

Contents

1	□ START HERE - REF Manager v2.0	1
1.1	What is REF Manager?	2
1.2	□ What's New in Version 2.0?	2
1.3	□ Where to Start?	2
1.3.1	For New Users	2
1.3.2	For Administrators	2
1.3.3	Learning About Changes	3
1.4	□ Documentation Suite	3
1.5	⚡ Quick Start (Really Quick!)	3
1.5.1	1. Install (10 minutes)	3
1.5.2	2. First Login	4
1.5.3	3. Try v2.0 Features	4
1.6	□ Common Tasks → Quick Guide	4
1.7	□ Finding Information	5
1.7.1	By Question Type	5
1.7.2	By User Role	5
1.8	□ Pro Tips	5
1.9	□ 3-Step Recommendation	5
1.9.1	Step 1: Quick Start (20 minutes)	5
1.9.2	Step 2: Learn Features (1 hour)	6
1.9.3	Step 3: Master the System (ongoing)	6
1.10	□ Need Help?	6
1.11	□ Quick Checklist	6
1.12	□ Learning Path	6
1.13	□ Documentation Map	7
1.14	□ Version Information	7
1.15	□ Your Next Step	7

1 □ START HERE - REF Manager v2.0

Quick Overview and Navigation Guide

1.1 What is REF Manager?

REF Manager v2.0 is a comprehensive web application for managing Research Excellence Framework (REF) submissions at UK academic institutions. It helps you track research outputs, manage staff, coordinate reviews, and generate professional reports.

1.2 □ What's New in Version 2.0?

- **Employment Status Tracking** - Current vs Former staff
 - **Enhanced Categories** - 9 colleague types including non-independent researchers
 - **Internal Panel System** - Complete internal review management
 - **Task Management** - Track all REF activities with priorities and deadlines
 - **CSV Import** - Bulk import for outputs and colleagues
 - **Excel Export** - Export assignments with clickable PDF links
 - **Request Enhancements** - Mark as completed and delete with confirmation
 - **Dashboard Updates** - New widgets for Internal Panel and Tasks
-

1.3 □ Where to Start?

1.3.1 For New Users

- 1. First Time Setup (10 minutes)** → Read: **QUICK-START-GUIDE.md** - Installation in 6 steps - Create your first colleague and output - Try new v2.0 features
- 2. Learn the System** → Read: **USER-GUIDE.md** (sections as needed) - Dashboard overview - Feature-by-feature guide - Best practices - FAQ
- 3. If You Get Stuck** → Read: **TROUBLESHOOTING.md** - Common issues and fixes - Quick problem finder - Error message index

1.3.2 For Administrators

- 1. Installation and Deployment** → Read: **REF-MANAGER-README.md** - Complete system documentation - Installation guide - Production deployment - Maintenance procedures
- 2. Technical Details** → Read: **TECHNICAL-DOCUMENTATION.md** - Architecture overview - Database models - API reference - Development guide
- 3. Building Documentation PDFs** → Read: **README-TOOLS.md** - Documentation build process - Using build_docs.sh script - PDF generation

1.3.3 Learning About Changes

What Changed in v2.0? → Read: **CHANGELOG.md** - Complete version history - All v2.0 additions - Upgrade notes - Migration guide

1.4 □ Documentation Suite

Document	Purpose	Length	Read When
THIS FILE	Overview & navigation	5 min	Start here!
QUICK-START-GUIDE.md	Fast installation	15 pages	First install
USER-GUIDE.md	Complete feature guide	50 pages	Daily reference
REF-MANAGER-README.md	System documentation	60 pages	Installation & admin
TECHNICAL-DOCUMENTATION.md	Developer reference	35 pages	Development
TROUBLESHOOTING.md	Problem solving	15 pages	When stuck
CHANGELOG.md	Version history	10 pages	Upgrade planning
DOCUMENTATION-INDEX.md	Doc navigation	5 pages	Find info
README-TOOLS.md	Build instructions	5 pages	Generate PDFs

Total: ~155 pages of comprehensive documentation

1.5 ↘ Quick Start (Really Quick!)

1.5.1 1. Install (10 minutes)

```
# Create project directory
mkdir -p ~/ref-project-app
cd ~/ref-project-app

# Get the code (git clone or extract ZIP)
git clone [repository] ref-manager
cd ref-manager

# Set up Python environment
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

```

# Configure
cp .env.example .env
# Edit .env and add SECRET_KEY

# Initialize database
python manage.py migrate
python manage.py createsuperuser

# Start server
python manage.py runserver

```

1.5.2 2. First Login

1. Go to <http://localhost:8000>
2. Log in with superuser credentials
3. Explore the dashboard

1.5.3 3. Try v2.0 Features

Add a colleague with new categories: - Colleagues → Add Colleague - Set Employment Status: Current - Choose Category: Non-Independent Researcher (for post-docs!)

Create a task: - Tasks → Create Task - Set priority and due date - Track your REF activities

Import outputs: - Outputs → Import - Download template - Upload CSV file

Export with links: - Export → Assignments - Choose filters - Export to Excel with clickable PDF links

1.6 ☐ Common Tasks → Quick Guide

Task	Where to Look
Install system	QUICK-START-GUIDE.md
Add colleague	USER-GUIDE.md → Colleague Management
Add output	USER-GUIDE.md → Output Management
Import data	USER-GUIDE.md → Data Import/Export
Set up Internal Panel	USER-GUIDE.md → Internal Panel System
Create tasks	USER-GUIDE.md → Task Management
Export assignments	USER-GUIDE.md → Data Import/Export
Generate reports	USER-GUIDE.md → Report Generation
Fix errors	TROUBLESHOOTING.md
Deploy to production	REF-MANAGER-README.md → Production Deployment
Understand changes	CHANGELOG.md
Build PDFs	README-TOOLS.md

1.7 □ Finding Information

1.7.1 By Question Type

“How do I...?” → USER-GUIDE.md (feature-specific sections)

“What is...?” → REF-MANAGER-README.md (Features section)

“Why isn’t it working?” → TROUBLESHOOTING.md

“What changed?” → CHANGELOG.md

“How does it work technically?” → TECHNICAL-DOCUMENTATION.md

1.7.2 By User Role

REF Coordinator: - QUICK-START-GUIDE.md (setup) - USER-GUIDE.md (daily use) - DOCUMENTATION-INDEX.md (navigation)

Department Administrator: - REF-MANAGER-README.md (full system) - USER-GUIDE.md (features) - TROUBLESHOOTING.md (support)

System Administrator: - REF-MANAGER-README.md (deployment) - TECHNICAL-DOCUMENTATION.md (architecture) - TROUBLESHOOTING.md (maintenance)

Developer: - TECHNICAL-DOCUMENTATION.md (development) - CHANGELOG.md (version history) - README-TOOLS.md (build process)

1.8 □ Pro Tips

1. **Read the Quick Start first** - Even if you’re experienced, it’s only 10 minutes
 2. **Use the User Guide as reference** - Don’t try to read it all at once
 3. **Bookmark Troubleshooting** - You’ll need it eventually
 4. **Check the FAQ** - USER-GUIDE.md has extensive FAQ section
 5. **Build PDFs** - Easier to share and read offline (see README-TOOLS.md)
 6. **Index is your friend** - DOCUMENTATION-INDEX.md helps navigate
-

1.9 □ 3-Step Recommendation

1.9.1 Step 1: Quick Start (20 minutes)

1. Read QUICK-START-GUIDE.md (10 min)
2. Install and run (10 min)
3. Log in and explore dashboard

1.9.2 Step 2: Learn Features (1 hour)

1. USER-GUIDE.md → Dashboard Overview (10 min)
2. USER-GUIDE.md → Colleague Management (15 min)
3. USER-GUIDE.md → Output Management (15 min)
4. USER-GUIDE.md → v2.0 New Features (20 min)

1.9.3 Step 3: Master the System (ongoing)

- Use USER-GUIDE.md as daily reference
 - Check TROUBLESHOOTING.md when stuck
 - Read CHANGELOG.md for updates
 - Explore TECHNICAL-DOCUMENTATION.md for deep understanding
-

1.10 ☰ Need Help?

Can't find what you need? 1. Check DOCUMENTATION-INDEX.md for navigation
2. Use Ctrl+F to search in documents 3. Check USER-GUIDE.md → FAQ section 4. Look in TROUBLESHOOTING.md 5. Contact system administrator

Found an error in documentation? - Note document name and section - Email details to maintainer - Help improve for everyone!

1.11 ☰ Quick Checklist

Before you start: - [] Have Python 3.10+ installed - [] Have 500MB free disk space - [] Have internet connection - [] Know your role (coordinator/admin/user)

After installation: - [] Can log in successfully - [] Dashboard displays correctly - [] Added first colleague - [] Added first output - [] Tried v2.0 features

For daily use: - [] Bookmarked USER-GUIDE.md - [] Know how to find information - [] Comfortable with basic features - [] Know where to get help

1.12 ☰ Learning Path

Day 1: Installation + Quick Start - Install system (QUICK-START-GUIDE.md) - Explore dashboard - Add test data

Day 2: Core Features - Colleague management (USER-GUIDE.md) - Output management (USER-GUIDE.md) - Dashboard widgets

Day 3: v2.0 Features - Employment status tracking - Colleague categories - Internal Panel system - Task management

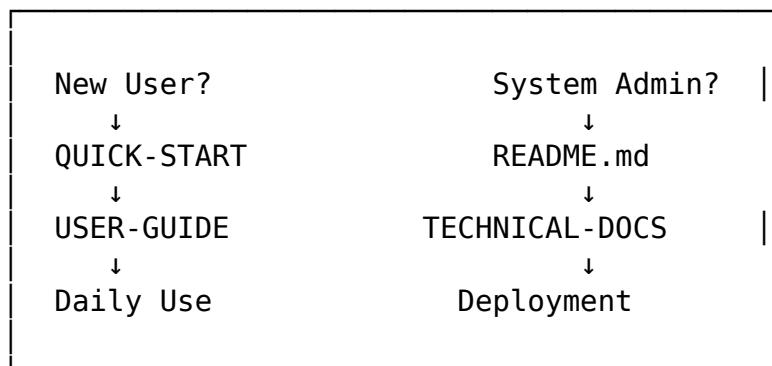
Day 4: Advanced Features - CSV import - Excel export - Request management - Report generation

Day 5: Administration - Production deployment (if needed) - Backup procedures - Troubleshooting - Performance optimization

1.13 □ Documentation Map

START HERE (this file)

↓



↓

Need Help?

↓

TRROUBLESHOOTING

↓

Still Stuck?

↓

Contact Administrator

1.14 □ Version Information

Current Version: 2.0.0

Release Date: November 3, 2025

Documentation Updated: November 3, 2025

What's in v2.0: - Employment status tracking - 9 colleague categories - Internal Panel system - Task management - CSV import - Excel export with links - Enhanced requests - Dashboard improvements

Upgrading from v1.0? → See CHANGELOG.md for migration guide

1.15 □ Your Next Step

Choose your path:

- **I'm brand new** Read QUICK-START-GUIDE.md next
 - **I'm upgrading** Read CHANGELOG.md next
 - **I want technical details** Read TECHNICAL-DOCUMENTATION.md next
 - **I'm having problems** Read TROUBLESHOOTING.md next
 - **I want to explore** Read USER-GUIDE.md next
-

Welcome to REF Manager v2.0!

We've built comprehensive documentation to help you succeed. Start with the Quick Start Guide, use the User Guide as your daily reference, and don't hesitate to check the Troubleshooting Guide when needed.

Questions? Check DOCUMENTATION-INDEX.md for complete navigation.

Ready? Let's get started with QUICK-START-GUIDE.md!

Prepared by: George Tsoulas

Institution: University of York

For: REF Manager v2.0 Users

Good luck with your REF submission! ☺

Contents

1 REF Manager v2.0 - Complete Documentation	2
1.1 □ Table of Contents	3
1.2 □ Overview	3
1.2.1 Key Benefits	3
1.3 □ What's New in v2.0	3
1.3.1 Major Features Added	3
1.4 □ System Requirements	5
1.4.1 Minimum Requirements	5
1.4.2 Required Software	6
1.4.3 Optional Software	6
1.5 □ Installation	6
1.5.1 Quick Start (Development)	6
1.5.2 Detailed Installation Steps	7
1.6 ☀ Configuration	9
1.6.1 Database Configuration	9
1.6.2 Email Configuration (Optional)	10
1.6.3 Static Files Configuration	11
1.6.4 Media Files Configuration	11
1.7 □ Features	11
1.7.1 1. Dashboard	11
1.7.2 2. Colleague Management	12
1.7.3 3. Research Output Management	12
1.7.4 4. Critical Friends System	13
1.7.5 5. Internal Panel System □ NEW in v2.0	13
1.7.6 6. Request Management	13
1.7.7 7. Task Management □ NEW in v2.0	13
1.7.8 8. Report Generation	14
1.7.9 9. Data Import/Export □ ENHANCED in v2.0	14
1.7.10 10. User Management	14
1.8 □ Running the Application	15
1.8.1 Development Server	15
1.8.2 Background Service (systemd)	15
1.8.3 Using Screen (Alternative)	16

1.9 □ Production Deployment	16
1.9.1 Using Gunicorn + Nginx	16
1.9.2 Using Docker (Optional)	19
1.10 □ Maintenance & Backup	20
1.10.1 Regular Maintenance Tasks	20
1.10.2 Backup Procedures	20
1.10.3 Restore Procedures	22
1.10.4 Log Management	23
1.10.5 Update Procedures	23
1.11 □ Troubleshooting	24
1.11.1 Common Issues	24
1.11.2 Debug Mode	27
1.11.3 Getting Help	27
1.12 □ Support	28
1.12.1 Documentation Resources	28
1.12.2 Getting Help	28
1.12.3 Reporting Issues	28
1.12.4 Feature Requests	29
1.13 □ License	29
1.14 □ Acknowledgments	29
1.15 □ Appendix	29
1.15.1 System Architecture	29
1.15.2 Technology Stack	30
1.15.3 File Structure	30
1.15.4 Database Schema Overview	31
1.15.5 Common Commands Reference	31

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. [Overview](#)
 2. [What's New in v2.0](#)
 3. [System Requirements](#)
 4. [Installation](#)
 5. [Configuration](#)
 6. [Features](#)
 7. [Running the Application](#)
 8. [Production Deployment](#)
 9. [Maintenance & Backup](#)
 10. [Troubleshooting](#)
 11. [Support](#)
-

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/HEAD/install.sh)"
```

```
# 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" "$BACKUP_DIR/db.sqlite3.$TIMESTAMP"
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_$TIMESTAMP.sql.gz"
find "$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 > "$BACKUP_DIR/database_$TIMESTAMP.gz"
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_manager_db

```

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 trace-back) - 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.

