Science & Technology Facilities Council
ISIS

Hannah Griffin, Tom Griffin, Marcus Noble,
Daniel Pope, Matt Tallyn

Visitor Details Management System

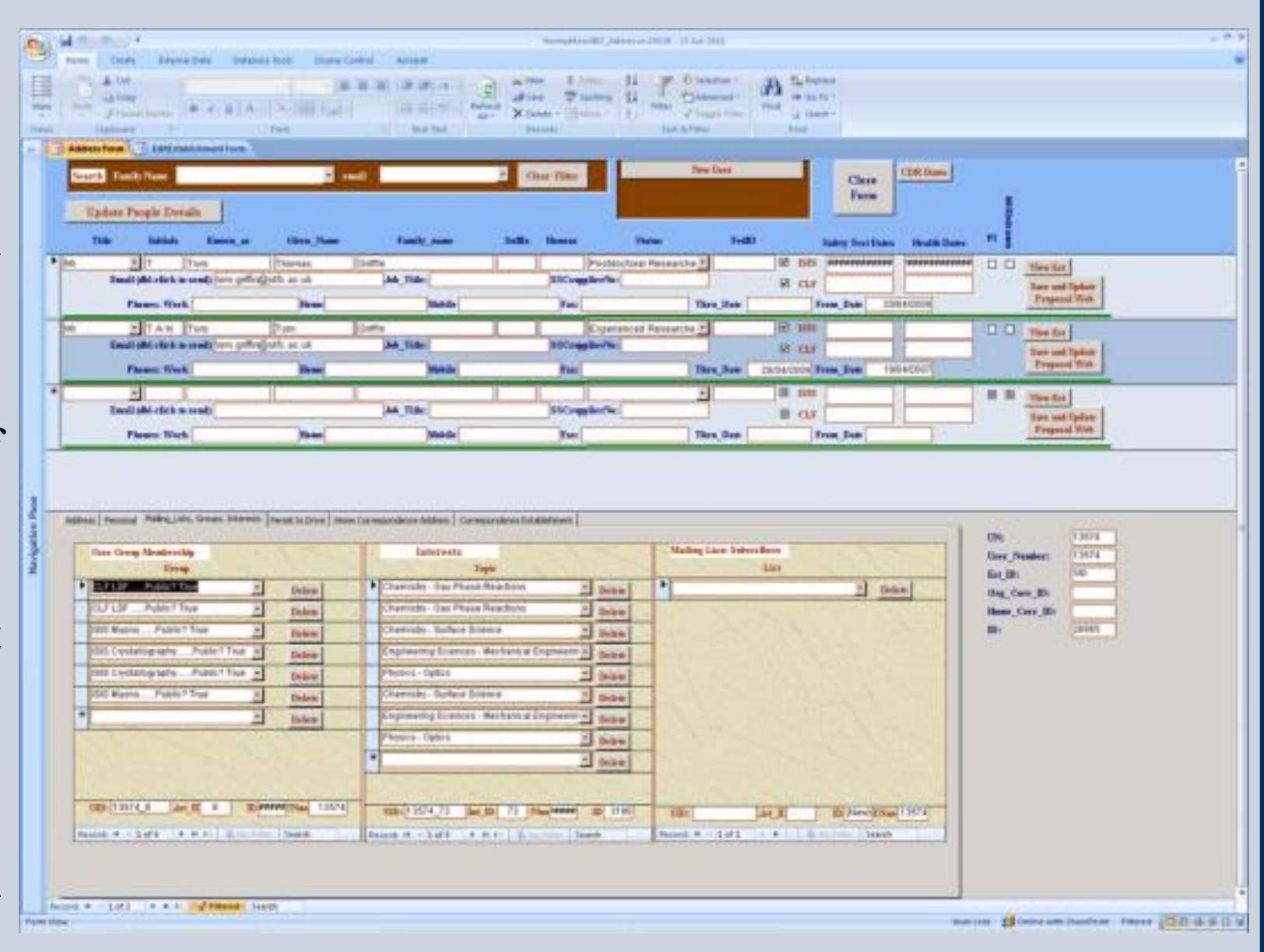
Overview

- Controls access to the user details database
- Performs authentication for logins
- Restricts access to data based on a user's permissions
- Allows modification of users details by the user office team
- Allows the creation of Federal Ids (STFC Active Directory)
- Exposes a method to allow Wi-Fi logons for current visitors

