Contents

1 REF Manager v2.0 - Complete Documentation

Research Excellence Framework Submission Management System

Version: 2.0.0
Last Updated: November 3, 2025
Author: George Tsoulas, Department of Language and Linguistic Science, University of York

---

1.1 Table of Contents

1.2 Overview

REF Manager is a comprehensive Django-based web application designed to streamline the management of Research Excellence Framework (REF) submissions for UK academic institutions. The system provides robust tools for tracking research outputs, managing staff information, coordinating review processes, and generating detailed reports.

1.2.1 Key Benefits

- **Centralized Management**: Single platform for all REF-related activities
- **Multi-user Access**: Role-based permissions for different user types
- **Quality Tracking**: Monitor quality profiles using REF rating system (4*, 3*, 2*, 1*)
- **Review Coordination**: Manage both internal panel and external critical friend reviews
- **Task Management**: Track all REF-related tasks with priorities and deadlines
- **Professional Reports**: Generate LaTeX/PDF reports for various stakeholders
- **Data Import/Export**: Bulk operations via CSV and Excel formats
- **Employment Tracking**: Maintain complete staff history including former employees
- **Flexible Categorization**: Classify colleagues by employment status and research role

---

1.3 What's New in v2.0

1.3.1 Major Features Added

1.3.1.1 1. Employment Status Tracking

- **Current vs Former Staff**: Distinguish between current and former employees

- **Employment End Dates**: Track when employment ended for historical records
- **Data Preservation**: Keep research outputs linked to former staff for REF reporting
- **Smart Filtering**: Forms and dropdowns automatically show only current staff
- **Status Badges**: Visual indicators for employment status throughout the interface

1.3.1.2 2. Enhanced Colleague Categories Nine distinct categories for comprehensive staff classification:

- **Independent Researcher**: Established researchers with independent research programs
- **Non-Independent Researcher**: Post-doctoral researchers and early career staff
- **Post-Doctoral Researcher**: Specific category for post-docs
- **Research Assistant**: Research support staff
- **Academic Staff**: Teaching and research faculty
- **Support Staff**: Administrative and technical support
- **Current Employee**: Generic current staff category
- **Former Employee**: Past employees (for legacy data)
- **Co-author (External)**: External collaborators and co-authors

Benefits: - Precise identification of post-docs for REF submission rules - Clear distinction between independent and non-independent researchers - Better tracking of external collaborators - Improved reporting on staff composition

1.3.1.3 3. Internal Panel System A complete internal evaluation system separate from external Critical Friends:

Features: - Panel member management with roles (Chair, Member, Specialist, External Liaison) - Internal assignment tracking with status monitoring - Quality rating recommendations aligned with REF standards - Dashboard statistics for panel workload and progress - Complete CRUD operations for panel members and assignments - Review status tracking (Assigned, In Progress, Completed, Deferred)

Why Separate from Critical Friends? - Internal Panel: University's own evaluation committee - Critical Friends: External reviewers from other institutions - Different workflows and reporting requirements - Separate dashboard widgets for each

1.3.1.4 4. Task Management System Comprehensive task tracking for all REF-related activities:

Task Features: - Multiple categories: Administrative, Submission, Review, Meeting, Documentation, Deadline, Other - Four priority levels: Low, Medium, High, Urgent - Status tracking: Pending, In Progress, Completed, Cancelled - User assignment and ownership - Due dates and start dates - Dashboard widget showing task overview - Complete task list with filtering and search - Task creation, editing, and completion tracking

Perfect for: - Submission deadlines - Meeting schedules - Administrative requirements - Review coordination - Documentation tasks

1.3.1.5 5. CSV Import Functionality

Bulk data import for efficient data management:

Import Capabilities: - Research outputs with full metadata - Colleague information - Duplicate detection and handling - Automatic categorization (external co-authors) - Validation and error reporting - Batch processing

CSV Import Workflow: 1. Prepare CSV file with required columns 2. Upload via web interface 3. System validates data and detects duplicates 4. Review import summary 5. Confirm import 6. Automatic linking to existing colleagues

1.3.1.6 6. Excel Export with Clickable Links

Professional export functionality for review coordination:

Export Features: - Review assignments for Internal Panel and Critical Friends - Clickable hyperlinks to paper PDFs - Color-coded by reviewer type (blue: internal, yellow: external) - Comprehensive filtering options: - By reviewer type (internal/external) - By specific reviewer - By output author - By assignment status - Professional formatting with frozen headers - Includes reviewer contact information, quality ratings, and notes

Use Cases: - Email assignments to reviewers with direct links to papers - Progress tracking spreadsheets - Meeting preparation materials - Report generation

1.3.1.7 7. Request Management Enhancements

Enhanced functionality for handling REF-related requests:

New Features: - Mark requests as completed with timestamp - Delete requests with confirmation - Status tracking (Pending, In Progress, Completed) - Visual status indicators - User-friendly confirmation pages - Success messages and feedback

1.3.1.8 8. Dashboard Improvements

Enhanced dashboard with new widgets:

New Dashboard Components: - Internal Panel Statistics card showing: - Total panel members - Active assignments - Completed reviews - Pending reviews - Average quality ratings - Task Overview widget showing: - Urgent tasks - Overdue tasks - Tasks by status - Tasks by priority - Employment status breakdown - Category distribution charts

---

1.4 □ System Requirements

1.4.1 Minimum Requirements

- **Operating System**: Linux (Ubuntu 20.04+), macOS 10.15+, or Windows 10+
- **Python**: 3.10 or higher (tested up to Python 3.13)
- **RAM**: 2GB minimum (4GB recommended)
- **Disk Space**: 500MB for application + database
- **Browser**: Modern browser (Chrome, Firefox, Safari, Edge - latest 2 versions)