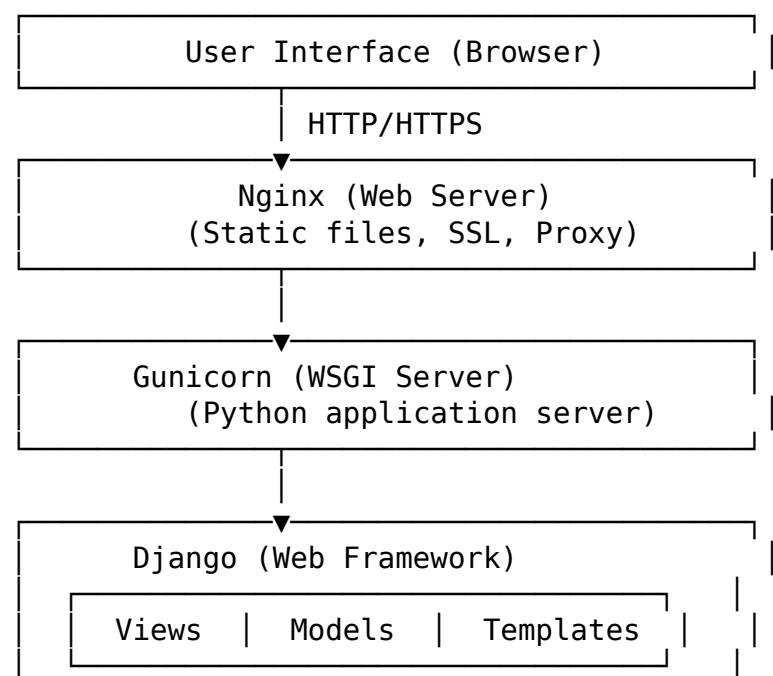
1.14 ☐ Acknowledgments

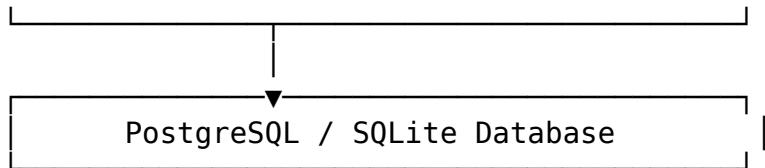
Developed for the Department of Language and Linguistic Science, University of York, to support REF 2029 submission management.

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

```

```

└── 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 - **Critical-FriendAssignment**: 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.

REF Manager v2.0

Quick Start Guide

Get up and running in 10 minutes!

George Tsoulas

Department of Language and Linguistic Science

University of York

November 3, 2025

Version 2.0.0

Abstract

This Quick Start Guide will help you install and configure REF Manager v2.0 in approximately 10 minutes. Whether you're setting up for the first time or upgrading from version 1.0, this guide provides step-by-step instructions to get your system running quickly.

Contents

1 Prerequisites	3
1.1 Quick Check	3
2 Installation (10 Minutes)	3
2.1 Step 1: Download and Setup (2 minutes)	3
2.2 Step 2: Create Virtual Environment (1 minute)	3
2.3 Step 3: Install Dependencies (3 minutes)	3
2.4 Step 4: Configure Environment (1 minute)	4
2.5 Step 5: Initialize Database (2 minutes)	4
2.6 Step 6: Start the Server (1 minute)	4
3 First Steps After Installation	5
3.1 Log In (1 minute)	5
3.2 Explore the Dashboard (2 minutes)	5
3.3 Add Your First Colleague (3 minutes)	5
4 Try v2.0 Features (5 minutes)	6
4.1 Create a Task	6
4.2 Set Up Internal Panel	6
4.3 Try CSV Import	6
5 Daily Workflow	7
5.1 Typical REF Manager Day	7
5.1.1 Morning (5 minutes)	7
5.1.2 During the Day	7
5.1.3 Weekly (15 minutes)	7
5.1.4 Monthly (30 minutes)	7
6 Running as Background Service	8
6.1 Option 1: Using Screen (Simplest)	8
6.2 Option 2: Using systemd (Production)	8
6.3 Manage the Service	8
7 Quick Commands Reference	10
7.1 Daily Commands	10
7.2 Admin Tasks	10
7.3 Database Tasks	10
8 Quick Fixes for Common Issues	11
8.1 Can't Log In	11
8.2 Page Not Loading / CSS Not Working	11
8.3 "No Such Table" Errors	11
8.4 Python 3.13 Compatibility	11
9 What's New in v2.0	12
9.1 Major Additions	12
10 Next Steps	12
11 Pro Tips	13

12 Checklist	13
12.1 Installation	13
12.2 Initial Setup	13
12.3 Testing v2.0 Features	14
12.4 Production (Optional)	14

1 Prerequisites

Before you start, ensure you have:

- Python 3.10 or higher installed
- Terminal/Command Prompt access
- Internet connection
- 500MB free disk space

1.1 Quick Check

Verify your Python version:

```
1 python3 --version
```

You should see version 3.10 or higher.

2 Installation (10 Minutes)

2.1 Step 1: Download and Setup (2 minutes)

```
1 # Create project directory
2 mkdir -p ~/ref-project-app
3 cd ~/ref-project-app
4
5 # If using Git
6 git clone https://github.com/yourusername/ref-manager.git
7 cd ref-manager
```

Or extract ZIP file if downloaded manually.

2.2 Step 2: Create Virtual Environment (1 minute)

```
1 # Create virtual environment
2 python3 -m venv venv
3
4 # Activate it
5 # Linux/macOS:
6 source venv/bin/activate
7
8 # Windows:
9 venv\Scripts\activate
10
11 # You should see (venv) in your prompt
```

2.3 Step 3: Install Dependencies (3 minutes)

```
1 # Upgrade pip
2 pip install --upgrade pip
3
4 # Install all required packages
5 pip install -r requirements.txt
6
7 # If on Python 3.13 and getting errors:
8 pip install --break-system-packages -r requirements.txt
```

What gets installed:

- Django 4.2+ (web framework)
- django-crispy-forms (beautiful forms)
- openpyxl (Excel export) **NEW in v2.0**
- gunicorn (production server)
- psycopg2-binary (PostgreSQL support)

2.4 Step 4: Configure Environment (1 minute)

```
1 # Copy environment template
2 cp .env.example .env
3
4 # Generate a secret key
5 python -c "from django.core.management.utils import get_random_secret_key;
6     print(get_random_secret_key())"
7
8 # Edit .env file
9 nano .env # or use any text editor
```

Minimum .env configuration:

```
1 SECRET_KEY=paste-the-secret-key-here
2 DEBUG=True
3 ALLOWED_HOSTS=localhost,127.0.0.1
```

2.5 Step 5: Initialize Database (2 minutes)

```
1 # Create database tables
2 python manage.py migrate
3
4 # Create admin user
5 python manage.py createsuperuser
6 # Enter username, email, and password when prompted
```

2.6 Step 6: Start the Server (1 minute)

```
1 # Run development server
2 python manage.py runserver
3
4 # Server should start at http://127.0.0.1:8000
```

Congratulations!

Your REF Manager is now running! Open your browser and go to:
<http://localhost:8000>

3 First Steps After Installation

3.1 Log In (1 minute)

1. Open <http://localhost:8000>
2. Click “Login” in the top right
3. Enter your superuser credentials
4. You’ll see the dashboard

3.2 Explore the Dashboard (2 minutes)

The dashboard shows:

- **Submission Statistics:** Output counts and quality profile
- **Staff Summary:** Current and former colleagues
- **Review Progress:** Critical Friends and Internal Panel
- **Task Overview:** NEW - Urgent and overdue tasks
- **Recent Activity:** Latest changes

3.3 Add Your First Colleague (3 minutes)

1. Click “Colleagues” in navigation
2. Click “Add Colleague” button
3. Fill in:
 - First name, Last name
 - Email
 - Unit of Assessment
 - FTE (Full-Time Equivalent, e.g., 1.0)
 - **Employment Status:** Current (NEW in v2.0)
 - **Category:** Choose appropriate type (NEW in v2.0)
4. Click “Save”

NEW Categories in v2.0

- Independent Researcher
- Non-Independent Researcher (for post-docs!)
- Post-Doctoral Researcher
- Research Assistant
- Academic Staff
- Support Staff
- Co-author (External)

4 Try v2.0 Features (5 minutes)

4.1 Create a Task

1. Click “**Tasks**” in navigation
2. Click “**Create Task**”
3. Fill in:
 - Title
 - Category (e.g., “Submission”, “Administrative”)
 - Priority (Low, Medium, High, Urgent)
 - Due Date
 - Description
4. Click “**Save**”

4.2 Set Up Internal Panel

1. Click “**Internal Panel**” in navigation
2. Click “**Add Panel Member**”
3. Select a colleague
4. Choose role (Chair, Member, Specialist)
5. Click “**Save**”

4.3 Try CSV Import

1. Click “**Outputs**” → “**Import**”
2. Download the CSV template
3. Fill in output data
4. Upload CSV file
5. Review and confirm import

5 Daily Workflow

5.1 Typical REF Manager Day

5.1.1 Morning (5 minutes)

- Check dashboard for urgent tasks
- Review overdue reviews
- Check recent activity

5.1.2 During the Day

- Add new outputs as they're published
- Update colleague information
- Assign papers for review
- Track review progress
- Complete tasks

5.1.3 Weekly (15 minutes)

- Review quality profile
- Check submission progress
- Update request status
- Generate reports

5.1.4 Monthly (30 minutes)

- Full review progress audit
- Update staff status (current/former)
- Export data for meetings
- Generate comprehensive reports

6 Running as Background Service

6.1 Option 1: Using Screen (Simplest)

```
1 # Install screen
2 sudo apt-get install screen
3
4 # Start screen session
5 screen -S ref-manager
6
7 # Run server
8 cd ~/ref-project-app/ref-manager
9 source venv/bin/activate
10 python manage.py runserver 0.0.0.0:8000
11
12 # Detach: Press Ctrl+A, then D
13
14 # Reattach later
15 screen -r ref-manager
```

6.2 Option 2: Using systemd (Production)

Create service file:

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

Add this content (replace YOUR_USERNAME):

```
1 [Unit]
2 Description=REF Manager Django Application
3 After=network.target
4
5 [Service]
6 Type=simple
7 User=YOUR_USERNAME
8 WorkingDirectory=/home/YOUR_USERNAME/ref-project-app/ref-manager
9 Environment="PATH=/home/YOUR_USERNAME/ref-project-app/ref-manager/venv/bin"
10 ExecStart=/home/YOUR_USERNAME/ref-project-app/ref-manager/venv/bin/gunicorn
11   \
12   --workers 3 \
13   --bind 0.0.0.0:8000 \
14   ref_manager.wsgi:application
15 Restart=always
16
17 [Install]
18 WantedBy=multi-user.target
```

Enable and start:

```
1 sudo systemctl enable ref-manager
2 sudo systemctl start ref-manager
3 sudo systemctl status ref-manager
```

6.3 Manage the Service

```
1 # Start
2 sudo systemctl start ref-manager
3
4 # Stop
```

```
5 sudo systemctl stop ref-manager
6
7 # Restart (after code changes)
8 sudo systemctl restart ref-manager
9
10 # Check status
11 sudo systemctl status ref-manager
12
13 # View logs
14 sudo journalctl -u ref-manager -f
```

7 Quick Commands Reference

7.1 Daily Commands

```
1 # Start server (development)
2 python manage.py runserver
3
4 # Start server (accessible from other computers)
5 python manage.py runserver 0.0.0.0:8000
6
7 # Access Django shell
8 python manage.py shell
9
10 # Create backup
11 cp db.sqlite3 backups/db.sqlite3.$(date +%Y%m%d)
```

7.2 Admin Tasks

```
1 # Create new user
2 python manage.py createsuperuser
3
4 # Change password
5 python manage.py changepassword username
6
7 # Collect static files (production)
8 python manage.py collectstatic --noinput
```

7.3 Database Tasks

```
1 # Create migrations (after model changes)
2 python manage.py makemigrations
3
4 # Apply migrations
5 python manage.py migrate
6
7 # Show migration status
8 python manage.py showmigrations
9
10 # Access database directly
11 python manage.py dbshell
```

8 Quick Fixes for Common Issues

8.1 Can't Log In

```
1 # Reset password
2 python manage.py changepassword yourusername
```

8.2 Page Not Loading / CSS Not Working

```
1 # Collect static files
2 python manage.py collectstatic --noinput
3
4 # Restart server
5 # Press Ctrl+C, then run again
6 python manage.py runserver
```

8.3 “No Such Table” Errors

```
1 # Run migrations
2 python manage.py migrate
```

8.4 Python 3.13 Compatibility

If you see: `TypeError: unsupported operand type(s) for *: 'decimal.Decimal' and 'float'`
This is already fixed in v2.0 code. Update to latest version.