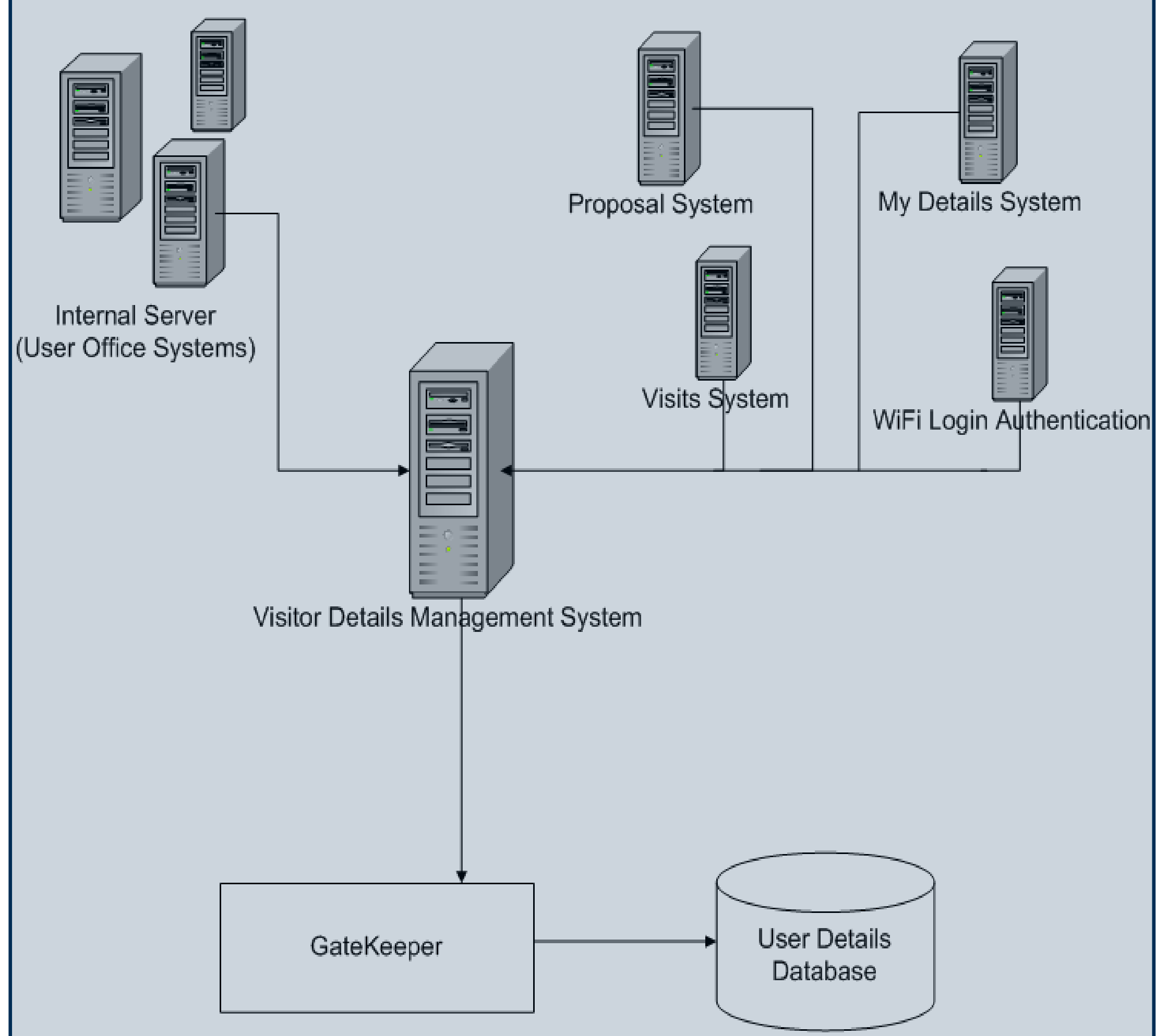
Old Systems

Previously...
- Microsoft Access applications
- Old and slow
- Required too many manual steps by the user office team
- Stores details in two separate databases that easily became out of sync
- Data had become unreliable and was causing problems in other systems that depended on it

