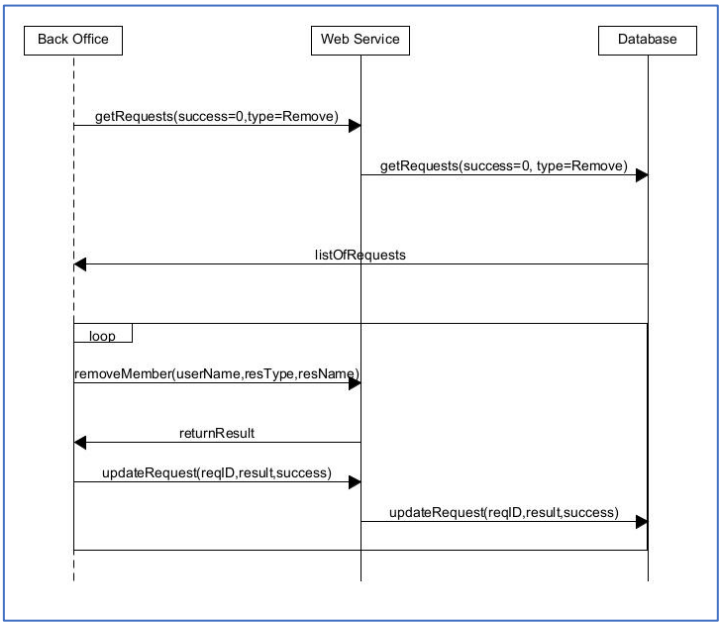


Figure 33: Sequence Diagram - Remove Member

6.3.3 Notify User

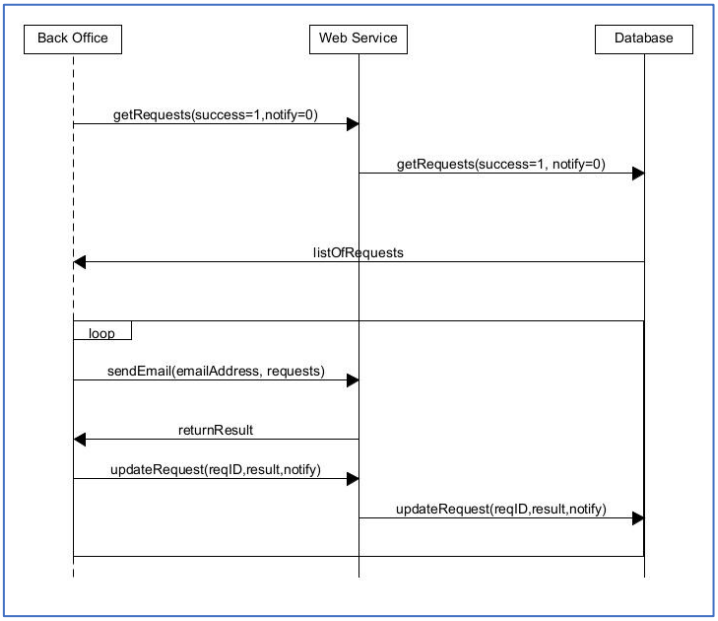


Figure 34: Sequence Diagram - Notify User

7 Database Schema

All data associated with the application is stored in a Microsoft SQL (MSQL) database. The application will access the data through a series of SQL statements. To reduce the risk of SQL injection attacks, all SQL statements will be parameterised. Each statement will be executed via the web service and will require a valid session token to be processed.

The following diagram shows the database schema, including the relationships between the various tables.