9 What's New in v2.0

9.1 Major Additions

- ✓ **Employment Status Tracking** - Current vs Former staff
- ✓ **Enhanced Categories** - 9 colleague types
- ✓ **Internal Panel System** - Internal reviewer management
- ✓ **Task Management** - Track all activities
- ✓ **CSV Import** - Bulk operations
- ✓ **Excel Export** - Export with clickable links
- ✓ **Enhanced Requests** - Mark complete, delete
- ✓ **Dashboard Updates** - New widgets

10 Next Steps

After Quick Start:

1. **Read the User Guide (USER-GUIDE.md)**
 - Detailed feature documentation
 - Best practices
 - Advanced workflows
2. **Check Technical Documentation (TECHNICAL-DOCUMENTATION.md)**
 - System architecture
 - Development guide
 - API reference
3. **Review Troubleshooting Guide (TROUBLESHOOTING.md)**
 - Detailed problem solving
 - Performance optimization
 - Security considerations
4. **Read the Changelog (CHANGELOG.md)**
 - Version history
 - All changes and improvements
 - Migration notes

11 Pro Tips

1. **Regular Backups:** Back up database daily

```
1 cp db.sqlite3 backups/db.sqlite3.$(date +%Y%m%d)  
2
```

2. **Use Categories:** Properly categorize colleagues for better reporting
3. **Track Everything:** Use tasks to track all REF activities
4. **Export Often:** Export data regularly for meetings and reports
5. **Former Staff:** Don't delete former staff - mark as "Former" instead
6. **CSV Import:** Use CSV import for bulk data entry - much faster!
7. **Excel Export:** Export assignments with links to send to reviewers
8. **Dashboard:** Check dashboard daily for urgent items
9. **Internal vs External:** Use Internal Panel for university reviewers, Critical Friends for external
10. **Quality Ratings:** Update ratings as papers are evaluated

12 Checklist

Use this checklist to ensure proper setup:

12.1 Installation

- Python 3.10+ installed
- Virtual environment created
- Dependencies installed
- Environment variables configured
- Database initialized
- Superuser created
- Server runs without errors

12.2 Initial Setup

- First login successful
- Dashboard displays correctly
- Added at least one colleague
- Added at least one output
- Created a task (v2.0)
- Set up Internal Panel member (v2.0)

12.3 Testing v2.0 Features

- Tried colleague categories
- Marked someone as former staff
- Created an internal panel member
- Created and completed a task
- Tested CSV import
- Tested Excel export with links

12.4 Production (Optional)

- Background service configured
- Automatic startup enabled
- Backup script created
- Staff trained

Need More Help?

See the complete documentation suite or contact your system administrator.

Ready to become a REF Manager pro? Read the User Guide next!

Contents

1 REF Manager v2.0 - User Guide	3
1.1 □ Table of Contents	3
1.2 □ Introduction	4
1.2.1 What's New in v2.0	4
1.2.2 Who Should Use This Guide	4
1.3 □ Dashboard Overview	4
1.3.1 Dashboard Widgets	4
1.3.2 Dashboard Best Practices	6
1.4 □ Colleague Management	6
1.4.1 Adding a Colleague	6
1.4.2 Understanding Colleague Categories □ NEW	7
1.4.3 Employment Status Management □ NEW	8
1.4.4 Viewing Colleagues	8
1.4.5 Required Outputs Calculation	9
1.4.6 Best Practices	9
1.5 □ Research Output Management	9
1.5.1 Adding an Output	9
1.5.2 Quality Ratings	10
1.5.3 Bulk Operations	11
1.5.4 Filtering and Searching	12
1.5.5 Output Detail View	12
1.5.6 Best Practices	12
1.6 □ Critical Friends System	12
1.6.1 About Critical Friends	12
1.6.2 Adding a Critical Friend	13
1.6.3 Assigning Outputs to Critical Friends	13
1.6.4 Assignment Tracking	13
1.6.5 Export Assignments □ NEW in v2.0	14
1.6.6 Managing Reviews	14
1.6.7 Critical Friends Dashboard	15
1.6.8 Best Practices	15
1.7 □ Internal Panel System □ NEW in v2.0	15
1.7.1 About Internal Panel	15

1.7.2 Setting Up Internal Panel	16
1.7.3 Assigning Outputs for Internal Review	16
1.7.4 Assignment Status Tracking	17
1.7.5 Recording Internal Reviews	17
1.7.6 Internal Panel Dashboard Widget	18
1.7.7 Panel Meetings	18
1.7.8 Internal vs External Review Workflow	19
1.7.9 Best Practices	19
1.8 □ Task Management □ NEW in v2.0	19
1.8.1 About Tasks	19
1.8.2 Creating a Task	20
1.8.3 Task Categories Explained	20
1.8.4 Priority Levels	21
1.8.5 Managing Tasks	21
1.8.6 Task Dashboard Widget	22
1.8.7 Task List Views	22
1.8.8 Task Best Practices	22
1.8.9 Task Examples	23
1.9 □ Request Management	23
1.9.1 About Requests	23
1.9.2 Creating a Request	23
1.9.3 Request Status Flow	24
1.9.4 Managing Requests	24
1.9.5 Request List Features	25
1.9.6 Request Detail View	25
1.9.7 Request Dashboard	25
1.9.8 Best Practices	25
1.10 □ Data Import/Export □ ENHANCED in v2.0	26
1.10.1 CSV Import □ NEW	26
1.10.2 Excel Export □ NEW	27
1.10.3 Other Export Options	28
1.10.4 Import/Export Best Practices	28
1.11 □ Report Generation	29
1.11.1 Available Reports	29

1.11.2 Report Formats	29
1.11.3 Generating Reports	30
1.11.4 Report Customization	30
1.11.5 Report Best Practices	30
1.12 □ Best Practices	30
1.12.1 Data Quality	30
1.12.2 Workflow Efficiency	31
1.12.3 Quality Assurance	31
1.12.4 Team Coordination	31
1.12.5 Data Management	31
1.12.6 Security	31
1.13 □ Frequently Asked Questions	32
1.13.1 General Questions	32
1.13.2 Output Management	32
1.13.3 Import/Export □ NEW	32
1.13.4 Tasks □ NEW	33
1.13.5 Reviews	33
1.13.6 Technical	33
1.13.7 Troubleshooting	34
1.14 □ Getting Help	34
1.15 □ Additional Resources	35

1 REF Manager v2.0 - User Guide

Complete Feature Guide and Best Practices

Version: 2.0.0

Last Updated: November 3, 2025

For: All REF Manager Users

1.1 □ Table of Contents

1. Introduction
2. Dashboard Overview
3. Colleague Management
4. Research Output Management
5. Critical Friends System

6. Internal Panel System NEW
 7. Task Management NEW
 8. [Request Management](#)
 9. Data Import/Export ENHANCED
 10. [Report Generation](#)
 11. [Best Practices](#)
 12. [Frequently Asked Questions](#)
-

1.2 Introduction

Welcome to the REF Manager v2.0 User Guide. This comprehensive guide covers all features and provides best practices for effective REF submission management.

1.2.1 What's New in v2.0

REF Manager v2.0 introduces major enhancements:

- **Employment Status Tracking**: Current vs Former staff
- **Enhanced Categories**: 9 colleague types including non-independent researchers
- **Internal Panel System**: Complete internal review management
- **Task Management**: Track all REF activities
- **CSV Import**: Bulk data import
- **Excel Export**: Export assignments with clickable links
- **Enhanced Dashboard**: New widgets for tasks and internal panel

1.2.2 Who Should Use This Guide

- **REF Coordinators**: Managing overall submission
 - **Department Administrators**: Supporting REF process
 - **Academic Staff**: Contributing outputs and participating in reviews
 - **Panel Members**: Internal and external reviewers
-

1.3 Dashboard Overview

The dashboard is your central hub for monitoring REF submission progress.

1.3.1 Dashboard Widgets

- 1.3.1.1 1. Submission Statistics** **What it shows:** - Total number of outputs
- Quality profile breakdown (4, 3, 2, 1, Unclassified) - Average quality score - Distribution by Unit of Assessment

How to use it: - Quick assessment of submission quality - Identify areas needing improvement - Track progress toward quality targets - Monitor UoA balance

Example interpretation:

Total Outputs: 125
4*: 45 (36%) ← Excellent! Target: 30%+
3*: 50 (40%) ← Good solid base
2*: 20 (16%) ← Acceptable
1*: 5 (4%) ← Review these
Unclassified: 5 (4%) ← Need evaluation
Average: 3.04 ← Above target of 3.0

1.3.1.2 2. Staff Summary What it shows: - Total current colleagues - Former staff count - Category breakdown - Average outputs per colleague - Employment status distribution

How to use it: - Monitor staff contributions - Identify colleagues needing more outputs - Track employment changes - Plan submission strategy

1.3.1.3 3. Review Progress What it shows: - Critical Friends assignments - Internal Panel assignments □ NEW - Pending reviews - Completed reviews - Average review turnaround time

How to use it: - Monitor review deadlines - Identify bottlenecks - Follow up on overdue reviews - Track reviewer workload

1.3.1.4 4. Internal Panel Statistics □ NEW in v2.0 What it shows: - Total panel members - Active assignments - Completed internal reviews - Quality rating distribution from internal reviews - Pending assessments

How to use it: - Monitor internal evaluation progress - Balance panel member workload - Track internal quality consensus - Identify outputs needing attention

Example:

Panel Members: 8
Active Assignments: 45
Completed Reviews: 78
Pending: 12
Average Internal Rating: 3.2

1.3.1.5 5. Task Overview □ NEW in v2.0 What it shows: - Urgent tasks (due within 7 days) - Overdue tasks - Tasks by status (Pending, In Progress, Completed) - Tasks by priority - Upcoming deadlines

How to use it: - Prioritize daily work - Never miss deadlines - Track team progress - Identify blockers

Color coding: - □ Red: Overdue - immediate action required - □ Yellow: Due soon - urgent attention needed - □ Green: On track - monitor progress - ○ Gray: Completed - archive reference

1.3.1.6 6. Recent Activity **What it shows:** - Latest outputs added - Recent reviews completed - New requests - System updates - Recent status changes

How to use it: - Stay informed of changes - Quick access to recent items - Monitor team activity - Audit trail

1.3.2 Dashboard Best Practices

1. **Check Daily:** Review dashboard each morning
 2. **Act on Red Flags:** Address urgent items immediately
 3. **Monitor Trends:** Track quality and completion over time
 4. **Share:** Use dashboard in team meetings
 5. **Export:** Take screenshots for reports
-

1.4 □ Colleague Management

Manage all staff involved in REF submissions, including current and former employees.

1.4.1 Adding a Colleague

Step-by-step:

1. Click “**Colleagues**” in navigation
2. Click “**Add Colleague**” button
3. Fill in **Basic Information**:
 - First Name *
 - Last Name *
 - Email *
 - Title (e.g., Professor, Dr., etc.)
4. Set **Employment Details**: □ ENHANCED in v2.0
 - **Employment Status:** Current or Former *
 - **Employment End Date:** (if Former)
 - **FTE** (Full-Time Equivalent): 0.1 to 1.0 *
 - **Unit of Assessment:** Select primary UoA *
5. Choose **Colleague Category**: □ NEW in v2.0
 - Independent Researcher
 - Non-Independent Researcher (for post-docs)
 - Post-Doctoral Researcher
 - Research Assistant
 - Academic Staff
 - Support Staff
 - Current Employee
 - Former Employee
 - Co-author (External)
6. Add **Optional Information**:
 - Office location
 - Phone number

- Research interests
- Biography
- Profile photo

7. Click “**Save**”

1.4.2 Understanding Colleague Categories □ NEW

1.4.2.1 Independent Researcher

- **Who:** Established researchers with independent research programs
- **Characteristics:** Can be sole author, lead major projects, supervise PhDs
- **Example:** Professors, Senior Lecturers with established track record
- **REF significance:** Can submit as independent researchers

1.4.2.2 Non-Independent Researcher

- **Who:** Post-doctoral researchers and early career staff
- **Characteristics:** Working under supervision, not yet independent
- **Example:** Post-docs, Research Fellows in early career
- **REF significance:** Important to identify for submission rules

1.4.2.3 Post-Doctoral Researcher

- **Who:** Specific designation for post-doc positions
- **Characteristics:** Between PhD and permanent position
- **Example:** Fixed-term post-doctoral research staff
- **REF significance:** May have different submission rules

1.4.2.4 Research Assistant

- **Who:** Research support staff
- **Characteristics:** Assist with research but don't lead projects
- **Example:** Lab technicians, research coordinators
- **REF significance:** Contributions need to be tracked

1.4.2.5 Academic Staff

- **Who:** Teaching and research faculty
- **Characteristics:** Both teaching and research responsibilities
- **Example:** Lecturers, Senior Lecturers
- **REF significance:** Expected to contribute outputs

1.4.2.6 Support Staff

- **Who:** Administrative and technical support
- **Characteristics:** Support research but don't produce outputs
- **Example:** Lab managers, research administrators
- **REF significance:** Usually not submitting outputs

1.4.2.7 Current Employee

- **Who:** Generic current staff category
- **Characteristics:** Currently employed
- **Example:** When specific category unclear
- **REF significance:** Active contributors

1.4.2.8 Former Employee

- **Who:** Past employees whose data we retain
- **Characteristics:** No longer employed
- **Example:** Staff who left during REF period
- **REF significance:** Their outputs still count

1.4.2.9 Co-author (External)

- **Who:** External collaborators
- **Characteristics:** Not department staff
- **Example:** Co-authors from other institutions
- **REF significance:** Track collaborations

1.4.3 Employment Status Management □ NEW

1.4.3.1 Current Staff Characteristics: - Actively employed - Appear in drop-down menus - Can be assigned outputs and reviews - Contribute to active statistics

Best practices: - Keep information up-to-date - Track FTE changes - Update categories as roles change - Monitor output contributions

1.4.3.2 Former Staff When to use: - Staff who left during REF period - Staff who retired - Staff who moved to other institutions

Why keep them: - Their outputs still count for REF - Historical record maintenance - Audit trail - Acknowledgment of contributions

How to mark as Former: 1. Edit colleague 2. Change Employment Status to "Former" 3. Set Employment End Date 4. Save

What happens: - Data preserved - Removed from active dropdowns - Still linked to outputs - Appears in former staff reports

1.4.4 Viewing Colleagues

1.4.4.1 List View Features: - Filter by employment status (Current/Former) - Filter by category - Sort by name, UoA, output count - Search by name or email - Quick view of key stats

Category Statistics □ NEW: Shows breakdown like:

Independent: 15
Non-Independent: 5
Post-Doctoral: 3
Academic Staff: 25
External Collaborators: 12

Employment Status Badges □ NEW: - □ Current: Green badge - ○ Former: Gray badge

1.4.4.2 Detail View What you see: - Complete profile information - Employment history - Research outputs list - Quality profile - Review assignments - Contact information - Required vs actual outputs

Actions available: - Edit information - Mark as former - Assign outputs - View reports - Export data

1.4.5 Required Outputs Calculation

Formula:

$$\text{Required Outputs} = \text{FTE} \times 2.5$$

Examples: - FTE 1.0: 2.5 outputs required - FTE 0.8: 2.0 outputs required - FTE 0.5: 1.25 outputs required

Dashboard shows: - Actual outputs submitted - Required outputs - Surplus or deficit - Percentage completion

1.4.6 Best Practices

1. **Accurate FTE:** Essential for correct calculations
 2. **Update Promptly:** Change status when staff leave
 3. **Choose Right Category:** Important for reporting
 4. **Don't Delete:** Mark as Former instead
 5. **External Co-authors:** Tag them properly
 6. **Regular Audits:** Check accuracy monthly
-

1.5 □ Research Output Management

Track all research outputs eligible for REF submission.