We have been working to rewrite these applications using ASP .NET and Java JSP



Structure Overview



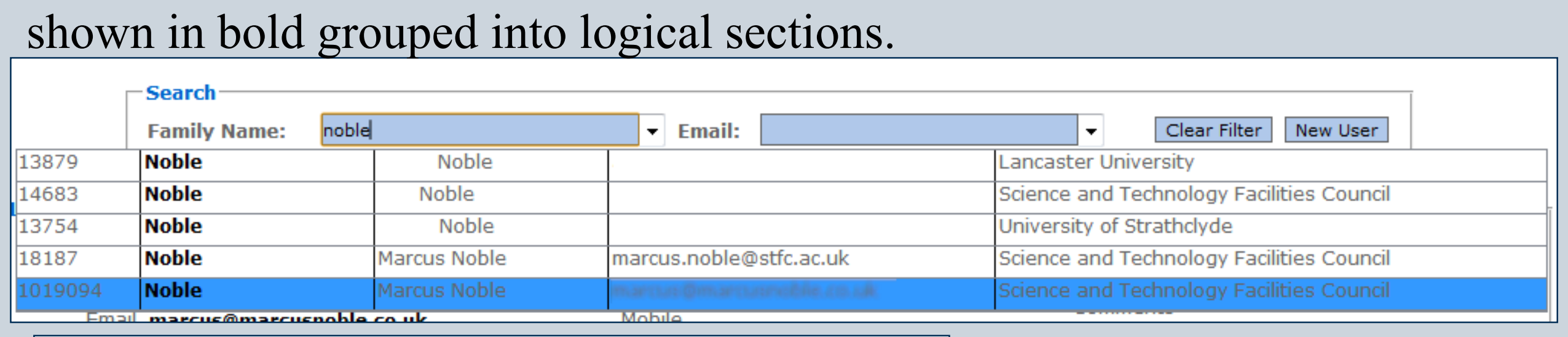
Wireless

We have provided a function for central IT team to query our database for current visitors and allow them access to the Wi-Fi network after successfully authenticating. We achieve this by creating a view in the Visits Database that is populated with all users currently on a visit +/- 1 week. We then expose a SOAP web service method that can be used to check if a given users exists on this list and, providing their logon credentials are correct, grant them access to the Wi-Fi on site.
This enabled us to move away from issuing paper "tokens" that users would need to collect from the user office and avoid the need of using Federal IDs to log on.

```java
@WebMethod
@ExcludeClassInterceptors
public Boolean isUserOnVisit(String userNumber){
    log.debug("Checking if user ["+userNumber+"] is currently on a visit");
    Query q = em.createQuery("SELECT DISTINCT v FROM Visitor v WHERE " +
            "(v.arrival <= :now AND v.departure >= :now) "+
            "OR (v.arrival > :now AND v.arrival < :upper) "+
            "OR (v.departure < :now AND v.departure > :lower) ");
    q.setParameter("now", new Date());
    Calendar cal = new GregorianCalendar();
    cal.add(Calendar.DATE, -7);
    q.setParameter("lower", cal.getTime());
    cal.add(Calendar.DATE, 14);
    q.setParameter("upper", cal.getTime());
    List<Visitor> currentVisitors = q.getResultList();
    if(currentVisitors==null || currentVisitors.isEmpty()){
        log.debug("User is NOT on a visit");
        return false;
    }
    for(Visitor v : currentVisitors){
        if(v.getUserNumber().toString().equals(userNumber)){
            log.debug("User IS on a visit");
            return true;
        }
    }
    log.debug("User is NOT on a visit");
    return false;
}
```