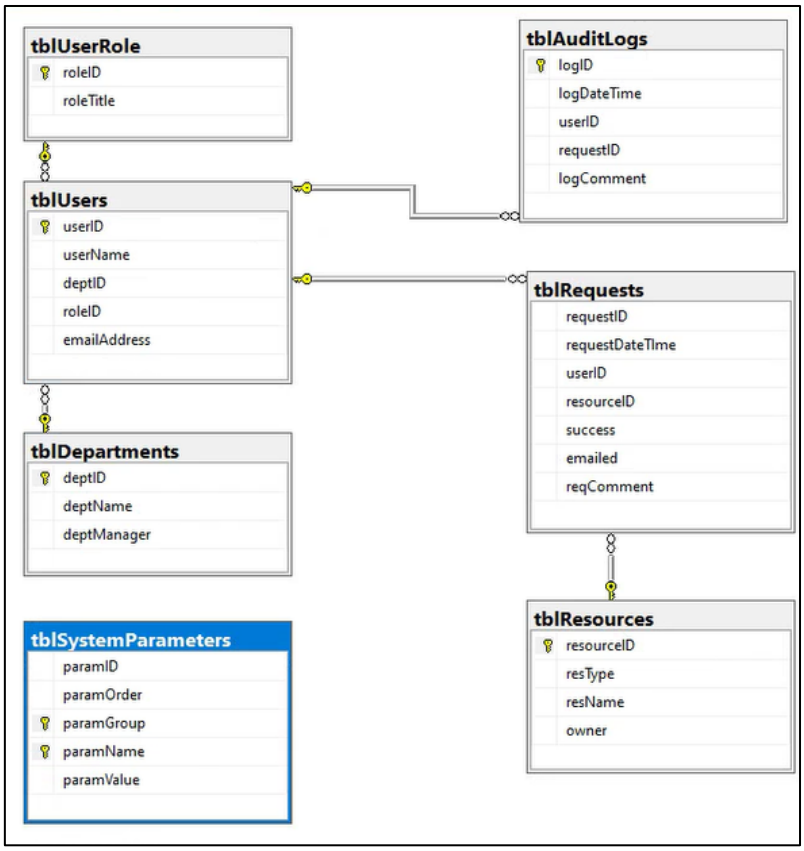


Figure 35: Database diagram

7.1 SQL database table creation

This section of the document describes the database tables outlined in Figure 35 and outlines the SQL statements required to create them.

7.1.1 tblResources

This table stores a full list of all resources that can be managed by the application user. One record exists for each managed resource. Each resource record has a unique GUID field. This field will be used to associate the resource with other tables where relevant.

Table 1 : SQL to create tblResources

```
CREATE TABLE [dbo].[tblResources](
	[resourceID] [uniqueidentifier] ROWGUIDCOL  NOT NULL,
	[resType] [varchar](30) NOT NULL,
	[resName] [varchar](100) NOT NULL,
	[owner] [varchar](100) NOT NULL,
 CONSTRAINT [PK_tblResources] PRIMARY KEY CLUSTERED
(
	[resourceID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
```

Table 2: Data table structure - tblResources

Field Name	Field Type	Default Value	Key Field
resourceID	uniqueidentifier	New GUID	Yes
resType	varchar(30)		Unique
resName	varchar(100)		Unique
Owner	varchar(100)		

7.1.2 tblUsers

This table stores a list of all authorised users for the system. This table allows the application to identify what department the user is associated with and what role the perform within the application. One record exists for each user of the application. Each user record contains a unique ID, used to reference the user in all relevant tables.

Table 3: SQL to create tblUsers

```
CREATE TABLE [dbo].[tblUsers](
	[userID] [uniqueidentifier] ROWGUIDCOL  NOT NULL,
	[userName] [varchar](100) NOT NULL,
	[deptID] [uniqueidentifier] NOT NULL,
	[roleID] [uniqueidentifier] NOT NULL,
	[emailAddress] [varchar](100) NOT NULL,
 CONSTRAINT [PK_tblUsers] PRIMARY KEY CLUSTERED
(
	[userID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
```

Table 4: Data Table Structure - tblUsers

Field Name	Field Type	Default Value	Key Field
userID	uniqueidentifier	New GUID	Yes
userName	varchar(100)		Unique
department	varchar(100)		
roleID	varchar(100)		
emailAddress	varchar(100)		

7.1.3 tblRequests

This table stores details of all requests made using the application. One record exists for each request made. Each record contains a unique ID, used to reference the request in all relevant tables.

Table 5: SQL to create tblRequests

```
CREATE TABLE [dbo].[tblRequests](
	[requestID] [uniqueidentifier] ROWGUIDCOL  NOT NULL,
	[requestDateTime] [datetime] NOT NULL,
	[requestType] [varchar(6)] NOT NULL,
	[userID] [uniqueidentifier] NOT NULL,
	[resourceID] [uniqueidentifier] NOT NULL,
	[success] [bit] NOT NULL,
	[emailed] [bit] NOT NULL,
	[reqComment] [varchar](max) NULL
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
```

Table 6: Data Table Structure - tblRequests

Field Name	Field Type	Default Value	Key Field
requestID	uniqueidentifier	New GUID	Yes
requestDateTime	DateTime	GetDate()	
requestType	Varchar(6)		
userID	uniqueidentifier		
resourceID	uniqueidentifier		
success	Bit	0	
emailed	Bit	0	
reqComment	varchar(MAX)		

7.1.4 tblAuditLogs

This table stores details of all operations carried out by users and operators. This acts as an audit trail of the system. Each record contains a unique ID.