1.5.1 Adding an Output

Step-by-step:

1. Click “**Outputs**” in navigation
2. Click “**Add Output**” button
3. Fill in **Publication Details**:

- **Title:** Complete publication title *
 - **All Authors:** Full author list in correct order *
 - **Publication Venue:** Journal, conference, or publisher *
 - **Publication Type:** * Choose from:
 - Journal Article
 - Conference Paper
 - Book
 - Book Chapter
 - Monograph
 - Report
 - Other
4. Set **Publication Information:**
 - **Publication Date:** * (YYYY-MM-DD format)
 - **Volume, Issue, Pages:** If applicable
 - **DOI:** Digital Object Identifier
 - **URL:** Online location
 5. Upload **PDF File:**
 - Click “Choose File”
 - Select PDF of publication
 - Maximum 20MB
 - Stored securely
 6. Set **REF Details:** *
 - **Lead Author:** Select from colleagues dropdown
 - **Unit of Assessment:** Primary UoA
 - **Quality Rating:** 4, 3, 2, 1, Unclassified
 - **REF Eligible:** Yes/No checkbox
 - **Status:** Draft, Under Review, Accepted, Published
 7. Add **Additional Information:**
 - **Keywords:** Comma-separated
 - **Abstract:** Brief summary
 - **Notes:** Internal notes for team
 8. Click “Save”

1.5.2 Quality Ratings

1.5.2.1 Understanding the Rating System 4* (World-Leading) - Quality: World-leading in originality, significance, rigour - Impact: International importance - Examples: Published in top journals, highly cited - Target: 30% or more

3* (Internationally Excellent) - Quality: Internationally excellent - Impact: Makes important contribution - Examples: Good international journals - Target: 50% or more

2* (Recognized Internationally) - Quality: Recognized internationally - Impact: Some contribution - Examples: Decent journals, conference papers - Target: Acceptable supporting base

1* (Recognized Nationally) - Quality: Recognized nationally - Impact: Limited contribution - Examples: Local publications - Target: Minimize

Unclassified - Not yet evaluated - Awaiting review - Insufficient information - Target: Evaluate all outputs

1.5.2.2 Rating Best Practices

1. **Multiple Reviews:** Get both internal and external opinions
2. **Evidence-Based:** Use citations, journal rankings
3. **Consistency:** Apply criteria uniformly
4. **Documentation:** Record reasoning
5. **Regular Review:** Ratings can change over time

1.5.3 Bulk Operations

1.5.3.1 CSV Import □ NEW in v2.0 Perfect for: - Importing many outputs at once - Migrating from other systems - Batch updates

Process:

1. Download Template:

- Go to Outputs → Import
- Click “Download CSV Template”
- Open in Excel or text editor

2. Fill in Data: Required columns:

- title
- all_authors
- publication_venue
- publication_type
- publication_date
- quality_rating
- author_email (to link to colleague)
- uoa
- ref_eligible (true/false)

Optional columns:

- doi
- url
- volume, issue, pages
- keywords
- abstract
- notes

3. Prepare CSV:

- Save as UTF-8 CSV
- Check for special characters
- Verify email addresses match colleagues
- Ensure dates in YYYY-MM-DD format

4. Upload:

- Click “Choose File”
- Select your CSV
- Click “Upload and Validate”

5. Review:

- System checks for errors
- Shows validation results
- Detects duplicates
- Auto-links to colleagues

6. Confirm:

- Review summary

- Check matched colleagues
- Click “Confirm Import”
- Outputs created

CSV Import Tips: - Start with template - don't create from scratch - Test with 2-3 rows first - External co-authors auto-categorized - Duplicates detected by title and DOI - Partial matches require manual review

1.5.4 Filtering and Searching

Filter Options: - Quality rating (4, 3, 2, 1, Unclassified) - Author (by colleague) - Unit of Assessment - Publication type - REF eligibility - Status - Date range

Search: - By title keywords - By author name - By venue - By DOI

Sorting: - By quality (highest first) - By date (newest first) - By title (alphabetical) - By author

1.5.5 Output Detail View

Information Shown: - Full publication details - PDF viewer (if uploaded) - Quality rating with visual indicator - Link to DOI (if available) - All authors with colleague links - Review history - Assignment status

Actions Available: - Edit output - Update quality rating - Upload/replace PDF - Assign to reviewers - Generate report - Export data - Delete (with confirmation)

1.5.6 Best Practices

1. **Complete Information:** Fill all fields possible
 2. **Upload PDFs:** Makes review easier
 3. **Accurate Authors:** Critical for credit assignment
 4. **DOI Links:** Verify accessibility
 5. **Regular Updates:** Add outputs as published
 6. **Quality Evidence:** Document rating rationale
 7. **CSV for Bulk:** Use import for many outputs
-

1.6 ☐ Critical Friends System

Manage external peer reviewers from other institutions.

1.6.1 About Critical Friends

What are Critical Friends? - External reviewers from other UK institutions - Provide independent quality assessment - Help calibrate ratings - Offer feedback on submission strategy

Why use them? - External validation - Cross-institution benchmarking - Fresh perspective - REF preparation

1.6.2 Adding a Critical Friend

Step-by-step:

1. Click “**Critical Friends**” in navigation
2. Click “**Add Critical Friend**” button
3. Fill in **Contact Information**: *
 - First Name
 - Last Name
 - Email
 - Institution
4. Add **Professional Details**:
 - Position/Title
 - Department
 - Expertise Areas (comma-separated)
 - Unit of Assessment
 - Biography
5. Set **Availability**:
 - Available/Unavailable toggle
 - Maximum assignments
 - Preferred review timeline
6. Click “**Save**”

1.6.3 Assigning Outputs to Critical Friends

Method 1: From Output Detail

1. View output
2. Click “**Assign to Critical Friend**”
3. Select critical friend
4. Set due date and priority
5. Add notes
6. Click “**Assign**”

Method 2: From Critical Friend Profile

1. View critical friend
2. Click “**Assign Output**”
3. Select output
4. Set details
5. Click “**Assign**”

1.6.4 Assignment Tracking

Assignment Status: - **Assigned**: Sent to reviewer, waiting - **In Progress**: Reviewer working on it - **Completed**: Review received - **Deferred**: Postponed - **Canceled**: No longer needed

Priority Levels: - **High:** Urgent, key outputs - **Medium:** Normal priority - **Low:** When convenient

1.6.5 Export Assignments □ NEW in v2.0

Perfect for: - Sending assignments to reviewers - Progress tracking - Meeting preparation - Email distribution

Process:

1. Go to “**Export**” → “**Assignments**”
2. Set filters:
 - Reviewer Type: Critical Friends
 - Specific Reviewer: (optional)
 - Author: (optional)
 - Status: (optional)
3. Click “**Export to Excel**”

Excel Output: - **Color-coded:** Yellow background for Critical Friends - **Clickable Links:** Direct links to PDF files - **Complete Info:** - Reviewer name and email - Output title and author - Quality rating - Review status - Due date - Priority - Notes

Use Cases:

Scenario 1: Send to Reviewer

- Export assignments for specific reviewer
- Email Excel file with links to papers
- Reviewer can click directly to PDFs

Scenario 2: Progress Meeting

- Export all assignments
- Show workload distribution
- Track completion rates
- Identify delays

Scenario 3: Reporting

- Export by status
- Show completed reviews
- Calculate turnaround times
- Quality analysis

1.6.6 Managing Reviews

Recording Feedback:

1. Go to assignment
2. Click “**Update Status**” → “**In Progress**”
3. When complete:
 - Set status to “**Completed**”
 - Record rating received
 - Add review notes
 - Set completion date

4. Save

Review Content to Track: - Quality rating suggested - Strengths identified - Weaknesses noted - Recommendations - Additional comments

1.6.7 Critical Friends Dashboard

What it Shows: - Total critical friends - Active vs. inactive - Current assignments - Completed reviews - Average turnaround time - Workload distribution

1.6.8 Best Practices

1. **Reciprocity:** Offer to review for them too
 2. **Clear Instructions:** Provide review guidelines
 3. **Reasonable Deadlines:** Allow 2-4 weeks
 4. **Avoid Overload:** Max 5-10 papers per reviewer
 5. **Follow Up:** Send reminders politely
 6. **Thank Reviewers:** Acknowledge their time
 7. **Use Excel Export:** Makes communication easy ☐ NEW
-

1.7 ☰ Internal Panel System ☐ NEW in v2.0

Manage your university's internal evaluation committee, separate from external Critical Friends.

1.7.1 About Internal Panel

What is the Internal Panel? - University's own evaluation committee - Internal quality assurance - First line of review before external - Different from Critical Friends (external reviewers)

Why Separate System? - Different workflow and process - Internal vs external perspective - Different reporting requirements - Separate dashboard tracking

Key Differences:

Aspect	Internal Panel	Critical Friends
Location	Same institution	External institutions
Purpose	Internal QA	External validation
Timing	First review	Second review
Access	More frequent	Limited engagement
Dashboard	Separate widget	Separate widget

1.7.2 Setting Up Internal Panel

1.7.2.1 Adding Panel Members Step-by-step:

1. Click “**Internal Panel**” in navigation
2. Click “**Add Panel Member**” button
3. **Select Colleague:** *
 - Choose from current staff
 - Must be in colleague list first
4. **Assign Role:** * Choose from:
 - **Chair:** Leads the panel, makes final calls
 - **Member:** Regular panel member
 - **Specialist:** Subject matter expert for specific areas
 - **External Liaison:** Coordinates with external reviewers
5. **Set Details:**
 - Expertise Area: Research specialization
 - Available: Yes/No toggle
 - Maximum concurrent assignments
 - Notes: Special skills or preferences
6. Click “**Save**”

1.7.2.2 Panel Member Roles Explained **Chair** - Leads panel meetings - Makes tie-breaking decisions - Coordinates panel activities - Reports to department head - Typically senior professor

Member - Regular review duties - Attends panel meetings - Provides quality assessments - Discusses borderline cases - Typically experienced academics

Specialist - Expert in specific research area - Consulted for specialist outputs - Provides detailed technical review - May not attend all meetings - Typically leading researcher in field

External Liaison - Coordinates with Critical Friends - Manages external communications - Tracks external reviews - Collates feedback - Typically diplomatic senior staff

1.7.3 Assigning Outputs for Internal Review

Method 1: From Output

1. View output details
2. Click “**Assign Internal Panel**”
3. Select panel member (filtered by expertise)
4. Set assignment details:
 - **Status:** Assigned (default)
 - **Priority:** High/Medium/Low
 - **Due Date:** Target completion
 - **Expected Rating:** Optional pre-assessment
5. Add **Review Notes:**
 - Specific aspects to focus on
 - Known concerns
 - Context information

6. Click “**Assign**”

Method 2: From Panel Member

1. View panel member profile
2. Click “**Assign Output**”
3. Select output from list
4. Set assignment details
5. Save

Method 3: Bulk Assignment

1. Go to “**Internal Panel**” → “**Bulk Assign**”
2. Select multiple outputs
3. Choose panel members
4. Distribute evenly or by expertise
5. Set common due date
6. Confirm

1.7.4 Assignment Status Tracking

Status Options:

Assigned - Initial state - Reviewer notified - Awaiting their review - Shows in “Pending” on dashboard

In Progress - Reviewer actively working - May have questions - Partial feedback possible - Shows in “Active” on dashboard

Completed - Review finished - Rating and feedback provided - Ready for panel discussion - Shows in “Completed” on dashboard

Deferred - Temporarily postponed - Needs more information - Awaiting author revision - Shows in “Deferred” on dashboard

1.7.5 Recording Internal Reviews

When review is complete:

1. Open assignment
2. Click “**Record Review**”
3. Fill in **Review Details**:
 - **Rating Recommendation**: 4, 3, 2, 1
 - **Confidence Level**: High/Medium/Low
 - **Strengths**: What works well
 - **Weaknesses**: Areas of concern
 - **Recommendations**: Improvements needed
 - **Internal Notes**: For panel only
4. Set **Status** to “**Completed**”
5. Set **Completion Date**
6. Click “**Save**”

1.7.6 Internal Panel Dashboard Widget

Displays:

Internal Panel Summary

Panel Members: 8

- └─ Chair: 1
- └─ Members: 5
- └─ Specialists: 2
- └─ Available: 7

Active Assignments: 45

- └─ Assigned: 12
- └─ In Progress: 18
- └─ Completed: 78
- └─ Pending: 15

Average Rating: 3.2

- └─ 4*: 28 (36%)
- └─ 3*: 35 (45%)
- └─ 2*: 12 (15%)
- └─ 1*: 3 (4%)

Workload Distribution:

- └─ Prof. Smith: 8 active
- └─ Dr. Jones: 6 active
- └─ Others: Balanced

1.7.7 Panel Meetings

Use the system for: - Generate meeting agenda (exports) - Review disagreements (filter by rating variance) - Track decisions made - Record consensus ratings - Assign follow-up actions

Meeting Workflow:

1. Before Meeting:

- Export assignments by status
- Identify contentious outputs
- Prepare discussion list
- Send to panel

2. During Meeting:

- Review ratings
- Discuss disagreements
- Reach consensus
- Assign actions

3. After Meeting:

- Update ratings in system
- Record decisions
- Assign re-reviews if needed
- Send outputs to Critical Friends

1.7.8 Internal vs External Review Workflow

Typical Process:

1. Output Added
↓
2. Internal Panel Assignment
↓
3. Internal Review Completed
↓
4. Panel Discussion
↓
5. Provisional Rating Set
↓
6. Critical Friend Assignment
↓
7. External Review Received
↓
8. Compare Internal vs External
↓
9. Final Rating Decision
↓
10. REF Submission

1.7.9 Best Practices

1. **Chair Leadership:** Strong chair essential for consistency
 2. **Balanced Workload:** Distribute assignments fairly
 3. **Expertise Matching:** Assign by subject area
 4. **Timely Reviews:** Set realistic deadlines
 5. **Regular Meetings:** Monthly panel meetings
 6. **Document Decisions:** Record all rating changes
 7. **Internal First:** Always review internally before external
 8. **Use Dashboard:** Monitor progress daily
 9. **Consistent Criteria:** Use same standards throughout
 10. **Feedback Loop:** Share learning across panel
-

1.8 Task Management NEW in v2.0

Track all REF-related activities beyond just papers and reviews.