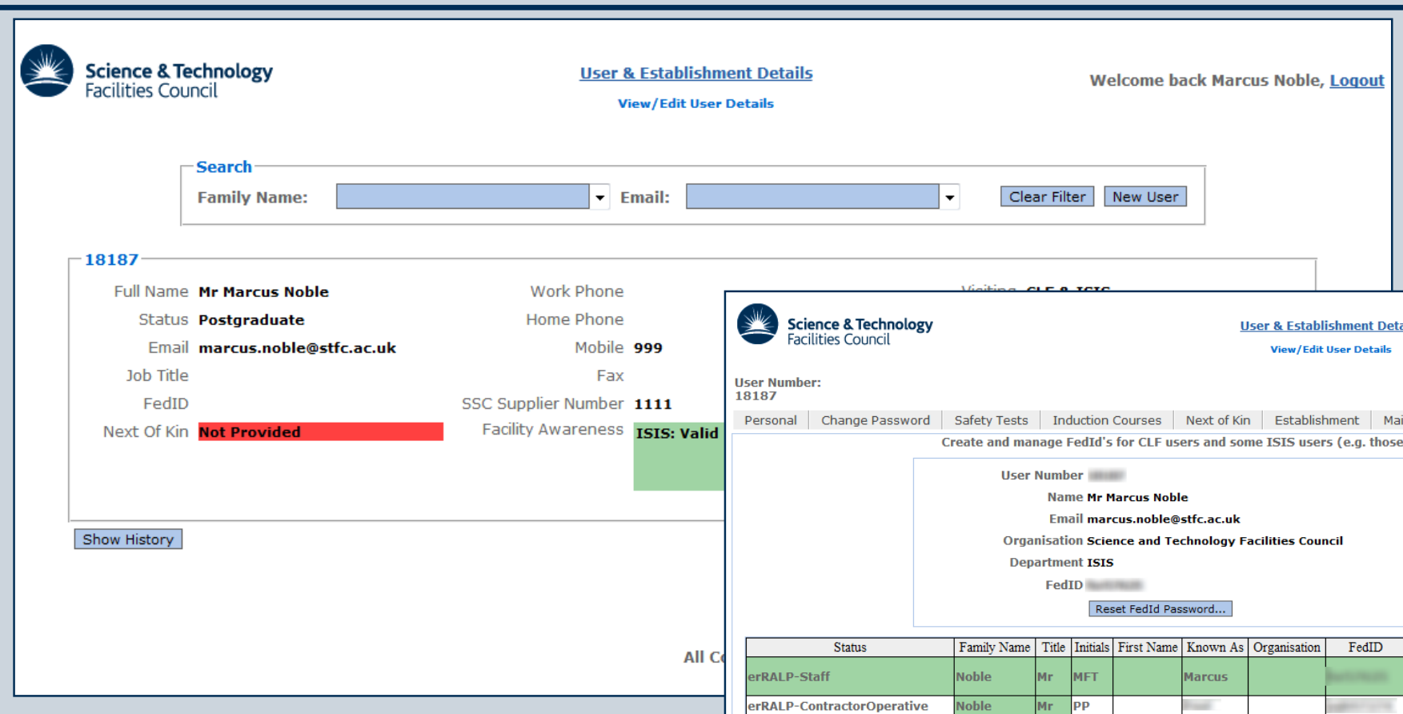
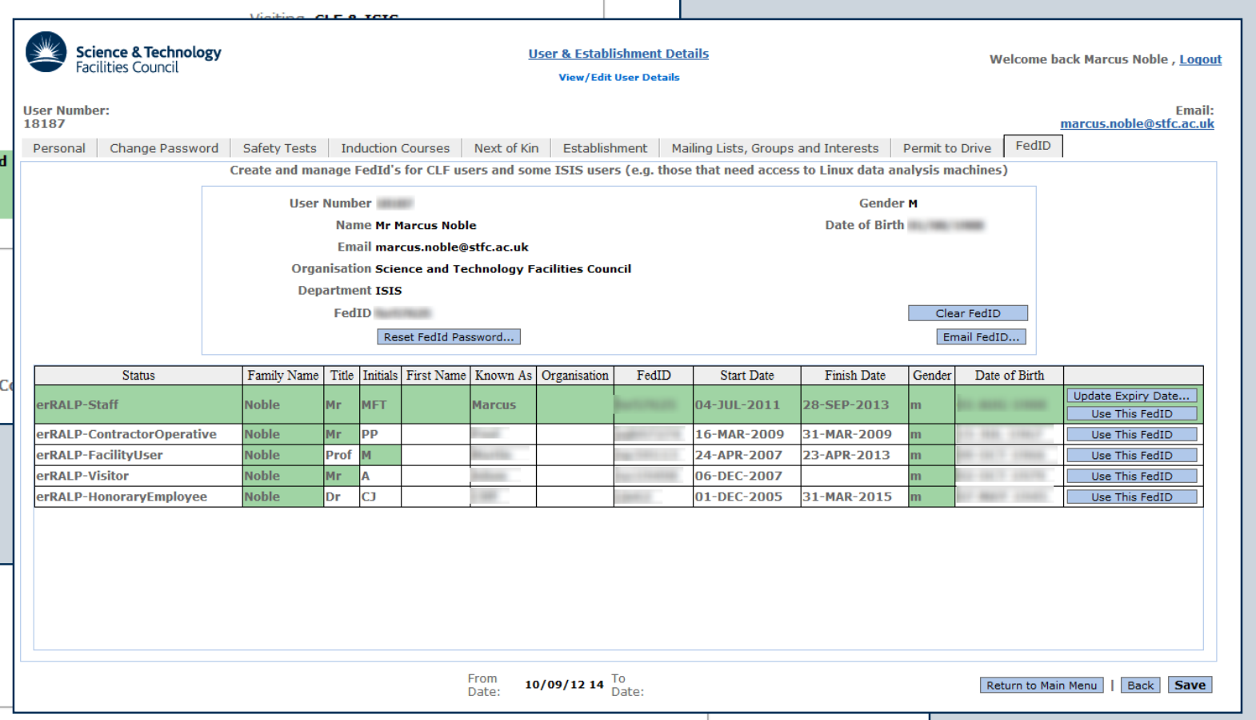