1.4.2 Required Software

1. **Python 3.10+**

   ```
   python3 --version  # Should show 3.10 or higher
   ```

2. **pip** (Python package installer)

   ```
   pip3 --version
   ```

3. **Git** (for version control)

   ```
   git --version
   ```

4. **Database** (Choose one):
   - **SQLite** (included with Python, recommended for development)
   - **PostgreSQL** 12+ (recommended for production)

1.4.3 Optional Software

- **LaTeX** (for PDF report generation)

  ```
  # Ubuntu/Debian
  sudo apt-get install texlive-full

  # macOS
  brew install --cask mactex

  # Windows
  # Download from https://www.tug.org/texlive/
  ```

- **Nginx** (for production deployment)
- **Gunicorn** (for production WSGI server)

1.5 □ Installation

1.5.1 Quick Start (Development)

```
# 1. Clone or download the repository
git clone https://github.com/yourusername/ref-manager.git
cd ref-manager

# 2. Create virtual environment
python3 -m venv venv

# 3. Activate virtual environment
# Linux/macOS:
source venv/bin/activate
# Windows:
venv\Scripts\activate
```

```
# 4. Install dependencies
pip install -r requirements.txt

# 5. Set up environment variables
cp .env.example .env
# Edit .env and set your SECRET_KEY

# 6. Run migrations
python manage.py migrate

# 7. Create superuser
python manage.py createsuperuser

# 8. Load sample data (optional)
python manage.py loaddata sample_data.json

# 9. Run development server
python manage.py runserver

# 10. Access the application
# Open browser to: http://localhost:8000
```

1.5.2 Detailed Installation Steps

1.5.2.1 1. System Preparation Ubuntu/Debian:

```
sudo apt-get update
sudo apt-get install python3 python3-pip python3-venv git
```

macOS:

```
# Install Homebrew if not already installed
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEA

# Install Python and Git
brew install python git
```

Windows: - Download Python from https://www.python.org/downloads/ - Download Git from https://git-scm.com/download/win - During Python installation, check “Add Python to PATH”

1.5.2.2 2. Create Project Directory

```
mkdir -p ~/ref-project-app
cd ~/ref-project-app
```

1.5.2.3 3. Clone/Download Application Option A: Using Git

```
git clone https://github.com/yourusername/ref-manager.git
cd ref-manager
```

Option B: Download ZIP - Download the ZIP file - Extract to `~/ref-project-app/ref-manager` - Navigate to the directory

1.5.2.4 4. Virtual Environment Setup

```
# Create virtual environment
python3 -m venv venv

# Activate (Linux/macOS)
source venv/bin/activate

# Activate (Windows)
venv\Scripts\activate

# Your prompt should now show (venv)
```

1.5.2.5 5. Install Python Packages

```
pip install --upgrade pip
pip install -r requirements.txt
```

Key packages installed: - Django 4.2+ (web framework) - django-crispy-forms (form styling) - crispy-bootstrap4 (Bootstrap 4 integration) - openpyxl (Excel file handling) - python-dotenv (environment variable management) - gunicorn (production WSGI server) - psycopg2-binary (PostgreSQL adapter, if using PostgreSQL)

Python 3.13 Note: If you encounter issues with package installations on Python 3.13, you may need to use the `--break-system-packages` flag:

```
pip install --break-system-packages -r requirements.txt
```

1.5.2.6 6. Environment Configuration

```
# Copy example environment file
cp .env.example .env

# Edit environment file
nano .env  # or use your preferred editor
```

Essential .env settings:

```
# Django Settings
SECRET_KEY=your-secret-key-here-generate-a-new-one
DEBUG=True
ALLOWED_HOSTS=localhost,127.0.0.1

# Database (leave commented for SQLite development)
# DB_ENGINE=django.db.backends.postgresql
# DB_NAME=ref_manager_db
# DB_USER=your_database_user
# DB_PASSWORD=your_database_password
# DB_HOST=localhost
# DB_PORT=5432
```

Generate SECRET_KEY:

```
python -c "from django.core.management.utils import get_random_secret_key; print(g
```

1.5.2.7 7. Database Setup SQLite (Development):

```
# Migrations create the database automatically
python manage.py migrate
```

PostgreSQL (Production):

```
# Create database
sudo -u postgres createdb ref_manager_db
sudo -u postgres createuser ref_manager_user
sudo -u postgres psql

# In psql:
ALTER USER ref_manager_user WITH PASSWORD 'secure_password';
GRANT ALL PRIVILEGES ON DATABASE ref_manager_db TO ref_manager_user;
\q

# Update .env with PostgreSQL settings
# Then run migrations
python manage.py migrate
```

1.5.2.8 8. Create Admin User

```
python manage.py createsuperuser
```

Enter: - Username (e.g., admin) - Email address - Password (enter twice)

1.5.2.9 9. Collect Static Files

```
python manage.py collectstatic --noinput
```

1.5.2.10 10. Load Sample Data (Optional)

```
# If sample data is provided
python manage.py loaddata sample_data.json
```

1.6 ⚙ Configuration

1.6.1 Database Configuration

1.6.1.1 SQLite (Default for Development) No additional configuration needed. Database file created automatically at `db.sqlite3`.