1.8.1 About Tasks

What are Tasks? - Action items related to REF submission - Deadlines and milestones - Administrative activities - Meeting actions - Documentation requirements

Why use Tasks? - Never miss deadlines - Coordinate team activities - Track progress - Assign responsibilities - Audit trail

1.8.2 Creating a Task

Step-by-step:

1. Click “**Tasks**” in navigation
2. Click “**Create Task**” button
3. Fill in **Task Details**: *
 - **Title**: Brief, clear description
 - **Description**: Full details and context
4. Set **Classification**:
 - **Category**: * Choose from:
 - Administrative
 - Submission
 - Review
 - Meeting
 - Documentation
 - Deadline
 - Other
 - **Priority**: * Choose from:
 - Low: Nice to have
 - Medium: Important
 - High: Very important
 - Urgent: Critical/immediate
5. Set **Timing**:
 - **Start Date**: When work begins
 - **Due Date**: * Completion deadline
6. **Assignment**:
 - **Assigned To**: Select user (optional)
 - **Created By**: Auto-filled
7. Set **Status**: * Initial status:
 - Pending (default)
 - In Progress
 - Completed
 - Cancelled
8. Click “**Create Task**”

1.8.3 Task Categories Explained

Administrative - Paperwork and forms - Budget requests - Staff approvals - Policy compliance - Examples: “Complete REF declaration form”, “Submit budget request”

Submission - Direct submission activities - Data preparation - Format compliance - Upload activities - Examples: “Prepare final output list”, “Upload PDFs to REF portal”

Review - Review coordination - Follow-up with reviewers - Collate feedback - Rating decisions - Examples: “Chase external reviews”, “Schedule panel meeting”

Meeting - Meeting organization - Agenda preparation - Attendance coordination - Minute taking - Examples: “Book room for panel meeting”, “Send meeting invites”

Documentation - Document creation - Report writing - Record keeping - Archive management - Examples: “Write quality profile report”, “Document rating decisions”

Deadline - Critical dates - Submission cutoffs - External deadlines - Milestone markers - Examples: “REF portal opens”, “Final submission deadline”

Other - Miscellaneous tasks - One-off activities - Uncategorized items - Examples: “Research REF guidelines”, “Contact other departments”

1.8.4 Priority Levels

Low Priority - Can wait - No immediate deadline - Nice to have - Examples: “Update team photos”, “Improve documentation”

Medium Priority - Normal importance - Standard deadlines - Regular workflow - Examples: “Add new outputs”, “Schedule reviews”

High Priority - Very important - Near-term deadline - Significant impact - Examples: “Complete quality ratings”, “Submit internal review”

Urgent Priority - Critical/immediate - Overdue or due today - Blocks other work - Examples: “Final submission TODAY”, “Emergency panel meeting”

1.8.5 Managing Tasks

1.8.5.1 Updating Status As work progresses:

1. Open task
2. Click “**Edit**”
3. Change **Status**:
 - Pending → In Progress (when starting)
 - In Progress → Completed (when done)
 - Any → Cancelled (if no longer needed)
4. Add **Notes**: What was done
5. Save

1.8.5.2 Completing Tasks Quick completion:

1. View task
2. Click “**Mark Complete**” button
3. Confirm
4. Task marked complete with timestamp

1.8.5.3 Filtering Tasks Filter options: - By category - By priority - By status - By assigned user - By date range - Overdue only - My tasks only

1.8.5.4 Sorting Tasks Sort by: - Due date (soonest first) - Default - Priority (urgent first) - Created date - Status - Category

1.8.6 Task Dashboard Widget

Displays:

Task Overview

Urgent Tasks: 3 □

- └ Final submission (Due: Today)
- └ Emergency review (Due: Tomorrow)
- └ Budget approval (Due: This week)

Overdue Tasks: 2 ▲

- └ Panel meeting minutes
- └ External reviewer follow-up

By Status:

- └ Pending: 8
- └ In Progress: 12
- └ Completed: 156
- └ Cancelled: 4

By Priority:

- └ Urgent: 3
- └ High: 7
- └ Medium: 10
- └ Low: 4

1.8.7 Task List Views

Main Task List: - All tasks visible - Filter and sort controls - Quick status indicators
- Priority badges - Due date with countdown - Assignment info

My Tasks: - Tasks assigned to you - Personalized view - Focus on your work - Quick access

Team Tasks: - All team tasks - See workload distribution - Coordinate efforts - Track progress

1.8.8 Task Best Practices

1. **Be Specific:** Clear, actionable titles
2. **Set Realistic Deadlines:** Don't over-commit
3. **Use Categories:** Helps with filtering and reporting
4. **Assign Properly:** Right person for the job
5. **Update Promptly:** Keep status current
6. **Add Context:** Use description field fully
7. **Check Daily:** Review your tasks each morning
8. **Close Completed:** Mark done promptly
9. **Review Weekly:** Team task review
10. **Archive Old:** Keep list manageable

1.8.9 Task Examples

Good Task Examples:

- "Complete quality ratings for 15 outputs in UoA 27"
Category: Review, Priority: High, Due: 2 weeks
- "Send review assignments to Prof. Smith (5 papers)"
Category: Review, Priority: Medium, Due: 1 week
- "Finalize submission document for REF portal"
Category: Submission, Priority: Urgent, Due: 3 days
- "Schedule panel meeting for borderline cases"
Category: Meeting, Priority: High, Due: 1 week

Poor Task Examples:

- "Do stuff" - Too vague
 - "Things" - No context
 - "Review" - What needs reviewing?
 - "REF" - Too broad
-

1.9 □ Request Management

Track queries and issues from staff, administration, and external parties.

1.9.1 About Requests

What are Requests? - Queries about REF process - Requests for information - Support tickets - Issue tracking - Communication log

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

1.9.2 Creating a Request

Step-by-step:

1. Click "**Requests**" in navigation
2. Click "**Create Request**" button
3. Fill in **Request Details:** *
 - **Subject:** Brief description
 - **Description:** Full details
 - **Request Type:** * Select category
4. **Associate with Output** (if relevant):
 - Link to specific output
 - Context for the request
5. Set **Details:**

- **Priority:** High/Medium/Low
- **Status:** Pending (default)
- **Due Date:** Target response date
- **Requester:** Contact info

6. **Assignment:**

- Assign to colleague (optional)
- Auto-assigned to creator

7. Click “**Submit Request**”

1.9.3 Request Status Flow

Status Progression:

Pending

↓

In Progress (working on it)

↓

Completed □ NEW

OR

Any Status

↓

Deleted (with confirmation) □ NEW

1.9.4 Managing Requests

1.9.4.1 Updating Requests

1. View request
2. Click “**Edit**”
3. Update status/details
4. Add notes
5. Save

1.9.4.2 Mark as Completed □ NEW in v2.0 Quick completion:

1. View request
2. Click “**Mark as Complete**” button (green)
3. Confirm on completion page
4. Request marked completed with timestamp
5. Success message displayed

What happens: - Status set to “Completed” - Completion timestamp recorded - Visible in completed requests list - “Mark Complete” button hidden (for completed requests)

1.9.4.3 Delete Request □ NEW in v2.0 When to delete: - Duplicate requests - Spam or invalid requests - Accidentally created - No longer relevant

How to delete:

1. View request
2. Click “**Delete**” button (red)
3. Warning confirmation page appears
4. Read warning about permanent deletion
5. Click “**Confirm Delete**” to proceed
6. Request permanently removed
7. Success message shown

Warning: - Deletion is permanent - Cannot be undone - All associated data lost - Consider marking as “Completed” instead

1.9.5 Request List Features

List Display: - Status badges - Priority indicators - Associated output links - Requester information - Due dates - Quick actions

Action Buttons: ENHANCED in v2.0 - **View:** See full details - **Edit:** Update request - **Complete:** Mark as done (if not completed) - **Delete:** Remove permanently

Conditional Display: - Complete button hidden for completed requests - Delete always available - Edit available for all statuses

1.9.6 Request Detail View

Information Shown: - Full request details - Status history - Related output (if linked) - Requester contact - Priority and dates - Internal notes

Actions Card: - Update status - Mark complete - Edit details - Delete request - Email requester

1.9.7 Request Dashboard

Displays: - Total open requests - By status (Pending, In Progress, Completed) - By priority - Overdue requests - Recent activity - Assignment distribution

1.9.8 Best Practices

1. **Prompt Response:** Acknowledge within 24 hours
 2. **Clear Communication:** Keep requester informed
 3. **Link Outputs:** Associate with relevant papers
 4. **Track Progress:** Update status regularly
 5. **Complete When Done:** Mark as complete promptly NEW
 6. **Don't Delete Unless Sure:** Use completion instead NEW
 7. **Use Priorities:** Helps with workload management
 8. **Documentation:** Record decisions and actions
 9. **Follow Up:** Check on overdue requests
 10. **Archive Strategy:** Regular review of old requests
-

1.10 □ Data Import/Export □ ENHANCED in v2.0

Efficiently move data in and out of the system.

1.10.1 CSV Import □ NEW

1.10.1.1 Output Import When to use: - Importing many outputs at once - Migrating from other systems - Updating multiple outputs - Annual import of publications

Process:

1. Prepare Data:

- Go to Outputs → Import
- Download CSV template
- Fill in required columns
- Save as UTF-8 CSV

2. Required Columns:

- title
- all_authors
- publication_venue
- publication_type
- publication_date (YYYY-MM-DD)
- quality_rating (4, 3, 2, 1, or 0 for unclassified)
- author_email (links to colleague)
- uoa
- ref_eligible (true/false)

3. Optional Columns:

- doi
- url
- volume, issue, pages
- keywords
- abstract
- notes

4. Upload:

- Choose your CSV file
- Click “Upload and Validate”
- System checks data

5. Validation:

- Checks required fields
- Validates dates and ratings
- Matches email to colleagues
- Detects duplicate titles/DOIs
- Shows error report if issues

6. Review:

- See what will be imported
- Check colleague matches
- Review any warnings
- Check duplicate detection

7. Import:

- Click “Confirm Import”

- Outputs created
- Success summary shown
- Error log if any issues

CSV Tips: - Use template - exact column names required - UTF-8 encoding essential - Test with 2-3 rows first - External co-authors auto-categorized as “Co-author (External)” - Empty cells okay for optional fields - Dates must be YYYY-MM-DD format - Quality ratings: 4, 3, 2, 1, or 0 - Boolean fields: true/false (lowercase)

1.10.1.2 Colleague Import Process similar to outputs: - Download colleague template - Fill in colleague data - Include employment status and category
 NEW - Upload and validate - Review matches - Confirm import

Special handling: - Duplicate detection by email - Auto-links to User accounts if exist - Creates new users if needed - Can update existing colleagues

1.10.2 Excel Export NEW

1.10.2.1 Assignment Export with Links Perfect for: - Distributing papers to reviewers - Progress tracking - Meeting preparation - Email communication

Export Options:

- 1. Critical Friends Assignments:** - Yellow highlighted - External reviewer information - Paper links - Due dates and priorities
- 2. Internal Panel Assignments:** - Blue highlighted - Internal reviewer information - Paper links - Status and ratings
- 3. Combined Export:** - Both types color-coded - Complete overview - Workload comparison

Export Process:

1. Go to **Export → “Assignments”**
2. Set filters:
 - **Reviewer Type:** Internal/Critical Friends/Both
 - **Specific Reviewer:** (optional)
 - **Author:** (optional)
 - **Status:** (optional)
3. Click **“Export to Excel”**
4. File downloads automatically

Excel Features: - **Clickable PDF Links:** Direct access to papers - **Color Coding:** - Light Blue: Internal Panel - Light Yellow: Critical Friends - **Frozen Header:** Easy scrolling - **Formatted Columns:** Professional appearance - **Complete Information:** - Reviewer name and email - Output title - Author - Quality rating - Review status - Due date - Priority - PDF link - Notes

Using the Export:

Scenario 1: Email to Single Reviewer

1. Export filtered by specific reviewer
2. Reviewer sees only their assignments

3. Click links to access papers
4. Send via email with instructions

Scenario 2: Team Meeting

1. Export all assignments
2. Show on projector/screen
3. Discuss progress
4. Identify bottlenecks
5. Assign new reviews

Scenario 3: Progress Report

1. Export by status (completed)
2. Show completion rates
3. Calculate review times
4. Identify trends
5. Report to management

1.10.3 Other Export Options

1.10.3.1 Output Export CSV Export: - All output data - Custom field selection
- Filtered export - Date range export

Process: 1. Go to Outputs 2. Apply filters 3. Click “Export CSV” 4. Opens in Excel

1.10.3.2 Colleague Export Export Types: - Current staff list - Former staff list
- Category breakdown - Output statistics

Format: CSV

1.10.3.3 Report Export LaTeX/PDF Reports: - Submission overview - Quality profile - Staff progress - Review status - Comprehensive report

Process: 1. Go to Reports 2. Select report type 3. Choose format (LaTeX/Article/Report/Beamer) 4. Generate 5. Download

1.10.4 Import/Export Best Practices

1. **Regular Exports:** Weekly backups
2. **Test Imports:** Small batches first
3. **Clean Data:** Remove special characters
4. **UTF-8 Encoding:** Always
5. **Template Use:** Don’t create from scratch
6. **Validation:** Check import results
7. **Backup First:** Before bulk import
8. **Documentation:** Keep import logs
9. **Excel for Distribution:** Use export feature □ NEW
10. **CSV for Migration:** Bulk data movement

1.11 □ Report Generation

Create professional reports for stakeholders.