Table 7: SQL to create tblAuditLogs

```
CREATE TABLE [dbo].[tblAuditLogs](
	[logID] [uniqueidentifier] ROWGUIDCOL  NOT NULL,
	[logDateTime] [datetime] NOT NULL,
	[userID] [uniqueidentifier] NOT NULL,
	[requestID] [uniqueidentifier] NOT NULL,
	[logComment] [varchar](max) NULL,
 CONSTRAINT [PK_tblAuditLogs] PRIMARY KEY CLUSTERED
(
	[logID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
```

Table 8: Data Table Structure - tblAuditLogs

Field Name	Field Type	Default Value	Key Field
logID	uniqueidentifier	New GUID	
logDateTime	DateTime	GetDate()	
userID	uniqueidentifier		
requestID	uniqueidentifier		
logComment	varchar(MAX)		

7.1.5 tblDepartments

This table stores details of all the departments of the users. There is one record per department. Each record contains a unique ID, used to reference the department in all relevant tables.

Table 9: SQL to create tblDepartments

```
CREATE TABLE [dbo].[tblDepartments](
	[deptID] [uniqueidentifier] ROWGUIDCOL  NOT NULL,
	[deptName] [varchar](100) NOT NULL,
	[deptManager] [uniqueidentifier] NOT NULL,
 CONSTRAINT [PK_tblDepartments] PRIMARY KEY CLUSTERED
(
	[deptID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
```

Table 10: Data Table Structure - tblDepartments

Field Name	Field Type	Default Value	Key Field
deptID	uniqueidentifier	New GUID	Yes
deptName	varchar(100)		
deptManager	uniqueidentifier		

7.1.6 tblUserRole

This table stores details of all the user roles within the application. There is one record per user role. The role of the user determines what experience they have when using the system. Each record contains a unique ID, used to reference the user role in all relevant tables.

Table 11: SQL to create tblUserRole

```
CREATE TABLE [dbo].[tblUserRole](
	[roleID] [uniqueidentifier] ROWGUIDCOL  NOT NULL,
	[roleTitle] [varchar](100) NOT NULL,
 CONSTRAINT [PK_tblUserRole] PRIMARY KEY CLUSTERED
(
	[roleID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
```

Table 12: Data Table Structure - tblUserRole

Field Name	Field Type	Default Value	Key Field
roleID	uniqueidentifier	New GUID	Yes
roleTitle	varchar(100)		

7.1.7 tblSystemParameters

This table stores system parameters. There are a number of different system parameters, grouped by type and ordered numerically. Each record contains a unique ID, used to reference the user in all relevant tables.

Table 13: SQL to create tblSystemParameters

```
CREATE TABLE [dbo].[tblSystemParameters](
	[paramID] [uniqueidentifier] ROWGUIDCOL  NOT NULL,
	[paramOrder] [int] NOT NULL,
	[paramGroup] [varchar](100) NOT NULL,
	[paramName] [varchar](100) NOT NULL,
	[paramValue] [varchar](500) NOT NULL,
 CONSTRAINT [PK_tblSystemParameters] PRIMARY KEY CLUSTERED
(
	[paramGroup] ASC,
	[paramName] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
```

Table 14: Data Table Structure - tblSystemParameters

Field Name	Field Type	Default Value	Key Field
paramID	uniqueidentifier	New GUID	Yes
paramOrder	int		
paramGroup	varchar(100)		
paramName	varchar(100)		
paramValue	varchar(500)		

7.2 Use Case SQL statements

This section of the document will outline the SQL statements associated with each Use Case covered in Sections 6.1 and 6.2.

7.2.1 View Resources

Requestor	SQL
User front-end	SELECT resourceID, resType, resName, owner FROM tblResources WHERE resName=@username

7.2.2 View Staff

Requestor	SQL
User front-end	SELECT deptName from tblDepartments WHERE deptManager=@username
	SELECT userID, username, department, roleID, emailAddress from tblUsers WHERE department=@userdept

7.2.3 View Assignments

Requestor	SQL
User front-end	SELECT resourceID, resType, resName, owner FROM tblResources WHERE resName=@username

The above data set will then be used to query Active Directory to identify current assignments.