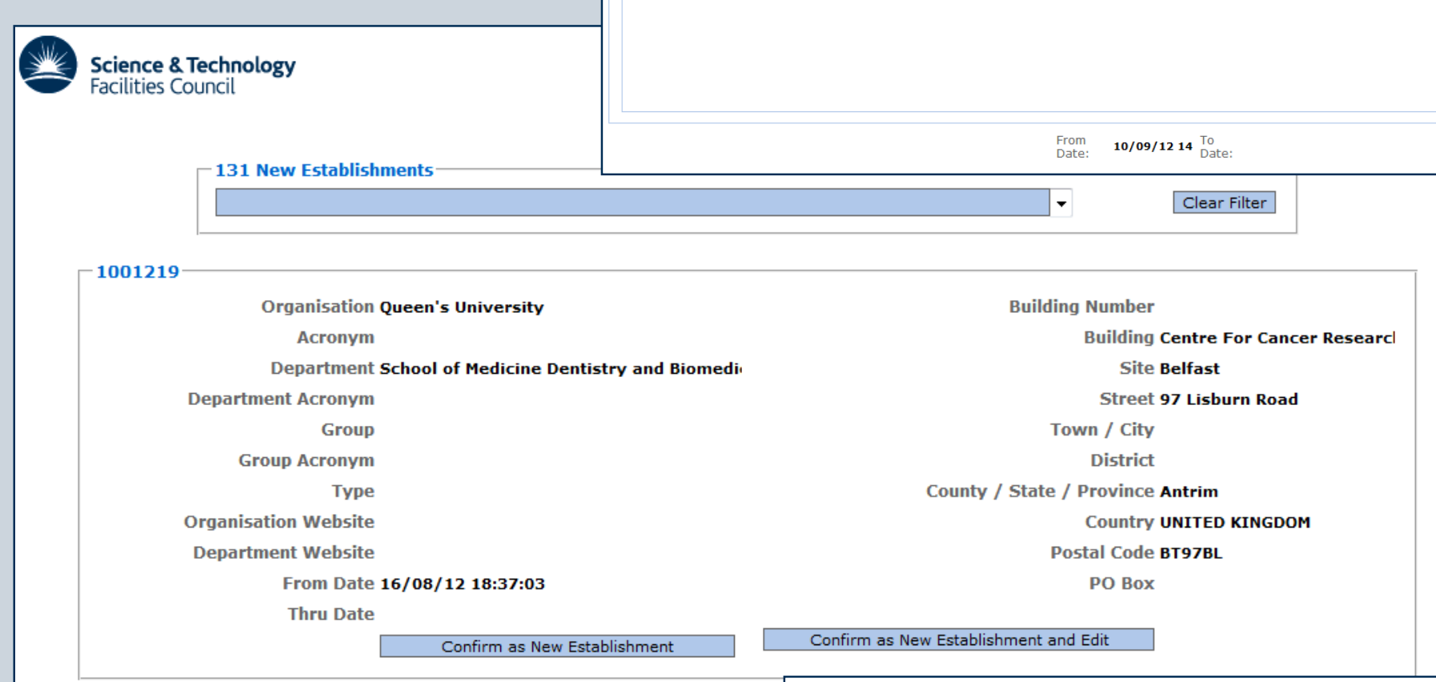
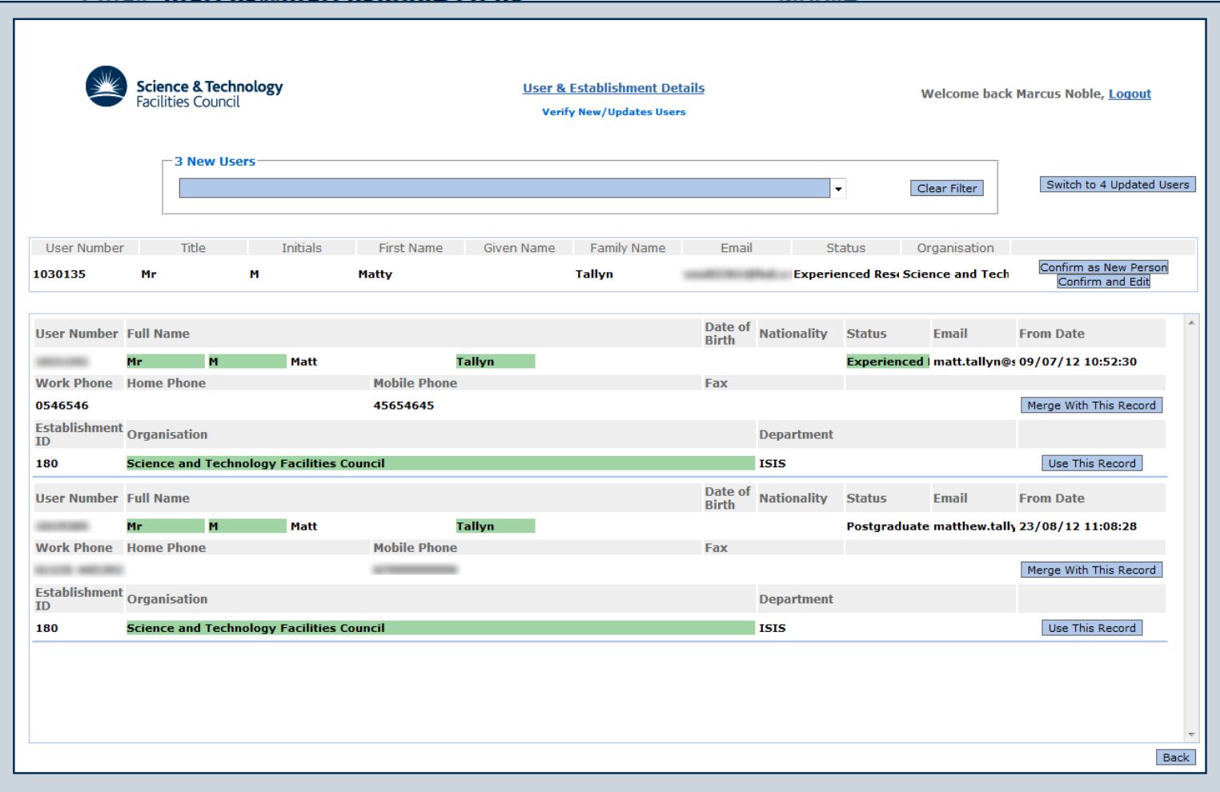