Pros: - Zero configuration - Perfect for development and testing - Single file, easy to backup - No server required

Cons: - Not suitable for production with multiple users - Limited concurrent access - Performance limitations with large datasets

1.6.1.2 PostgreSQL (Recommended for Production) Installation:

Ubuntu/Debian:

```
sudo apt-get install postgresql postgresql-contrib
```

macOS:

```
brew install postgresql
brew services start postgresql
```

Setup:

```
# Switch to postgres user
sudo -u postgres psql

# Create database and user
CREATE DATABASE ref_manager_db;
CREATE USER ref_manager_user WITH PASSWORD 'your_secure_password';
ALTER ROLE ref_manager_user SET client_encoding TO 'utf8';
ALTER ROLE ref_manager_user SET default_transaction_isolation TO 'read committed';
ALTER ROLE ref_manager_user SET timezone TO 'UTC';
GRANT ALL PRIVILEGES ON DATABASE ref_manager_db TO ref_manager_user;

# Exit psql
\q
```

Configure .env:

```
DB_ENGINE=django.db.backends.postgresql
DB_NAME=ref_manager_db
DB_USER=ref_manager_user
DB_PASSWORD=your_secure_password
DB_HOST=localhost
DB_PORT=5432
```

Run migrations:

```
python manage.py migrate
```

1.6.2 Email Configuration (Optional)

For task reminders and notifications:

```
EMAIL_BACKEND=django.core.mail.backends.smtp.EmailBackend
EMAIL_HOST=smtp.gmail.com
EMAIL_PORT=587
EMAIL_USE_TLS=True
EMAIL_HOST_USER=your-email@example.com
EMAIL_HOST_PASSWORD=your-app-specific-password
```

Gmail App Password: 1. Go to Google Account settings 2. Security → 2-Step Verification 3. App passwords → Generate password 4. Use generated password in .env

1.6.3 Static Files Configuration

```
# settings.py (already configured)
STATIC_URL = '/static/'
STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
STATICFILES_DIRS = [os.path.join(BASE_DIR, 'static')]
```

Collect static files:

```
python manage.py collectstatic --noinput
```

1.6.4 Media Files Configuration

```
# settings.py (already configured)
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

Create media directory:

```
mkdir -p media/pdfs
chmod 755 media
```

1.7 □ Features

1.7.1 1. Dashboard

Overview Widgets: - **Submission Statistics** - Total outputs - Quality profile (4*, 3*, 2*, 1*, Unclassified) - Average quality score - Outputs by Unit of Assessment (UoA)

- **Staff Summary**
  - Total colleagues (current)
  - Former staff count
  - Category breakdown
  - Outputs per staff member
  - Employment status distribution
- **Review Progress**
  - Critical Friends workload
  - Pending reviews
  - Completed reviews
  - Average review time
- **Internal Panel Statistics** □ NEW in v2.0
  - Total panel members
  - Active assignments
  - Completed internal reviews
  - Quality rating distribution

  - Pending assessments
- **Task Overview** □ NEW in v2.0
  - Urgent tasks
  - Overdue tasks
  - Tasks by status
  - Tasks by priority
  - Upcoming deadlines
- **Recent Activity**
  - Latest outputs added
  - Recent reviews
  - New requests
  - System updates

1.7.2 2. Colleague Management

Features: - Add, edit, and view colleague information - Employment status tracking (current/former) □ NEW in v2.0 - Nine colleague categories for precise classification □ NEW in v2.0 - Track employment end dates - Link to research outputs - Unit of Assessment assignment - Contact information - Research interests - Profile photos

Colleague Categories: □ NEW in v2.0 1. **Independent Researcher**: Established researchers 2. **Non-Independent Researcher**: Post-docs, early career 3. **Post-Doctoral Researcher**: Specific post-doc category 4. **Research Assistant**: Research support 5. **Academic Staff**: Teaching & research faculty 6. **Support Staff**: Administrative support 7. **Current Employee**: Generic current staff 8. **Former Employee**: Past employees 9. **Co-author (External)**: External collaborators

List View: - Filter by employment status - Filter by category - Sort by name, UoA, output count - Search functionality - Category statistics - Employment status badges

Detail View: - Full colleague profile - Research outputs list - Review assignments - Quality profile - Export colleague data

1.7.3 3. Research Output Management

Output Fields: - Title, authors, publication venue - Publication type (Journal Article, Book, Chapter, etc.) - DOI and URLs - PDF file upload - Publication date - Quality rating (4*, 3*, 2*, 1*, Unclassified) - REF eligibility status - Unit of Assessment - Keywords and abstract - Submission notes

Features: - Add, edit, delete outputs - Bulk import via CSV □ NEW in v2.0 - Filter by quality, author, UoA, type - Search functionality - PDF viewer integration - DOI linkage - Duplicate detection - Quality statistics

CSV Import: □ NEW in v2.0 - Upload CSV file with output metadata - Automatic validation - Duplicate detection - Link to existing colleagues - Auto-categorize external co-authors - Import summary and error reporting

1.7.4 4. Critical Friends System

External Reviewer Management: - Add critical friends (external reviewers) - Contact information and expertise - Institution and role - Availability status - Review history

Assignment Management: - Assign outputs to critical friends - Track review status - Set due dates and priorities - Record ratings and feedback - Completion dates - Review notes

Export Functionality: NEW in v2.0 - Export assignments to Excel - Clickable links to paper PDFs - Filter by reviewer, author, status - Color-coded formatting - Professional layout for distribution

1.7.5 5. Internal Panel System NEW in v2.0

Panel Member Management: - Add internal panel members from colleague list - Assign panel roles: - Chair - Member - Specialist - External Liaison - Track expertise areas - View workload and assignments

Internal Review Assignments: - Assign outputs to internal panel members - Track internal review status: - Assigned - In Progress - Completed - Deferred - Record quality rating recommendations - Set review deadlines - Priority levels - Internal review notes and feedback

Dashboard Integration: - Internal Panel statistics widget - Workload distribution - Completion rates - Average quality ratings - Pending reviews alert

Why Separate from Critical Friends? - **Internal Panel**: University's evaluation committee - **Critical Friends**: External peer reviewers - Different review workflows - Separate reporting requirements - Internal vs external perspective tracking

1.7.6 6. Request Management

Request Types: - Submission queries - Quality disputes - Administrative requests - External submissions - Documentation requests

Features: - Create and track requests - Assign to colleagues - Set priorities - Status tracking (Pending, In Progress, Completed) - Due dates - Linked outputs - Mark as completed NEW in v2.0 - Delete with confirmation NEW in v2.0 - Request history - Email notifications

Status Management: NEW in v2.0 - Mark requests as done with timestamp - Completion confirmation page - Visual status indicators - Automatic status updates - Delete requests with warning confirmation

1.7.7 7. Task Management NEW in v2.0

Comprehensive Task Tracking:

Task Categories: - Administrative - Submission - Review - Meeting - Documentation - Deadline - Other

Priority Levels: - Low - Medium - High - Urgent

Status Tracking: - Pending - In Progress - Completed - Cancelled

Task Features: - Create, edit, and delete tasks - Assign to users - Set due dates and start dates - Priority indicators - Status badges - Task description and notes - Completion tracking - Dashboard widget - Overdue task alerts - Filter and search functionality - Task list views

Dashboard Integration: - Urgent tasks widget - Overdue tasks alert - Tasks by status chart - Tasks by priority breakdown - Quick task creation

1.7.8 8. Report Generation

Available Reports: - Submission Overview - Quality Profile Analysis - Staff Progress Report - Review Status Report - Comprehensive Report (all combined)

Export Formats: - LaTeX source (.tex) - PDF (via LaTeX compilation) - Three document styles: - Article - Report - Beamer presentation

Report Content: - Executive summary - Quality distribution charts - Staff contribution analysis - Output listings by quality - Review progress metrics - Critical concerns - Strategic recommendations

1.7.9 9. Data Import/Export ENHANCED in v2.0

Import: - **CSV Import**: NEW - Research outputs with full metadata - Colleague information - Bulk upload interface - Validation and error reporting - Duplicate detection - Auto-categorization

Export: - **Excel Export with Links**: NEW - Review assignments - Clickable paper links - Color-coded by reviewer type - Professional formatting - Multiple filter options

- **CSV Export**:
  - Outputs, colleagues, assignments
  - Custom field selection
- **LaTeX/PDF Reports**:
  - Professional academic reports
  - Multiple templates
- **Data Backup**:
  - Complete database dump
  - JSON format

1.7.10 10. User Management

User Roles: - **Superuser/Admin**: Full system access - **Staff User**: Add/edit outputs, manage reviews - **View-Only User**: Read-only access to reports

Authentication: - Secure login system - Password reset functionality - Session management - Permission-based access control

Admin Panel: - Django admin interface - Bulk operations - Advanced filtering - Data import/export - System configuration

---

1.8 Running the Application

1.8.1 Development Server

Start the server:

```
# Activate virtual environment
source venv/bin/activate  # Linux/macOS
# or
venv\Scripts\activate  # Windows

# Run development server
python manage.py runserver

# Access at http://localhost:8000
```

Run on different port:

```
python manage.py runserver 8080
```

Allow external access:

```
python manage.py runserver 0.0.0.0:8000
```

1.8.2 Background Service (systemd)

Create service file:

```
sudo nano /etc/systemd/system/ref-manager.service
```

```
[Unit]
Description=REF Manager Django Application
After=network.target

[Service]
Type=simple
User=your-username
WorkingDirectory=/home/your-username/ref-project-app/ref-manager
Environment="PATH=/home/your-username/ref-project-app/ref-manager/venv/bin"
ExecStart=/home/your-username/ref-project-app/ref-manager/venv/bin/gunicorn \
    --workers 3 \
    --bind 0.0.0.0:8000 \
    ref_manager.wsgi:application
Restart=always
RestartSec=10

[Install]
WantedBy=multi-user.target
```

Manage service:

```
# Enable service
sudo systemctl enable ref-manager

# Start service
sudo systemctl start ref-manager

# Check status
sudo systemctl status ref-manager

# View logs
sudo journalctl -u ref-manager -f

# Restart service (after code changes)
sudo systemctl restart ref-manager

# Stop service
sudo systemctl stop ref-manager
```

1.8.3 Using Screen (Alternative)

```
# Install screen
sudo apt-get install screen

# Start new screen session
screen -S ref-manager

# Run server
python manage.py runserver 0.0.0.0:8000

# Detach: Press Ctrl+A, then D

# Reattach later
screen -r ref-manager

# List sessions
screen -ls
```

1.9 □ Production Deployment

1.9.1 Using Gunicorn + Nginx

1.9.1.1 1. Install Requirements

```
# Install Nginx
sudo apt-get install nginx

# Install Gunicorn (should be in requirements.txt)
pip install gunicorn
```

1.9.1.2 2. Configure Gunicorn Test Gunicorn:

```
cd ~/ref-project-app/ref-manager
source venv/bin/activate
gunicorn --bind 0.0.0.0:8000 ref_manager.wsgi:application
```

Create Gunicorn configuration:

```
nano gunicorn_config.py
```

```
import multiprocessing

bind = "127.0.0.1:8000"
workers = multiprocessing.cpu_count() * 2 + 1
worker_class = "sync"
worker_connections = 1000
timeout = 30
keepalive = 2
errorlog = "/home/your-username/ref-project-app/ref-manager/logs/gunicorn-error.log"
accesslog = "/home/your-username/ref-project-app/ref-manager/logs/gunicorn-access.log"
loglevel = "info"
```

Create logs directory:

```
mkdir -p logs
```

1.9.1.3 3. Configure Nginx Create Nginx configuration:

```
sudo nano /etc/nginx/sites-available/ref-manager
```

```
server {
    listen 80;
    server_name your-domain.com www.your-domain.com;

    client_max_body_size 20M;

    location = /favicon.ico { access_log off; log_not_found off; }

    location /static/ {
     alias /home/your-username/ref-project-app/ref-manager/staticfiles/;
    }

    location /media/ {
        alias /home/your-username/ref-project-app/ref-manager/media/;
    }

    location / {
        proxy_pass http://127.0.0.1:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
```

```
    }
}
```

Enable site:

```
sudo ln -s /etc/nginx/sites-available/ref-manager /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl restart nginx
```

1.9.1.4 4. Update Django Settings

```
# settings.py
DEBUG = False
ALLOWED_HOSTS = ['your-domain.com', 'www.your-domain.com', 'server-ip']

# Security settings
SECURE_SSL_REDIRECT = True
SESSION_COOKIE_SECURE = True
CSRF_COOKIE_SECURE = True
SECURE_BROWSER_XSS_FILTER = True
SECURE_CONTENT_TYPE_NOSNIFF = True
```

1.9.1.5 5. SSL/HTTPS with Let’s Encrypt

```
# Install Certbot
sudo apt-get install certbot python3-certbot-nginx

# Get certificate
sudo certbot --nginx -d your-domain.com -d www.your-domain.com

# Test renewal
sudo certbot renew --dry-run
```

1.9.1.6 6. Final Steps

```
# Collect static files
python manage.py collectstatic --noinput

# Run migrations
python manage.py migrate

# Create superuser if needed
python manage.py createsuperuser

# Set proper permissions
chmod -R 755 /home/your-username/ref-project-app/ref-manager
```

1.9.2 Using Docker (Optional)

Dockerfile:

```
FROM python:3.11-slim

ENV PYTHONDONTWRITEBYTECODE=1
ENV PYTHONUNBUFFERED=1

WORKDIR /app

RUN apt-get update && apt-get install -y \
    postgresql-client \
    && rm -rf /var/lib/apt/lists/*

COPY requirements.txt /app/
RUN pip install --no-cache-dir -r requirements.txt

COPY . /app/

RUN python manage.py collectstatic --noinput

EXPOSE 8000

CMD ["gunicorn", "--bind", "0.0.0.0:8000", "ref_manager.wsgi:application"]
```

docker-compose.yml:

```
version: '3.8'

services:
  db:
    image: postgres:13
    volumes:
      - postgres_data:/var/lib/postgresql/data
    environment:
      POSTGRES_DB: ref_manager_db
      POSTGRES_USER: ref_manager_user
      POSTGRES_PASSWORD: secure_password

  web:
    build: .
    command: gunicorn ref_manager.wsgi:application --bind 0.0.0.0:8000
    volumes:
      - .:/app
      - static_volume:/app/staticfiles
      - media_volume:/app/media
    ports:
      - "8000:8000"
    environment:
      - DEBUG=False
      - SECRET_KEY=your-secret-key
      - DB_ENGINE=django.db.backends.postgresql
```

```
      - DB_NAME=ref_manager_db
      - DB_USER=ref_manager_user
      - DB_PASSWORD=secure_password
      - DB_HOST=db
      - DB_PORT=5432
    depends_on:
      - db

  nginx:
    image: nginx:latest
    ports:
      - "80:80"
    volumes:
      - ./nginx.conf:/etc/nginx/conf.d/default.conf
      - static_volume:/app/staticfiles
      - media_volume:/app/media
    depends_on:
      - web

volumes:
  postgres_data:
  static_volume:
  media_volume:
```

Run with Docker:

```
docker-compose up -d
docker-compose exec web python manage.py migrate
docker-compose exec web python manage.py createsuperuser
```

1.10 Maintenance & Backup

1.10.1 Regular Maintenance Tasks

Weekly: - Check application logs for errors - Review disk space usage - Verify backup integrity - Check for Django security updates

Monthly: - Update Python packages - Review and archive old data - Check database size and performance - Update documentation

Quarterly: - Full system review - Performance optimization - Security audit - User feedback review

1.10.2 Backup Procedures

1.10.2.1 Database Backup SQLite:

```
# Backup database
cp db.sqlite3 backups/db.sqlite3.$(date +%Y%m%d-%H%M%S)
```

```
# Automated backup script
#!/bin/bash
BACKUP_DIR="$HOME/ref-project-app/backups"
mkdir -p "$BACKUP_DIR"
TIMESTAMP=$(date +%Y%m%d-%H%M%S)
cp "$HOME/ref-project-app/ref-manager/db.sqlite3" "$BACKUP_DIR/db.sqlite3.$TIMEST
find "$BACKUP_DIR" -name "db.sqlite3.*" -mtime +30 -delete
```

PostgreSQL:

```
# Backup
pg_dump -U ref_manager_user ref_manager_db > backups/ref_manager_$(date +%Y%m%d-
%H%M%S).sql