1.11.1 Available Reports

1.11.1.1 1. Submission Overview Contents: - Total outputs by UoA - Quality profile breakdown - Staff contribution summary - Timeline and milestones - Key statistics

Best for: - Department meetings - Progress updates - Management reporting

1.11.1.2 2. Quality Profile Analysis Contents: - Detailed quality breakdown - Comparison to targets - Trend analysis - Recommendations - Critical concerns

Best for: - Strategy meetings - Quality improvement - REF preparation

1.11.1.3 3. Staff Progress Report Contents: - Individual staff contributions - Required vs. actual outputs - Quality by staff member - Productivity analysis - Staff categories breakdown □ NEW

Best for: - Performance reviews - Resource allocation - Staff development

1.11.1.4 4. Review Status Report Contents: - Internal panel progress - Critical friends status - Pending reviews - Completed reviews - Turnaround times

Best for: - Review coordination - Deadline management - Workload balancing

1.11.1.5 5. Comprehensive Report Contents: - ALL of the above combined - Executive summary - Detailed analysis - Appendices - Complete data

Best for: - Final submission - External audits - Complete overview

1.11.2 Report Formats

LaTeX Source (.tex) - Editable source file - Compile with xelatex/pdfflatex - Customize styling - Add institution branding

PDF (via LaTeX) - Professional appearance - Ready to distribute - Print-ready - Includes graphics

Document Styles: 1. **Article:** Standard academic article format 2. **Report:** Multi-chapter report format 3. **Beamer:** Presentation slides

1.11.3 Generating Reports

Process:

1. Go to **Reports**
2. Click “**Generate Report**”
3. Select **Report Type**
4. Choose **Format**:
 - LaTeX Source
 - PDF (requires LaTeX)
 - Beamer Presentation
5. Set **Options**:
 - Date range
 - Include/exclude sections
 - Filtering options
6. Click “**Generate**”
7. Download when ready

1.11.4 Report Customization

LaTeX reports can be customized: - Institution logo - Color scheme - Header/footer - Section ordering - Chart styles - Data included

Edit the .tex file: - Change title and author - Add/remove sections - Adjust formatting - Include additional analysis

Compile:

```
xelatex report.tex  
xelatex report.tex # Second run for TOC
```

1.11.5 Report Best Practices

1. **Regular Generation**: Monthly reports
 2. **Archive Reports**: Keep version history
 3. **Customize**: Add institution branding
 4. **Review Before Sharing**: Check for errors
 5. **PDF for Distribution**: Most compatible
 6. **LaTeX for Editing**: More flexibility
 7. **Beamer for Presentations**: Quick slides
 8. **Data Quality**: Ensure data current first
-

1.12 □ Best Practices

1.12.1 Data Quality

1. **Complete Information**: Fill all fields
2. **Accurate Dates**: Check publication dates

3. **Verified Links:** Test DOI and URLs
4. **Current Status:** Update employment promptly
5. **Consistent Categories:** Use categories correctly
6. **Quality Evidence:** Document rating decisions

1.12.2 Workflow Efficiency

1. **Use CSV Import:** For bulk operations
2. **Excel Export:** For reviewer distribution □ NEW
3. **Dashboard Daily:** Check each morning
4. **Task Management:** Track all activities □ NEW
5. **Regular Updates:** Keep data current
6. **Batch Operations:** Group similar tasks

1.12.3 Quality Assurance

1. **Internal First:** Always review internally before external
2. **Multiple Opinions:** Get several reviews per output
3. **Panel Meetings:** Regular discussion of ratings
4. **Documentation:** Record all decisions
5. **Consistency:** Apply criteria uniformly
6. **Evidence-Based:** Use citations, rankings

1.12.4 Team Coordination

1. **Clear Roles:** Define responsibilities
2. **Regular Meetings:** Weekly or bi-weekly
3. **Communication:** Keep team informed
4. **Task Assignment:** Use task system □ NEW
5. **Progress Tracking:** Monitor dashboards
6. **Feedback Loop:** Learn and improve

1.12.5 Data Management

1. **Regular Backups:** Daily database backups
2. **Version Control:** Track changes
3. **Access Control:** Appropriate permissions
4. **Data Validation:** Check imports carefully
5. **Archive Old Data:** Keep system fast
6. **Export Regularly:** Multiple backup formats

1.12.6 Security

1. **Strong Passwords:** Enforce policy
2. **Regular Updates:** Keep Django current
3. **Access Logs:** Monitor usage

4. **HTTPS**: Use SSL in production
 5. **Backup Security**: Encrypt backups
 6. **User Training**: Security awareness
-

1.13 □ Frequently Asked Questions

1.13.1 General Questions

Q: What's the difference between Internal Panel and Critical Friends? A: Internal Panel are university staff reviewing internally. Critical Friends are external reviewers from other institutions. Use Internal Panel for first review and quality assurance, Critical Friends for external validation.

Q: Should I delete former staff? A: NO! Mark them as "Former" instead. This preserves their outputs and contribution history while removing them from active dropdowns.

Q: What's the difference between colleague categories? A: Categories identify research roles: Independent researchers can submit independently, Non-independent (like post-docs) may have different rules, External co-authors are from other institutions. Choose the category that best describes their status for REF purposes.

Q: How often should I back up? A: Daily for production systems. Weekly minimum for development. Use automated backup scripts.

1.13.2 Output Management

Q: What if an output has multiple authors from our department? A: Select the lead author in the "Author" field. All authors are listed in "All Authors" field. The lead author gets credit for required output calculation.

Q: Can I change quality ratings later? A: Yes! Ratings can be updated as you get more reviews and information. Track the reasons for changes.

Q: What if I don't know the quality rating yet? A: Use "Unclassified" initially. Update after internal and external reviews.

Q: How do I handle a book with multiple chapters? A: Each chapter can be a separate output if eligible. Or treat the whole book as one output. Follow REF guidelines for your discipline.

1.13.3 Import/Export □ NEW

Q: Why did my CSV import fail? A: Common issues: - File not UTF-8 encoded - Missing required columns - Invalid dates (use YYYY-MM-DD) - Email doesn't match colleague in system - Special characters in data

Q: Can I update existing outputs via CSV? A: Yes! If title or DOI matches, you can choose to update existing outputs. Be careful not to overwrite good data.

Q: Why can't I click the links in my exported Excel? A: Links work when PDFs are uploaded and accessible. Check: - PDF was uploaded for that output - You have network access to server - File path is correct

Q: Can I email the Excel export directly to reviewers? A: Yes! That's exactly what it's designed for. Reviewers can click links to access papers directly.

1.13.4 Tasks □ NEW

Q: Should I create a task for every output? A: No. Tasks are for administrative activities, deadlines, meetings, and coordination. Use assignments for output reviews.

Q: Can I assign one task to multiple people? A: Not directly. Create separate tasks or use the description to mention multiple people involved.

Q: What happens to completed tasks? A: They remain in the system for audit trail. Filter by status to hide completed tasks from daily view.

Q: When should I use “Urgent” priority? A: For critical items due today or tomorrow, or things blocking other work. Don't overuse - saves meaning for true urgencies.

1.13.5 Reviews

Q: How many reviews should each output get? A: Recommended: 1-2 internal panel reviews plus 1 external critical friend review. More for borderline cases.

Q: What if internal and external reviews disagree? A: Common! This is why you need both. Panel should discuss, consider evidence, and make final decision. Document reasoning.

Q: How long should reviews take? A: Internal: 1-2 weeks. External: 2-4 weeks. Build in buffer time for delays.

Q: Can Critical Friends see internal reviews? A: No, they're separate systems. Internal reviews inform your decisions, external reviews provide validation. Don't share internal reviews unless you want to.

1.13.6 Technical

Q: What if the server stops? A: If using systemd:

```
sudo systemctl status ref-manager  
sudo systemctl restart ref-manager
```

Check logs for errors:

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

Q: How do I add another user? A: Via Django admin:

```
python manage.py createsuperuser
```

Or through admin panel (<http://localhost:8000/admin/>).

Q: Can I access from multiple computers? A: Yes! If running on a server: 1. Set ALLOWED_HOSTS in .env 2. Run server on 0.0.0.0:8000 3. Access via <http://server-ip:8000> For production, use Nginx/Gunicorn setup.

Q: How do I upgrade to a new version? A: 1. Backup database 2. Pull/download new code 3. Run: pip install -r requirements.txt 4. Run: python manage.py migrate 5. Run: python manage.py collectstatic 6. Restart server

1.13.7 Troubleshooting

Q: I can't log in! A: Reset password:

```
python manage.py changepassword yourusername
```

Q: Images/CSS not loading? A: Run:

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

Then restart server.

Q: “No such table” error? A: Run migrations:

```
python manage.py migrate
```

Q: Python 3.13 errors? A: v2.0 is compatible with Python 3.13. Update to latest version. If issues persist, convert Decimal to float in calculations (already done in v2.0).

1.14 Getting Help

Documentation: - README.md - System overview and installation - QUICK-START-GUIDE.md - 10-minute setup - TECHNICAL-DOCUMENTATION.md - Developer guide - TROUBLESHOOTING.md - Problem solving - CHANGELOG.md - Version history

Support: - System Administrator: [email] - Department IT: [email] - User Community: [forum/chat]

Reporting Issues: Include: - Django version - Python version - Operating system - Error message (full) - Steps to reproduce

Feature Requests: Submit via: - GitHub Issues - Email to admin - Department meetings

Version: 2.0.0

Last Updated: November 3, 2025

Prepared by: George Tsoulas

Department: Language and Linguistic Science, University of York

This guide covers all features of REF Manager v2.0. For additional support, refer to other documentation files or contact your system administrator.

1.15 ☐ Additional Resources

REF Guidelines: - REF 2029 Official Guidance: <https://www.ref.ac.uk/> - Panel Criteria and Working Methods - Submission Guidelines - Assessment Framework

Training Materials: - Video tutorials (if available) - Training sessions schedule - Quick reference cards - Workshop materials

Community: - User group meetings - Best practice sharing - Cross-department collaboration - External REF manager network

Happy REF Managing! ☐

For the latest updates and detailed technical information, please refer to the complete documentation suite.

Contents

1 REF Manager v2.0 - Technical Documentation	2
1.1 □ Table of Contents	3
1.2 □ Architecture Overview	3
1.2.1 System Architecture	3
1.2.2 Application Flow	4
1.3 □ Technology Stack	4
1.3.1 Backend	4
1.3.2 Frontend	5
1.3.3 Database	5
1.3.4 Server Infrastructure	5
1.3.5 Development Tools	5
1.4 □ Project Structure	5
1.4.1 Key Files Explained	7
1.5 □ Database Models	8
1.5.1 Model Relationships Diagram	8
1.5.2 Core Models	9
1.5.3 Database Queries Optimization	13
1.6 □ URL Configuration	14
1.6.1 Root URLs (ref_manager/urls.py)	14
1.6.2 App URLs (core/urls.py)	14
1.6.3 URL Pattern Naming Convention	15
1.7 □ Views and Business Logic	15
1.7.1 View Structure	15
1.7.2 Common View Patterns	15
1.7.3 Export Views (v2.0)	17
1.8 □ Forms and Validation	19
1.8.1 Form Classes	19
1.8.2 Custom Validation	20
1.9 □ Templates and Frontend	21
1.9.1 Template Inheritance	21
1.9.2 Base Template (base.html)	21
1.9.3 Template Tags and Filters	23
1.10 □ Static Files and Media	24

1.10.1 Static Files Structure	24
1.10.2 Media Files	25
1.10.3 Static Files Configuration	25
1.10.4 Collecting Static Files	25
1.11 □ Security Considerations	25
1.11.1 Authentication and Authorization	25
1.11.2 CSRF Protection	26
1.11.3 SQL Injection Prevention	26
1.11.4 XSS Prevention	26
1.11.5 File Upload Security	26
1.11.6 Production Security Settings	27
1.12 □ Testing	27
1.12.1 Test Structure	27
1.12.2 Running Tests	29
1.13 ↵ Performance Optimization	29
1.13.1 Database Optimization	29
1.13.2 Caching	30
1.13.3 Query Optimization Tips	31
1.14 □ Deployment Guide	31
1.15 □□ Development Workflow	32
1.15.1 Setting Up Development Environment	32
1.15.2 Code Style	32
1.15.3 Git Workflow	32
1.15.4 Adding New Features	33
1.16 □ Additional Resources	34

1 REF Manager v2.0 - Technical Documentation

Developer and System Administrator Reference

Version: 2.0.0

Last Updated: November 3, 2025

For: Developers, System Administrators, Technical Staff

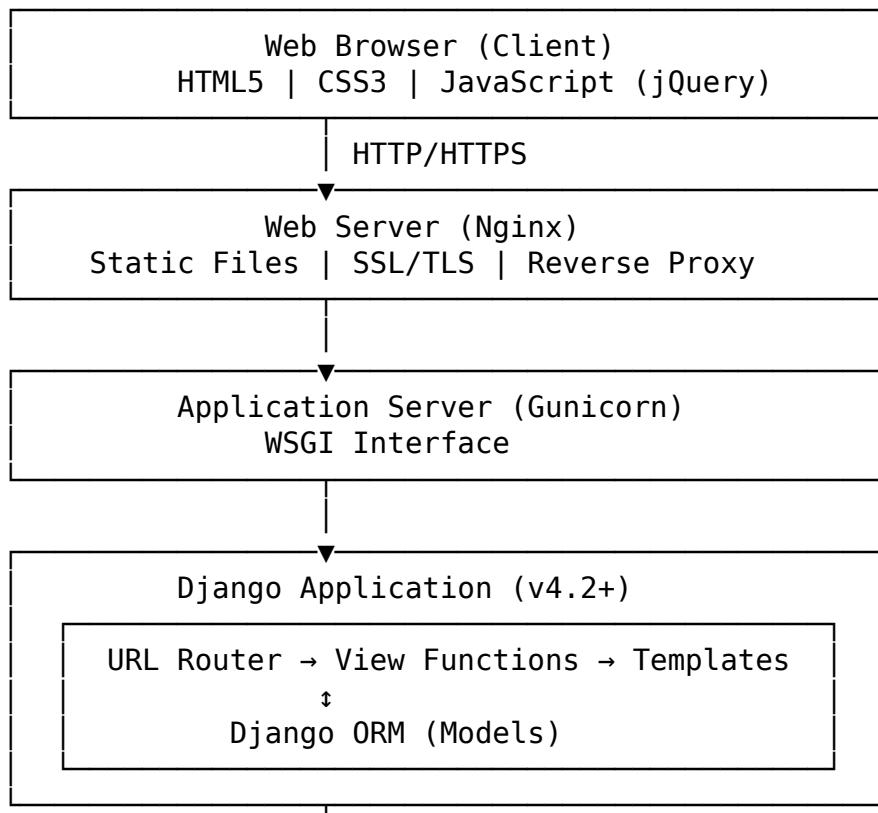
1.1 □ Table of Contents

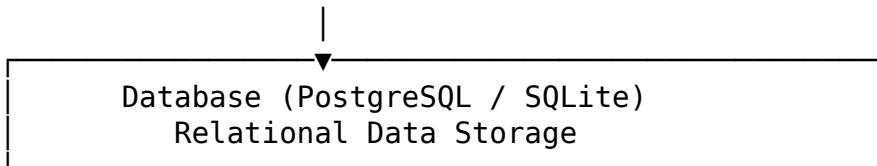
1. [Architecture Overview](#)
 2. [Technology Stack](#)
 3. [Project Structure](#)
 4. [Database Models](#)
 5. [URL Configuration](#)
 6. [Views and Business Logic](#)
 7. [Forms and Validation](#)
 8. [Templates and Frontend](#)
 9. [Static Files and Media](#)
 10. [Security Considerations](#)
 11. [Testing](#)
 12. [Performance Optimization](#)
 13. [API Reference](#)
 14. [Deployment Guide](#)
 15. [Development Workflow](#)
-

1.2 □ Architecture Overview

REF Manager follows Django's MTV (Model-Template-View) architecture pattern with a clear separation of concerns.