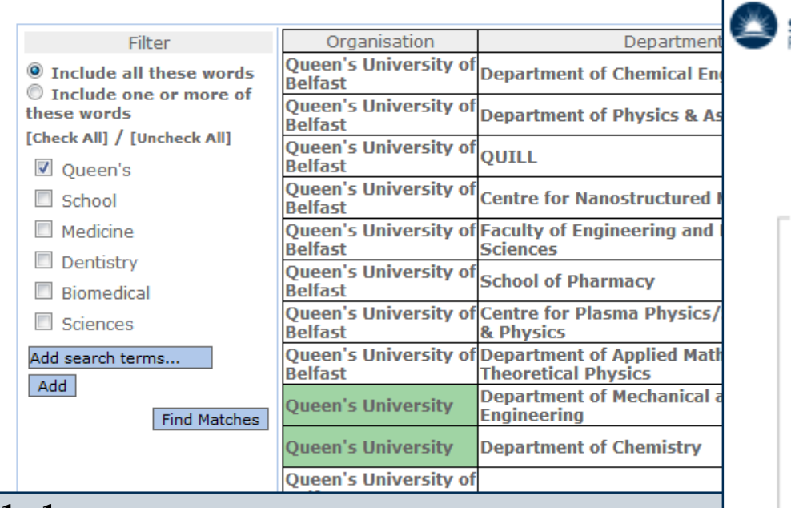
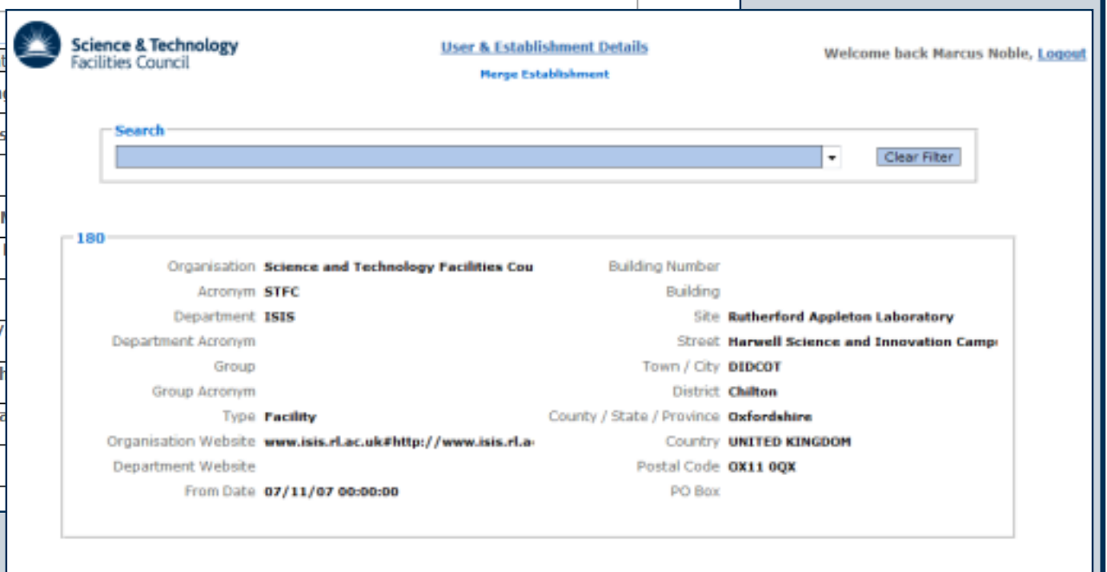
New Systems