7.2.4 Grant Access

Requestor	SQL
User front-end	INSERT INTO tblRequests (userID, resourceID, requestType) VALUES (@userid, @resourceid, “Add”)
Console application	UPDATE tblRequests SET success=@success, reqComment=@comment WHERE requestID=@reqid

7.2.5 Revoke Access

Requestor	SQL
User front-end	INSERT INTO tblRequests (userID, resourceID, requestType) VALUES (@userid, @resourceid, “Remove”)
Console application	UPDATE tblRequests SET success=@success, reqComment=@comment WHERE requestID=@reqid

7.2.6 Review Requests

Requestor	**SQL**
User front-end	SELECT requestID, requestDateTime, requestType, resName FROM tblRequests req INNER JOIN tblResources res ON req.resourceID=res.resourceID WHERE userID=@user

7.2.7 View Activities

Requestor	**SQL**
User front-end	SELECT logDateTime, requestType, requestDateTime, resourceName FROM tblAuditLogs al INNER JOIN tblRequests req ON al.requestID=req.requestID WHERE userID=@user

7.3 Administrator Use Cases

7.3.1 View Users

Requestor	**SQL**
Administrator front-end	SELECT userID, userName, emailAddress, deptname, roleTitle FROM tblUsers usrs INNER JOIN tblDepartments dpt ON usrs.department=dept.deptID INNER JOIN tblUserRole rol ON usrs.roleID=rol.roleID ORDER BY userName ASC

7.3.2 Add User

Requestor	**SQL**
Administrator front-end	INSERT INTO tblUsers (username, department, roleID, emailAddress) VALUES (@username, @dept, @role, @email)

7.3.3 Remove User

Requestor	**SQL**
Administrator front-end	DELETE FROM tblUsers WHERE userName=@username

7.3.4 View Resources

Requestor	**SQL**
Administrator front-end	SELECT resourceID, resType, resName, Owner FROM tblResources ORDER BY resType, resName

7.3.5 Add Resource

Requestor	**SQL**
Administrator front-end	INSERT INTO tblResources (resType, resName, Owner) VALUES (@restype, @resname, @owner)

7.3.6 Remove Resource

Requestor	**SQL**
Administrator front-end	DELETE FROM tblResources WHERE resID=@resid

7.3.7 Create Matches

Requestor	**SQL**
Administrator front-end	SELECT resourceID, resType, resName, owner FROM tblResources
Administrator front-end	SELECT userID, username FROM tblUsers WHERE username=@username
Administrator front-end	UPDATE tblResources SET owner=@username WHERE resourceID=@resid

7.3.8 Delete Matches

Requestor	**SQL**
Administrator front-end	UPDATE tblResources SET owner=null WHERE resourceID=@resid

7.3.9 View Matches

Requestor	**SQL**
Administrator front-end	SELECT resType, resName, userName, emailAddress, deptname FROM tblResources INNER JOIN tblUsers usrs ON usrs.userName=owner ORDER BY resName ASC

7.3.10 Review Requests

Requestor	**SQL**
Administrator front-end	SELECT requestID, requestDateTime, requestType, resName FROM tblRequests req INNER JOIN tblResources res ON req.resourceID=res.resourceID ORDER BY requestDateTime

7.3.11 View Activities

Requestor	**SQL**
Administrator front-end	SELECT logDateTime, requestType, requestDateTime, resourceName FROM tblAuditLogs al INNER JOIN tblRequests req ON al.requestID=req.requestID ORDER BY logDateTime

7.4 Back-office Use Cases

7.4.1 Add Member

Requestor	**SQL**

Back-End application	SELECT requestID, username, resourceType, resourceName FROM tblRequests req INNER JOIN tblResources res ON res.resourceID = req.resourceID INNER JOIN tblUsers usrs ON usrs.userID=req.userID WHERE success=@success AND requestType=@reqType
Back-End application	UPDATE tblRequests SET success=@success, reqComment=@comment WHERE requestID=@reqid

7.4.2 Remove Member

Requestor	**SQL**
Back-End application	SELECT requestID, username, resourceType, resourceName FROM tblRequests req INNER JOIN tblResources res ON res.resourceID = req.resourceID INNER JOIN tblUsers usrs ON usrs.userID=req.userID WHERE success=@success AND requestType=@reqType
Back-End application	UPDATE tblRequests SET success=@success, reqComment=@comment WHERE requestID=@reqid

7.4.3 Notify User

Requestor	**SQL**
Back-End application	SELECT requestDateTime, requestType, resourceName, username, reqComment from tblRequests req INNER JOIN tblResources res ON req.resourceID=res.resourceID INNER JOIN tblUsers usrs ON req.userID=usrs.userID WHERE success=@success AND emailed=@emailed