# Restore
psql -U ref_manager_user ref_manager_db < backups/ref_manager_20251103.sql

# Automated backup
#!/bin/bash
BACKUP_DIR="$HOME/ref-project-app/backups"
mkdir -p "$BACKUP_DIR"
TIMESTAMP=$(date +%Y%m%d-%H%M%S)
pg_dump -U ref_manager_user ref_manager_db | gzip > "$BACKUP_DIR/ref_manager_$TIME
find "$BACKUP_DIR" -name "ref_manager_*.sql.gz" -mtime +30 -delete
```

1.10.2.2 Media Files Backup

```
# Backup media files
tar -czf backups/media_$(date +%Y%m%d-%H%M%S).tar.gz media/

# Automated script
#!/bin/bash
BACKUP_DIR="$HOME/ref-project-app/backups"
mkdir -p "$BACKUP_DIR"
TIMESTAMP=$(date +%Y%m%d-%H%M%S)
cd "$HOME/ref-project-app/ref-manager"
tar -czf "$BACKUP_DIR/media_$TIMESTAMP.tar.gz" media/
find "$BACKUP_DIR" -name "media_*.tar.gz" -mtime +30 -delete
```

1.10.2.3 Complete System Backup

```
#!/bin/bash
# Full backup script

BACKUP_DIR="$HOME/ref-project-app/backups"
TIMESTAMP=$(date +%Y%m%d-%H%M%S)
PROJECT_DIR="$HOME/ref-project-app/ref-manager"

mkdir -p "$BACKUP_DIR"
cd "$PROJECT_DIR"

echo "Backing up database..."
```

```
if [ -f "db.sqlite3" ]; then
    cp db.sqlite3 "$BACKUP_DIR/db.sqlite3.$TIMESTAMP"
else
   pg_dump -U ref_manager_user ref_manager_db | gzip > "$BACKUP_DIR/database_$TIMES
fi

echo "Backing up media files..."
tar -czf "$BACKUP_DIR/media_$TIMESTAMP.tar.gz" media/

echo "Backing up configuration..."
cp .env "$BACKUP_DIR/env_$TIMESTAMP.txt"

echo "Creating archive..."
tar -czf "$BACKUP_DIR/complete_backup_$TIMESTAMP.tar.gz" \
    --exclude='venv' \
    --exclude='*.pyc' \
    --exclude='__pycache__' \
    --exclude='staticfiles' \
    -C "$HOME/ref-project-app" ref-manager

echo "Cleaning old backups (keeping last 30 days)..."
find "$BACKUP_DIR" -type f -mtime +30 -delete

echo "Backup completed: $BACKUP_DIR/complete_backup_$TIMESTAMP.tar.gz"
```

Schedule with cron:

```
# Edit crontab
crontab -e

# Add daily backup at 2 AM
0 2 * * * /path/to/backup-script.sh >> /path/to/backup.log 2>&1
```

1.10.3 Restore Procedures

From SQLite backup:

```
cp backups/db.sqlite3.20251103-143000 db.sqlite3
```

From PostgreSQL backup:

```
# Drop and recreate database
psql -U postgres
DROP DATABASE ref_manager_db;
CREATE DATABASE ref_manager_db;
GRANT ALL PRIVILEGES ON DATABASE ref_manager_db TO ref_manager_user;
\q

# Restore
gunzip -c backups/ref_manager_20251103.sql.gz | psql -U ref_manager_user ref_manag
```

From complete backup:

```
cd ~/ref-project-app
```

```
tar -xzf backups/complete_backup_20251103.tar.gz
cd ref-manager
source venv/bin/activate
python manage.py migrate
sudo systemctl restart ref-manager
```

1.10.4 Log Management

View logs:

```
# Django logs
tail -f logs/django.log

# Gunicorn logs
tail -f logs/gunicorn-access.log
tail -f logs/gunicorn-error.log

# Nginx logs
sudo tail -f /var/log/nginx/access.log
sudo tail -f /var/log/nginx/error.log

# System service logs
sudo journalctl -u ref-manager -f
```

Rotate logs:

```
# Create logrotate configuration
sudo nano /etc/logrotate.d/ref-manager
```

```
/home/your-username/ref-project-app/ref-manager/logs/*.log {
    daily
    rotate 30
    compress
    delaycompress
    notifempty
    missingok
    create 0644 your-username your-username
}
```

1.10.5 Update Procedures

Update Application Code:

```
cd ~/ref-project-app/ref-manager

# Backup first
./backup-script.sh

# Pull updates (if using git)
git pull origin main

# Activate virtual environment
```

```
source venv/bin/activate

# Update dependencies
pip install -r requirements.txt

# Run migrations
python manage.py migrate

# Collect static files
python manage.py collectstatic --noinput

# Restart service
sudo systemctl restart ref-manager
```

Update Python Packages:

```
source venv/bin/activate

# Check outdated packages
pip list --outdated

# Update specific package
pip install --upgrade package-name

# Update all packages (use caution)
pip install --upgrade -r requirements.txt

# Test application
python manage.py check
python manage.py test
```

1.11 Troubleshooting

1.11.1 Common Issues

1.11.1.1 1. Python 3.13 Compatibility

Problem: Decimal arithmetic issues in Python 3.13

Error:

```
TypeError: unsupported operand type(s) for *: 'decimal.Decimal' and 'float'
```

Solution: Convert Decimal to float in calculations:

```
# In models.py
@property
def required_outputs(self):
    if self.employment_status == 'current':
        return float(self.fte) * 2.5  # Convert Decimal to float
    return 0
```

1.11.1.2 2. Migration Issues **Problem:** Migration conflicts or errors

Solution:

```
# Show migrations
python manage.py showmigrations

# Fake a migration if needed
python manage.py migrate core --fake 0001

# Reset migrations (CAREFUL - DEVELOPMENT ONLY)
find . -path "*/migrations/*.py" -not -name "__init__.py" -delete
find . -path "*/migrations/*.pyc" -delete
python manage.py makemigrations
python manage.py migrate
```

1.11.1.3 3. Static Files Not Loading **Problem:** CSS/JS not appearing

Solution:

```
# Collect static files
python manage.py collectstatic --noinput

# Check settings
# DEBUG=True → Django serves static files
# DEBUG=False → Nginx serves static files

# Verify Nginx configuration
sudo nginx -t
sudo systemctl restart nginx
```

1.11.1.4 4. Permission Errors **Problem:** Permission denied errors

Solution:

```
# Fix file permissions
chmod -R 755 ~/ref-project-app/ref-manager

# Fix media directory
chmod -R 755 media

# Fix database (if SQLite)
chmod 664 db.sqlite3
chmod 775 .
```

1.11.1.5 5. Database Connection Issues **Problem:** Can’t connect to PostgreSQL

Solution:

```
# Check PostgreSQL is running
sudo systemctl status postgresql
```

```
# Test connection
psql -U ref_manager_user -d ref_manager_db -h localhost

# Check pg_hba.conf
sudo nano /etc/postgresql/13/main/pg_hba.conf
# Ensure: local all all peer → trust (for development)

# Restart PostgreSQL
sudo systemctl restart postgresql
```

1.11.1.6 6. Import Errors **Problem:** Module not found errors

Solution:

```
# Verify virtual environment is activated
which python  # Should show venv path

# Reinstall requirements
pip install --force-reinstall -r requirements.txt

# Check Python path
python -c "import sys; print('\n'.join(sys.path))"
```

1.11.1.7 7. Template Errors **Problem:** TemplateSyntaxError or template not found

Solution:

```
# Check template directories in settings.py
# Verify TEMPLATES configuration

# Check template exists
find . -name "template_name.html"

# Check template syntax
python manage.py check --tag templates
```

1.11.1.8 8. Form Errors **Problem:** Forms not saving or validation errors

Solution:

```
# Check model definitions match form fields
# Verify required fields have values
# Check form validation in views

# Debug in shell
python manage.py shell
from core.forms import YourForm
form = YourForm(data={'field': 'value'})
form.is_valid()
form.errors
```

1.11.1.9 9. CSV Import Failures **Problem:** CSV import not working

Solution: - Verify CSV encoding is UTF-8 - Check column headers match expected names - Ensure required fields are present - Check for special characters in data - Verify file size is within limits

1.11.1.10 10. Excel Export Issues **Problem:** Excel export fails or corrupted

Solution:

```
# Verify openpyxl is installed
pip list | grep openpyxl

# Reinstall if needed
pip install --force-reinstall openpyxl

# Check permissions for output directory
ls -la /tmp
```

1.11.2 Debug Mode

Enable debug mode temporarily:

```
# settings.py
DEBUG = True
DEBUG_PROPAGATE_EXCEPTIONS = True

# In views.py for detailed error pages
import traceback
try:
    # your code
except Exception as e:
    print(traceback.format_exc())
    raise
```

Django Debug Toolbar:

```
pip install django-debug-toolbar

# settings.py
INSTALLED_APPS += ['debug_toolbar']
MIDDLEWARE += ['debug_toolbar.middleware.DebugToolbarMiddleware']
INTERNAL_IPS = ['127.0.0.1']
```

1.11.3 Getting Help

Check logs first:

```
# Application logs
tail -n 100 logs/django.log

# Server logs
```

```
sudo journalctl -u ref-manager -n 100

# Nginx logs
sudo tail -n 100 /var/log/nginx/error.log
```

Run Django checks:

```
python manage.py check
python manage.py check --deploy  # Production checks
```

Test database:

```
python manage.py dbshell
```

Interactive shell:

```
python manage.py shell

# Test imports
from core.models import *
from core.views import *
```

1.12 Support

1.12.1 Documentation Resources

- **Quick Start Guide**: `QUICK-START-GUIDE.md`
- **User Guide**: `USER-GUIDE.md`
- **Technical Documentation**: `TECHNICAL-DOCUMENTATION.md`
- **Troubleshooting Guide**: `TROUBLESHOOTING.md`
- **Changelog**: `CHANGELOG.md`

1.12.2 Getting Help

Internal Support: - System Administrator: [your-email@york.ac.uk] - Department IT Support: [it-support@york.ac.uk]

External Resources: - Django Documentation: https://docs.djangoproject.com/ - REF Guidelines: https://www.ref.ac.uk/ - Stack Overflow: Tag your questions with [django]

1.12.3 Reporting Issues

When reporting issues, include: - Django version: `python manage.py version` - Python version: `python --version` - Operating system - Error messages (full traceback) - Steps to reproduce - Expected vs actual behavior

1.12.4 Feature Requests

Submit feature requests via: - GitHub Issues (if using GitHub) - Email to system administrator - Department meetings

---

1.13 License

Copyright © 2025 George Tsoulas, University of York

This software is developed for internal use at the Department of Language and Linguistic Science, University of York.

1.14 Acknowledgments

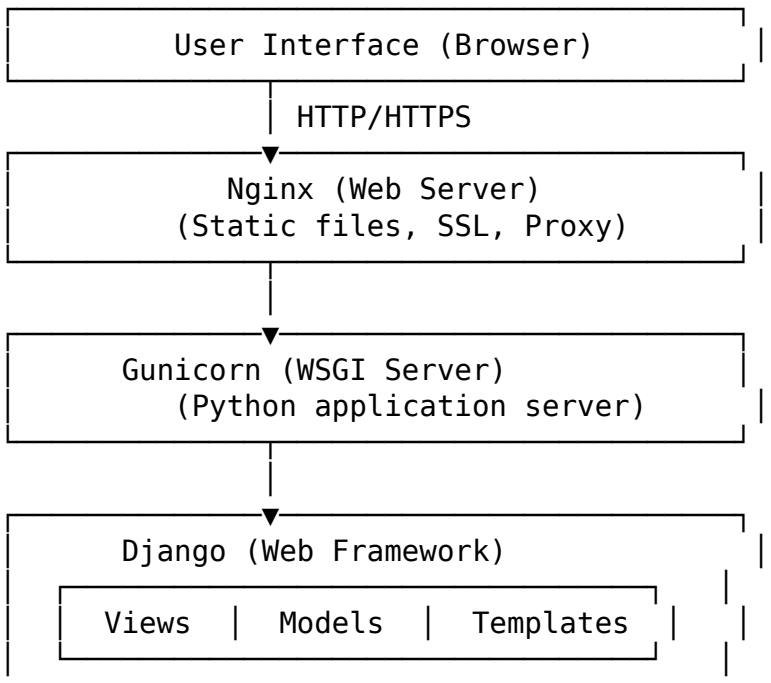
Developed for the Department of Language and Linguistic Science, University of York, to support REF 2029 submission management.

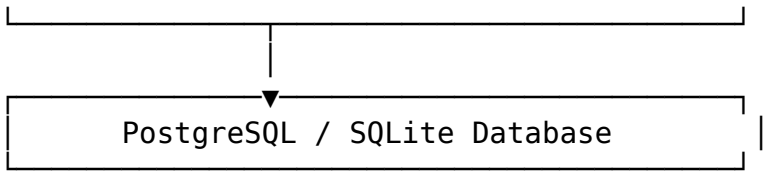
Special thanks to: - Department staff for testing and feedback - University IT services for infrastructure support - Django community for excellent framework and documentation

---

1.15 Appendix

1.15.1 System Architecture





1.15.2 Technology Stack

Backend: - Python 3.10+ - Django 4.2+ - Gunicorn (WSGI server)

Frontend: - HTML5 - CSS3 (Bootstrap 4) - JavaScript (jQuery)

Database: - SQLite (development) - PostgreSQL (production)

Server: - Nginx (web server) - Linux (Ubuntu 20.04+)

Additional: - LaTeX (report generation) - OpenPyXL (Excel export) - Python-dotenv (configuration)

1.15.3 File Structure

```
ref-manager/
├── core/                          # Main application
│   ├── migrations/                # Database migrations
│   ├── models.py                  # Data models
│   ├── views.py                   # View functions
│   ├── views_export.py            # Export functionality (v2.0)
│   ├── forms.py                   # Form definitions
│   ├── urls.py                    # URL routing
│   ├── admin.py                   # Admin configuration
│   └── templatetags/              # Custom template filters
├── ref_manager/                   # Project settings
│   ├── settings.py                # Django configuration
│   ├── urls.py                    # Root URL config
│   └── wsgi.py                    # WSGI application
├── templates/                     # HTML templates
│   ├── base.html                  # Base template
│   ├── core/                      # App templates
│   │   ├── dashboard.html
│   │   ├── colleague_list.html
│   │   ├── output_list.html
│   │   ├── task_list.html         # Task management (v2.0)
│   │   └── ...
│   └── registration/              # Auth templates
├── static/                        # Static files
│   ├── css/
│   ├── js/
│   └── img/
├── media/                         # Uploaded files
│   └── pdfs/                      # Research paper PDFs
├── logs/                          # Application logs
```

```
├── backups/                    # Backup directory
├── venv/                       # Virtual environment
├── requirements.txt            # Python dependencies
├── .env                        # Environment variables
├── .env.example                # Environment template
├── manage.py                   # Django management
├── gunicorn_config.py          # Gunicorn config
└── README.md                   # This file
```

1.15.4 Database Schema Overview

Main Tables: - **Colleague**: Staff information and employment tracking - **Output**: Research outputs and publications - **CriticalFriend**: External reviewers - **CriticalFriendAssignment**: External review assignments - **InternalPanelMember**: Internal reviewers (v2.0) - **InternalPanelAssignment**: Internal review assignments (v2.0) - **Request**: REF-related requests - **Task**: General task management (v2.0) - **User**: Django authentication

1.15.5 Common Commands Reference

```
# Development
python manage.py runserver
python manage.py shell
python manage.py dbshell

# Database
python manage.py makemigrations
python manage.py migrate
python manage.py sqlmigrate core 0001

# User management
python manage.py createsuperuser
python manage.py changepassword username

# Static files
python manage.py collectstatic
python manage.py findstatic style.css

# Testing
python manage.py test
python manage.py test core.tests.TestOutputModel

# Production
gunicorn ref_manager.wsgi:application
sudo systemctl status ref-manager
sudo systemctl restart ref-manager

# Maintenance
python manage.py check
```

```
python manage.py check --deploy
python manage.py showmigrations
python manage.py migrate --run-syncdb
```

Version: 2.0.0
Last Updated: November 3, 2025
Prepared by: George Tsoulas
Institution: Department of Language and Linguistic Science, University of York

For the latest documentation and updates, please refer to the documentation suite or contact your system administrator.