The new web-based system now groups together all the users and establishments basic details into a single simple page allowing quick access to the most relevant data. Additional details, such as next of kin details, mailing lists, Induction courses and the status of safety tests, and the ability to edit details are shown on a tabbed page. Colour is used to give a quick indication of the status of certain details such as the presence of next of kin details and the validity of safety tests.
The focus of the system is on the data so the interface has been kept clean of clutter and unnecessary elements and the information that matters is shown in bold grouped into logical sections.



Search
Family Name: noble Email: Clear Filter New User

13879	**Noble**	Noble		Lancaster University
14683	**Noble**	Noble		Science and Technology Facilities Council
13754	**Noble**	Noble		University of Strathclyde
18187	**Noble**	Marcus Noble	marcus.noble@stfc.ac.uk	Science and Technology Facilities Council
1019094	**Noble**	Marcus Noble	[redacted]	Science and Technology Facilities Council

We made use of jQuery and FlexBox, a JavaScript library to provide searchable dropdown lists to provide this quick and easy user search box that allows our user office staff to quickly find a user based on their surname or email address.

New features included:
- A single system controlling user logons for all other system
- Ability to merge users
- Ability to merge establishments
- Ability to locate and correct records with missing required data
- A comprehensive permissions system to allow restricted access to users details
- Ability to manage mailing lists, both public and internal, and their associated users

Benefits:
- Reduced the waiting time to load the system
- Used to clean up >400 user records
- Less checks required by user office team when a user updates their details (now only requires authorising email changes)
- Reduces duplicate user accounts and establishments due to merge function
- Fixes and new features can quickly and easily be rolled out due to being installed on a single server instead of individual's machines

Databases

- The system sits on top of a single database containing all details
  - This allows secure and reliable access to user details
  - Reduces redundancy
- Implements versioning to keep a history of changes
  - Implemented in code by populating a "Thru_Date" column when a record is no longer valid
  - Current details are identified by records where "Thru_Date" is null
- We built a few small tools to maintain data (access through the User Details System interface)
  - Check for missing establishments
  - Check for duplicate emails/user numbers
- A cloned database on a separate cluster for development and testing

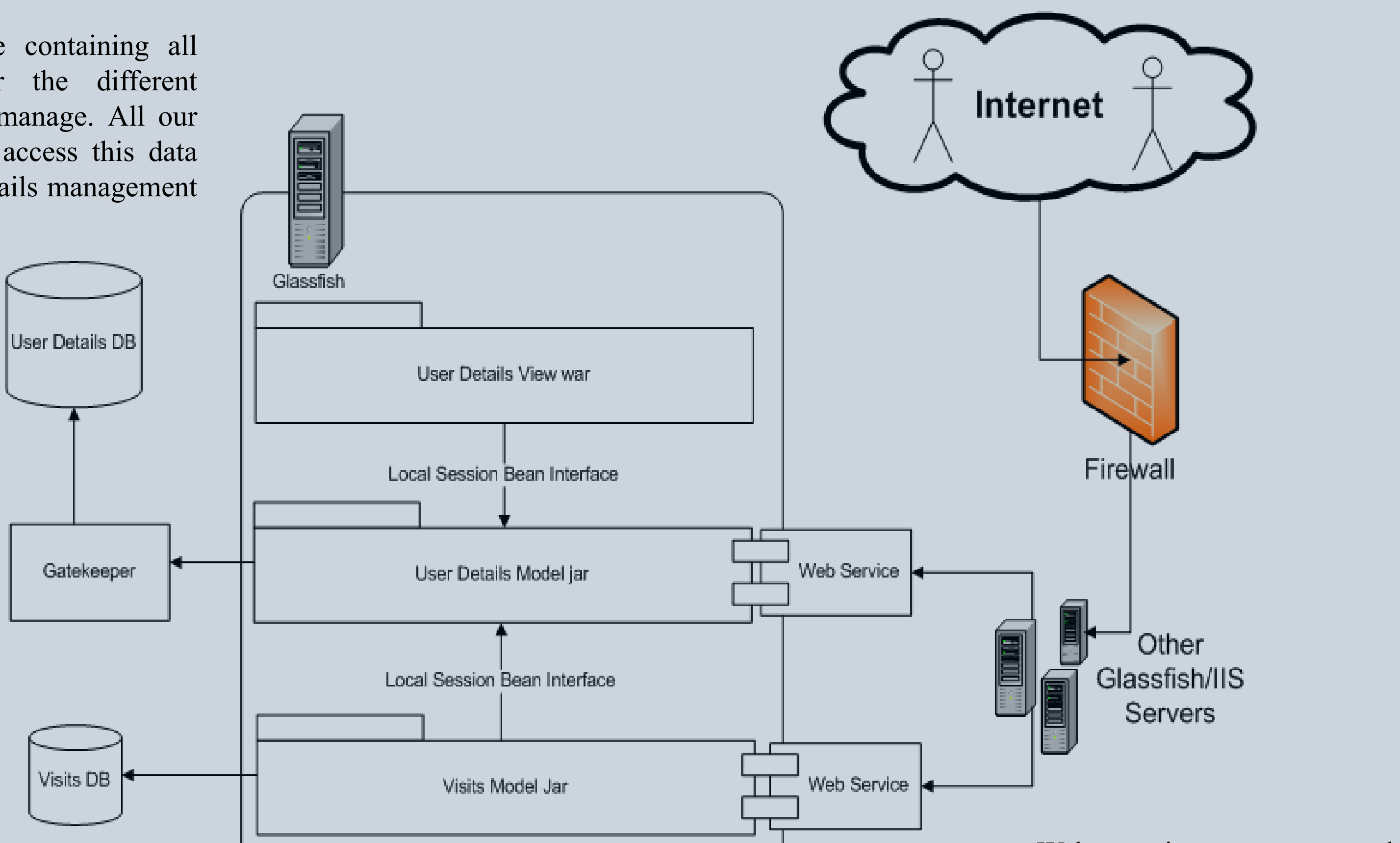
We have recently started looking into methods to scramble the data within our development schema to allow us to keep users data private in the event that we ever need to use some "real" test data off-site.

```sql
-- Create and populate table to contain random characters
-- (drop if already exists e.g. failed to read the drop statement on previous run)
BEGIN
    EXECUTE IMMEDIATE 'DROP TABLE temp';
    EXCEPTION WHEN OTHERS THEN NULL;
END;

CREATE TABLE temp (temp_id number(38) primary key, l_chars varchar(255),
                    u_chars varchar(255), nums varchar(255));
insert into temp values (0, 'enimadminimveniamquisnostr','ENIMADMINIMVENIAMQUISNOSTR',
                    '4859670213');
insert into temp values (1, 'udexercitationullamcorpers','UDEXERCITATIONULLAMCORPERS',
                    '2485976103');
insert into temp values (2, 'uscipitlobortisnislutaliqu','USCIPITLOBORTISNISLUTALIQU',
                    '6578942013');

-- Create a function to return a random collection of characters to be used
-- to replace data
CREATE OR REPLACE FUNCTION randomChars RETURN VARCHAR IS
    lChars VARCHAR2(255);
    uChars VARCHAR2(255);
    nChars VARCHAR2(255);
    rString VARCHAR2(255);
BEGIN
    select l_Chars INTO lChars FROM (select l_Chars from temp order by dbms_random.value)
        where rownum = 1 ;
    select u_Chars INTO uChars FROM (select u_Chars from temp order by dbms_random.value)
        where rownum = 1 ;
    select nums INTO nChars FROM (select nums from temp order by dbms_random.value)
        where rownum = 1 ;
    rString:= lChars || uChars || nChars;
    RETURN rString;
END;

-- Update person
update Person
    set
        PERSON.EMAIL = translate((select Person.EMAIL from Person),
            'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789',
            randomChars );

-- Drop the temporary table containing random chars
drop table temp;
```