1.2.1 System Architecture





1.2.2 Application Flow

```

User Request
  ↓
Nginx (port 80/443)
  ↓
Gunicorn (port 8000)
  ↓
Django URL Dispatcher (urls.py)
  ↓
View Function (views.py)
  ↓
Model Query (models.py via ORM)
  ↓
Database (PostgreSQL/SQlite)
  ↓
Template Rendering (templates/)
  ↓
HTTP Response
  ↓
User Browser
  
```

1.3 ☐ Technology Stack

1.3.1 Backend

Core Framework: - **Django 4.2+**: Web framework - **Python 3.10+**: Programming language (tested up to 3.13)

Key Libraries: - `django-crispy-forms`: Form rendering - `crispy-bootstrap4`: Bootstrap 4 integration - `openpyxl`: Excel file handling - `python-dotenv`: Environment management - `gunicorn`: WSGI HTTP server - `psycopg2-binary`: PostgreSQL adapter

Python Standard Library Used: - `datetime`, `timezone`: Date/time handling - `decimal.Decimal`: Precise calculations - `json`: JSON processing - `csv`: CSV file handling - `io.BytesIO`: In-memory file operations - `collections.defaultdict`: Data structures

1.3.2 Frontend

UI Framework: - **Bootstrap 4.6:** Responsive design - **Font Awesome 5:** Icons - **jQuery 3.6:** DOM manipulation

Styling: - Custom CSS for REF Manager branding - Bootstrap utilities - Responsive layouts

1.3.3 Database

Development: - **SQLite 3:** File-based database - Zero configuration - Perfect for development

Production: - **PostgreSQL 12+:** Robust relational database - ACID compliance - Advanced features

1.3.4 Server Infrastructure

Production Stack: - **Nginx:** Web server and reverse proxy - **Gunicorn:** Python WSGI HTTP server - **Ubuntu 20.04+:** Operating system - **systemd:** Service management

1.3.5 Development Tools

- **Git:** Version control
 - **pip:** Package management
 - **venv:** Virtual environments
 - **Django Debug Toolbar:** Development debugging (optional)
-

1.4 Project Structure

```
ref-manager/
  └── ref_manager/                                # Project configuration
    ├── __init__.py
    ├── settings.py                               # Django settings
    ├── urls.py                                  # Root URL configuration
    ├── wsgi.py                                   # WSGI application
    └── asgi.py                                  # ASGI application (future)

  └── core/                                       # Main application
    └── migrations/                             # Database migrations
      ├── __init__.py
      ├── 0001_initial.py
      ├── 0002_employment_status.py      # v2.0
      └── 0003_colleague_categories.py   # v2.0
```

```
    └── 0004_internal_panel.py          # v2.0
        └── 0005_tasks.py              # v2.0

    └── templatetags/                 # Custom template filters
        ├── __init__.py
        └── custom_filters.py         # Custom filters for templates

    └── __init__.py
    ├── models.py                   # Data models (1000+ lines)
    ├── views.py                    # View functions (2000+ lines)
    ├── views_export.py            # Export views (v2.0, 500+ lines)
    ├── forms.py                   # Form definitions (800+ lines)
    ├── urls.py                    # URL patterns (150+ lines)
    ├── admin.py                   # Admin configuration (300+ lines)
    ├── tests.py                   # Test cases
    └── apps.py                    # App configuration

    └── templates/                  # HTML templates
        ├── base.html
        └── registration/
            ├── login.html
            └── password_reset.html

    └── core/                       # App-specific templates
        ├── dashboard.html
        ├── colleague_list.html
        ├── colleague_detail.html
        ├── colleague_form.html
        ├── colleague_confirm_delete.html
        ├── output_list.html
        ├── output_detail.html
        ├── output_form.html
        ├── output_import.html      # v2.0
        ├── criticalfriend_list.html
        ├── criticalfriend_detail.html
        ├── criticalfriend_form.html
        ├── internalpanel_list.html # v2.0
        ├── internalpanel_detail.html # v2.0
        ├── internalpanel_form.html # v2.0
        ├── task_list.html          # v2.0
        ├── task_detail.html        # v2.0
        ├── task_form.html          # v2.0
        ├── request_list.html
        ├── request_detail.html
        └── request_form.html
```

```

    └── request_confirm_complete.html # v2.0
    └── request_confirm_delete.html # v2.0

    └── export_assignments.html      # v2.0
        └── report_generate.html

    └── static/                      # Static files
        └── css/
            └── style.css           # Custom styles
        └── js/
            └── script.js          # Custom JavaScript
        └── img/
            └── logo.png           # REF Manager logo

    └── staticfiles/                 # Collected static files (production)

    └── media/                      # User-uploaded files
        └── pdfs/                  # Research paper PDFs

    └── logs/                       # Application logs
        └── django.log
        └── gunicorn-access.log
        └── gunicorn-error.log

    └── backups/                    # Database backups

    └── documentation/             # Generated documentation
        └── pdf/
        └── latex/                  # LaTeX sources

    └── venv/                       # Virtual environment

    └── manage.py                  # Django management script
    └── requirements.txt           # Python dependencies
    └── .env                        # Environment variables (not in git)
    └── .env.example                # Environment template
    └── .gitignore                  # Git ignore rules
    └── gunicorn_config.py         # Gunicorn configuration
    └── README.md                  # Main documentation
    └── build_docs.sh              # Documentation build script

```

1.4.1 Key Files Explained

settings.py: Django configuration - Database settings - Installed apps - Middleware - Static/media file paths - Security settings

urls.py: URL routing - Root URL patterns - Include app URLs - Admin URLs

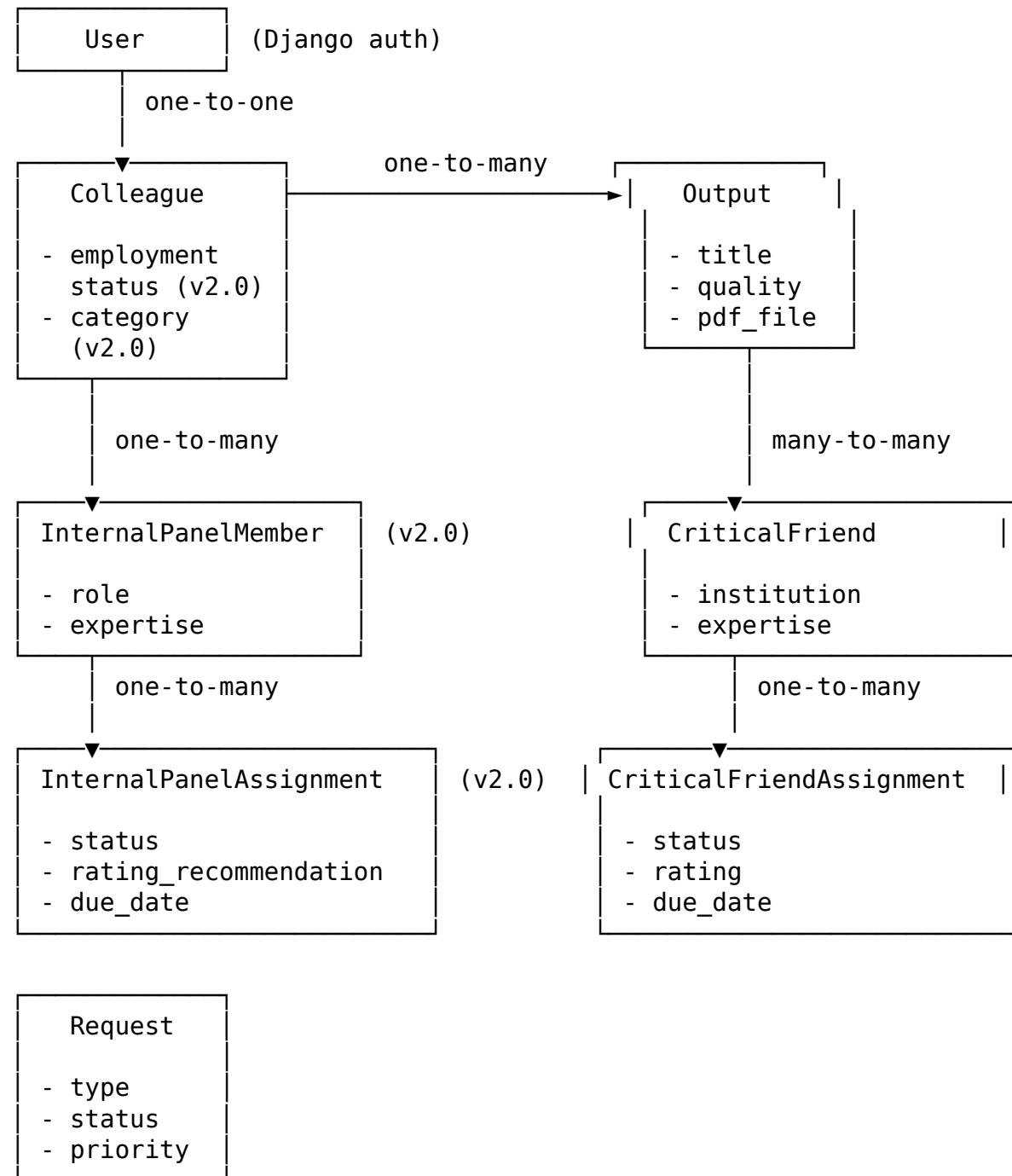
models.py: Data models - Colleague - Output - CriticalFriend, CriticalFriendAssignment - InternalPanelMember, InternalPanelAssignment (v2.0) - Request - Task (v2.0)

views.py: Business logic - Dashboard - CRUD operations - List/detail views - Form processing

views_export.py (v2.0): Export functionality - Excel export with links - CSV export - Assignment filtering

1.5 □ Database Models

1.5.1 Model Relationships Diagram



Task	(v2.0)
<ul style="list-style-type: none"> - category - priority - status - due_date 	

1.5.2 Core Models

1.5.2.1 Colleague Model

```
class Colleague(models.Model):
    """Staff member information"""

    user = models.OneToOneField(User, on_delete=models.CASCADE)
    title = models.CharField(max_length=50, blank=True)

    # Employment tracking (v2.0)
    employment_status = models.CharField(
        max_length=20,
        choices=[
            ('current', 'Current'),
            ('former', 'Former'),
        ],
        default='current'
    )
    employment_end_date = models.DateField(null=True, blank=True)

    # Category classification (v2.0)
    colleague_category = models.CharField(
        max_length=30,
        choices=[
            ('independent', 'Independent Researcher'),
            ('non_independent', 'Non-Independent Researcher'),
            ('postdoc', 'Post-Doctoral Researcher'),
            ('research_assistant', 'Research Assistant'),
            ('academic', 'Academic Staff'),
            ('support', 'Support Staff'),
            ('employee', 'Current Employee'),
            ('former', 'Former Employee'),
            ('coauthor', 'Co-author (External)'),
        ],
        default='employee'
    )

    fte = models.DecimalField(max_digits=3, decimal_places=2, default=1.0)
    unit_of_assessment = models.CharField(max_length=100, blank=True)
    office = models.CharField(max_length=100, blank=True)
```

```

phone = models.CharField(max_length=20, blank=True)
research_interests = models.TextField(blank=True)

@property
def required_outputs(self):
    """Calculate required outputs (FTE × 2.5)"""
    if self.employment_status == 'current':
        return float(self.fte) * 2.5 # Python 3.13 compatibility
    return 0

def __str__(self):
    return self.user.get_full_name()

```

1.5.2.2 Output Model

```

class Output(models.Model):
    """Research output/publication"""

    title = models.CharField(max_length=500)
    all_authors = models.TextField()
    publication_venue = models.CharField(max_length=300)

    PUBLICATION_TYPES = [
        ('article', 'Journal Article'),
        ('conference', 'Conference Paper'),
        ('book', 'Book'),
        ('chapter', 'Book Chapter'),
        ('monograph', 'Monograph'),
        ('report', 'Report'),
        ('other', 'Other'),
    ]
    publication_type = models.CharField(max_length=20, choices=PUBLICATION_TYPES)

    publication_date = models.DateField()
    volume = models.CharField(max_length=50, blank=True)
    issue = models.CharField(max_length=50, blank=True)
    pages = models.CharField(max_length=50, blank=True)
    doi = models.CharField(max_length=200, blank=True)
    url = models.URLField(max_length=500, blank=True)

    pdf_file = models.FileField(upload_to='pdfs/', null=True, blank=True)

    # REF specifics
    colleague = models.ForeignKey(
        Colleague,
        on_delete=models.CASCADE,
        related_name='outputs'
    )
    unit_of_assessment = models.CharField(max_length=100)

    QUALITY_CHOICES = [