8 Class Diagram

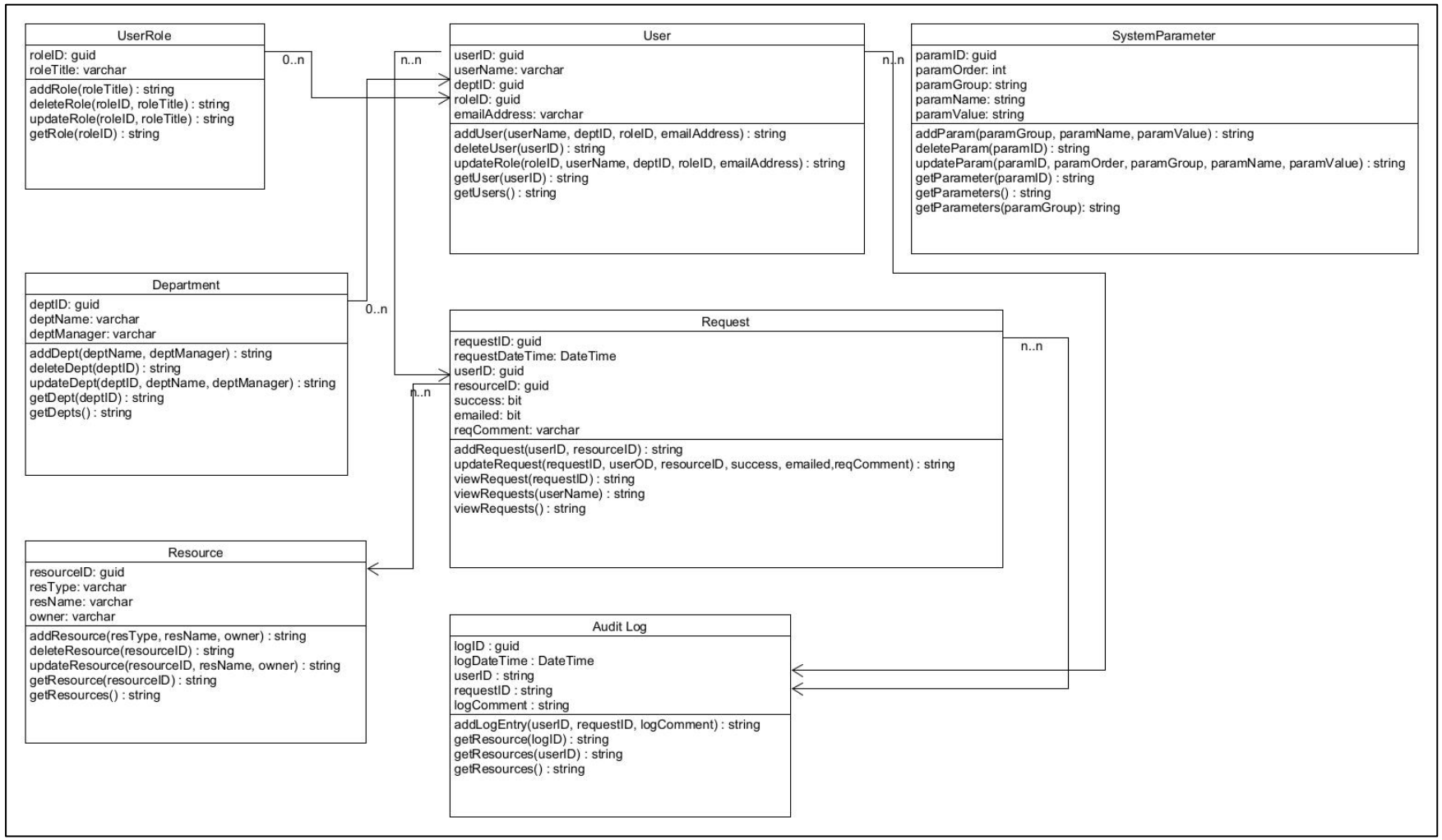


Figure 36: Class Diagram

9 Conclusion

The purpose of this document is to provide a detailed breakdown of the function of all aspects of the STARMAN application. The document provides all the necessary information to support the successful development of the STARMAN application.

The document covers the UI design, Use Cases for each function of the system and details of the structure and SQL code necessary for each function too.

10 Glossary of Terms

AD	Active Directory – The Microsoft infrastructure for managing user and resource objects and all aspects of primary levels of security.
AAD	Azure Active Directory – This is the cloud implementation of the Microsoft Active Directory platform and infrastructure (see AD above)
SETU	South-East Technological University
DC	Domain Controller – A special server in an AD environment that manages and controls all aspects of the AD structure. The DC stores the AD database and manages the replication of all AD data across other network servers as required
MIIS	Microsoft Internet Information Services – A platform for hosting web sites
SQL	Simple Query Language – an industry standard for generating structured queries to manage databases