Interfaces and Web services

A single database containing all users details for the different facilities that we manage. All our systems can then access this data via the Visitor Details management system API.

The Gatekeeper restricts access to data based on the role of the logged in user. This also handles filtering out users based on facility.
The bean is annotated with the `@Interceptors` annotation, which allow it to call methods from the Gatekeeper class before its own method is called. Any methods where we do not want the interceptor to run are annotated with `@ExcludeClassInterceptors`. For example, access to personal data is restricted to a few people, so the interceptor is called. However, the log in method can be called by anyone, so the interceptor is excluded.



```java
@Interceptors(GateKeeper.class)
@Stateless(name = "UserOfficeFacade", mappedName = "stfc-UserOffice-UserOfficeFacade")
public class UserOfficeFacadeBean implements UserOfficeFacade, UserOfficeLocal {
```

Extracts from the Bean. Class definition (above) and a method which doesn't use the Gatekeeper (below).

```java
@ExcludeClassInterceptors
public EntityManager getEm(){
    return em;
}
```

The method which gets called in the GateKeeper class

```java
@AroundInvoke
public Object logAccess(InvocationContext ic) throws Exception{
```

Web services are exposed internally to allow us to spread our systems across multiple virtual servers within out network and still allow sharing of information. This is also used by Central IT team to authenticate users for wireless access.

Cross Facility

- Our system is currently used to manage users from
  - ISIS
  - CLF
  - MICE experiment
  - Will be shortly rolled out to RCaH
- It has scope to handle more facilities in the future.
- Using a single system and database enables better cross-facility work and can provide more reliable up-to-date information between facilities. It also allows user office staff responsible for more than one facility to use a single tool
- Based on the logged in user, the gatekeeper ensures that only the relevant users' details are shown, e.g. the MICE admin team will be restricted to only managing MICE users (which may include users who are associated with other facilities as well).

Science & Technology Facilities Council ISIS
central laser facility
Research Complex at Harwell

Find out More

Business Applications Team
users@stfc.ac.uk
www.isis.stfc.ac.uk/groups/computing/business/

jQuery: http://jquery.com/
FlexBox: http://flexbox.codeplex.com/