```

```

        (4, '4* World-Leading'),
        (3, '3* Internationally Excellent'),
        (2, '2* Recognized Internationally'),
        (1, '1* Recognized Nationally'),
        (0, 'Unclassified'),
    ]
quality_rating = models.IntegerField(choices=QUALITY_CHOICES, default=0)

ref_eligible = models.BooleanField(default=True)

STATUS_CHOICES = [
    ('draft', 'Draft'),
    ('under_review', 'Under Review'),
    ('accepted', 'Accepted'),
    ('published', 'Published'),
]
status = models.CharField(max_length=20, choices=STATUS_CHOICES, default='draft')

keywords = models.CharField(max_length=500, blank=True)
abstract = models.TextField(blank=True)
notes = models.TextField(blank=True)

created_at = models.DateTimeField(auto_now_add=True)
updated_at = models.DateTimeField(auto_now=True)

def __str__(self):
    return self.title

```

1.5.2.3 InternalPanelMember Model (v2.0)

```

class InternalPanelMember(models.Model):
    """Internal panel member for evaluation"""

    colleague = models.ForeignKey(
        Colleague,
        on_delete=models.CASCADE,
        related_name='panel_memberships'
    )

    ROLE_CHOICES = [
        ('chair', 'Chair'),
        ('member', 'Member'),
        ('specialist', 'Specialist'),
        ('external_liaison', 'External Liaison'),
    ]
    role = models.CharField(max_length=20, choices=ROLE_CHOICES, default='member')

    expertise_area = models.CharField(max_length=200, blank=True)
    available = models.BooleanField(default=True)
    max_assignments = models.IntegerField(default=10)
    notes = models.TextField(blank=True)

```

```

    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f'{self.colleague.user.get_full_name()} ({self.get_role_display()})'

```

1.5.2.4 Task Model (v2.0)

```

class Task(models.Model):
    """General task tracking"""

    title = models.CharField(max_length=200)
    description = models.TextField()

    CATEGORY_CHOICES = [
        ('administrative', 'Administrative'),
        ('submission', 'Submission'),
        ('review', 'Review'),
        ('meeting', 'Meeting'),
        ('documentation', 'Documentation'),
        ('deadline', 'Deadline'),
        ('other', 'Other'),
    ]
    category = models.CharField(max_length=20, choices=CATEGORY_CHOICES)

    PRIORITY_CHOICES = [
        ('low', 'Low'),
        ('medium', 'Medium'),
        ('high', 'High'),
        ('urgent', 'Urgent'),
    ]
    priority = models.CharField(max_length=10, choices=PRIORITY_CHOICES, default='m')

    STATUS_CHOICES = [
        ('pending', 'Pending'),
        ('in_progress', 'In Progress'),
        ('completed', 'Completed'),
        ('cancelled', 'Cancelled'),
    ]
    status = models.CharField(max_length=15, choices=STATUS_CHOICES, default='pending')

    assigned_to = models.ForeignKey(
        User,
        on_delete=models.SET_NULL,
        null=True,
        blank=True,
        related_name='assigned_tasks'
    )
    created_by = models.ForeignKey(
        User,
        on_delete=models.CASCADE,

```

```

        related_name='created_tasks'
    )

start_date = models.DateField(null=True, blank=True)
due_date = models.DateField()
completed_at = models.DateTimeField(null=True, blank=True)

created_at = models.DateTimeField(auto_now_add=True)
updated_at = models.DateTimeField(auto_now=True)

@property
def is_overdue(self):
    """Check if task is overdue"""
    if self.status in ['completed', 'cancelled']:
        return False
    return timezone.now().date() > self.due_date

def __str__(self):
    return self.title

```

1.5.3 Database Queries Optimization

Best Practices:

```

# Use select_related() for foreign keys (one-to-one, many-to-one)
colleagues = Colleague.objects.select_related('user').all()

# Use prefetch_related() for reverse foreign keys and many-to-many
colleagues = Colleague.objects.prefetch_related('outputs').all()

# Combine for complex queries
outputs = Output.objects.select_related(
    'colleague',
    'colleague_user'
).prefetch_related(
    'internal_assignments',
    'critical_friend_assignments'
).all()

# Annotate for counts
from django.db.models import Count
colleagues = Colleague.objects.annotate(
    output_count=Count('outputs')
).all()

# Filter efficiently
current_colleagues = Colleague.objects.filter(
    employment_status='current'
).select_related('user')

```

1.6 □ URL Configuration

1.6.1 Root URLs (ref_manager/urls.py)

```
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('core.urls')),
    path('accounts/', include('django.contrib.auth.urls')),
]

# Serve media files in development
if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

1.6.2 App URLs (core/urls.py)

```
from django.urls import path
from . import views
from . import views_export # v2.0

app_name = 'core'

urlpatterns = [
    # Dashboard
    path('', views.dashboard, name='dashboard'),

    # Colleagues
    path('colleagues/', views.colleague_list, name='colleague_list'),
    path('colleagues/<int:pk>/', views.colleague_detail, name='colleague_detail'),
    path('colleagues/add/', views.colleague_create, name='colleague_create'),
    path('colleagues/<int:pk>/edit/', views.colleague_update, name='colleague_update'),
    path('colleagues/<int:pk>/delete/', views.colleague_delete, name='colleague_delete')

    # Outputs
    path('outputs/', views.output_list, name='output_list'),
    path('outputs/<int:pk>/', views.output_detail, name='output_detail'),
    path('outputs/add/', views.output_create, name='output_create'),
    path('outputs/<int:pk>/edit/', views.output_update, name='output_update'),
    path('outputs/<int:pk>/delete/', views.output_delete, name='output_delete'),
    path('outputs/import/', views.output_import, name='output_import'), # v2.0

    # Internal Panel (v2.0)
    path('internal-panel/', views.internal_panel_list, name='internal_panel_list'),
    path('internal-panel/<int:pk>/', views.internal_panel_detail, name='internal_panel_detail'),
    path('internal-panel/add/', views.internal_panel_create, name='internal_panel_create')
```

```

path('internal-panel/<int:pk>/edit/', views.internal_panel_update, name='internal-panel-update'),
path('internal-panel/<int:pk>/delete/', views.internal_panel_delete, name='internal-panel-delete'),
path('internal-panel/<int:pk>/assign/<int:output_id>/', views.internal_panel_assign, name='internal-panel-assign'),

# Tasks (v2.0)
path('tasks/', views.task_list, name='task_list'),
path('tasks/<int:pk>/', views.task_detail, name='task_detail'),
path('tasks/create/', views.task_create, name='task_create'),
path('tasks/<int:pk>/edit/', views.task_update, name='task_update'),
path('tasks/<int:pk>/delete/', views.task_delete, name='task_delete'),
path('tasks/<int:pk>/complete/', views.task_complete, name='task_complete'),

# Requests
path('requests/', views.request_list, name='request_list'),
path('requests/<int:pk>/', views.request_detail, name='request_detail'),
path('requests/create/', views.request_create, name='request_create'),
path('requests/<int:pk>/edit/', views.request_update, name='request_update'),
path('requests/<int:pk>/complete/', views.request_complete, name='request_complete'),
path('requests/<int:pk>/delete/', views.request_delete, name='request_delete'),

# Export (v2.0)
path('export/assignments/', views_export.export_assignments_view, name='export-assignments'),
path('export/assignments/excel/', views_export.export_assignments_excel, name='export-assignments-excel'),
path('export/assignments/csv/', views_export.export_assignments_csv, name='export-assignments-csv'),

# Reports
path('reports/', views.report_generate, name='report_generate'),
]

```

1.6.3 URL Pattern Naming Convention

- List views: `{model}_list`
 - Detail views: `{model}_detail`
 - Create views: `{model}_create`
 - Update views: `{model}_update`
 - Delete views: `{model}_delete`
 - Custom actions: `{model}_{action}`
-

1.7 Views and Business Logic

1.7.1 View Structure

REF Manager uses function-based views (FBVs) for clarity and simplicity.

1.7.2 Common View Patterns

1.7.2.1 List View Pattern

```

@login_required
def model_list(request):
    """List all instances with filtering"""

    # Get query parameters
    filter_param = request.GET.get('filter', 'all')
    search_query = request.GET.get('q', '')

    # Base queryset with optimization
    queryset = Model.objects.select_related('foreign_key').all()

    # Apply filters
    if filter_param != 'all':
        queryset = queryset.filter(field=filter_param)

    # Apply search
    if search_query:
        queryset = queryset.filter(
            Q(field1__icontains=search_query) |
            Q(field2__icontains=search_query)
        )

    # Pagination
    paginator = Paginator(queryset, 25)
    page_number = request.GET.get('page')
    page_obj = paginator.get_page(page_number)

    context = {
        'page_obj': page_obj,
        'filter_param': filter_param,
        'search_query': search_query,
    }

    return render(request, 'app/model_list.html', context)

```

1.7.2.2 Create/Update View Pattern

```

@login_required
def model_create(request):
    """Create new instance"""

    if request.method == 'POST':
        form = ModelForm(request.POST, request.FILES)
        if form.is_valid():
            instance = form.save(commit=False)
            instance.created_by = request.user
            instance.save()
            messages.success(request, 'Created successfully!')
            return redirect('app:model_detail', pk=instance.pk)
    else:
        form = ModelForm()

```

```

    return render(request, 'app/model_form.html', {'form': form})

@login_required
def model_update(request, pk):
    """Update existing instance"""

    instance = get_object_or_404(Model, pk=pk)

    if request.method == 'POST':
        form = ModelForm(request.POST, request.FILES, instance=instance)
        if form.is_valid():
            form.save()
            messages.success(request, 'Updated successfully!')
            return redirect('app:model_detail', pk=instance.pk)
    else:
        form = ModelForm(instance=instance)

    return render(request, 'app/model_form.html', {
        'form': form,
        'instance': instance
    })

```

1.7.3 Export Views (v2.0)

1.7.3.1 Excel Export with Links

```

from openpyxl import Workbook
from openpyxl.styles import Font, PatternFill, Alignment, Border, Side
from django.http import HttpResponseRedirect
from io.BytesIO import BytesIO

@login_required
def export_assignments_excel(request):
    """Export assignments to Excel with clickable PDF links"""

    # Get filter parameters
    reviewer_type = request.GET.get('reviewer_type', 'all')

    # Create workbook
    wb = Workbook()
    ws = wb.active
    ws.title = "Review Assignments"

    # Headers
    headers = [
        'Reviewer Name', 'Email', 'Output Title',
        'Author', 'Quality', 'Status', 'Due Date',
        'Priority', 'PDF Link', 'Notes'
    ]

```

```

# Style headers
header_fill = PatternFill(start_color="4472C4", end_color="4472C4", fill_type="solid")
header_font = Font(bold=True, color="FFFFFF")

for col_num, header in enumerate(headers, 1):
    cell = ws.cell(row=1, column=col_num)
    cell.value = header
    cell.fill = header_fill
    cell.font = header_font

# Query data
assignments = get_filtered_assignments(request)

# Add data rows
for row_num, assignment in enumerate(assignments, 2):
    # Color coding based on type
    if assignment.is_internal:
        fill = PatternFill(start_color="D9E1F2", fill_type="solid") # Light blue
    else:
        fill = PatternFill(start_color="FFF2CC", fill_type="solid") # Light yellow

    # Add data
    data = [
        assignment.reviewer_name,
        assignment.reviewer_email,
        assignment.output.title,
        assignment.output.colleague.user.get_full_name(),
        assignment.output.get_quality_rating_display(),
        assignment.get_status_display(),
        assignment.due_date.strftime('%Y-%m-%d') if assignment.due_date else '',
        assignment.get_priority_display(),
        'View PDF',
        assignment.notes
    ]

    for col_num, value in enumerate(data, 1):
        cell = ws.cell(row=row_num, column=col_num)
        cell.value = value
        cell.fill = fill

        # Make PDF link clickable
        if col_num == 9 and assignment.output.pdf_file:
            full_url = request.build_absolute_uri(assignment.output.pdf_file.url)
            cell.hyperlink = full_url
            cell.font = Font(color="0563C1", underline="single")

# Adjust column widths
column_widths = {'A': 20, 'B': 25, 'C': 40, 'D': 20, 'E': 15,
                 'F': 15, 'G': 12, 'H': 12, 'I': 15, 'J': 30}
for col, width in column_widths.items():

```

```

        ws.column_dimensions[col].width = width

    # Freeze header
    ws.freeze_panes = 'A2'

    # Create response
    output_buffer = BytesIO()
    wb.save(output_buffer)
    output_buffer.seek(0)

    # Generate filename
    timestamp = datetime.now().strftime('%Y%m%d_%H%M%S')
    filename = f'review_assignments_{timestamp}.xlsx'

    response = HttpResponse(
        output_buffer.read(),
        content_type='application/vnd.openxmlformats-officedocument.spreadsheetml.sheet'
    )
    response['Content-Disposition'] = f'attachment; filename="{filename}"'

    return response

```

1.8 □ Forms and Validation

1.8.1 Form Classes

Forms use django-crispy-forms for Bootstrap styling.

1.8.1.1 Model Form Example

```

from django import forms
from crispy_forms.helper import FormHelper
from crispy_forms.layout import Layout, Submit, Div, Field
from .models import Colleague

class ColleagueForm(forms.ModelForm):
    """Form for creating/updating colleagues"""

    class Meta:
        model = Colleague
        fields = [
            'title', 'fte', 'unit_of_assessment',
            'employment_status', 'employment_end_date',  # v2.0
            'colleague_category',  # v2.0
            'office', 'phone', 'research_interests'
        ]
        widgets = {
            'employment_end_date': forms.DateInput(attrs={'type': 'date'}),

```

```

'research_interests': forms.Textarea(attrs={'rows': 4}),
}

def __init__(self, *args, **kwargs):
    super().__init__(*args, **kwargs)

    # Crispy forms helper
    self.helper = FormHelper()
    self.helper.form_method = 'post'
    self.helper.add_input(Submit('submit', 'Save'))

    # Custom field attributes
    self.fields['fte'].widget.attrs['step'] = '0.1'
    self.fields['fte'].widget.attrs['min'] = '0.1'
    self.fields['fte'].widget.attrs['max'] = '1.0'

def clean_employment_end_date(self):
    """Validate employment end date"""
    status = self.cleaned_data.get('employment_status')
    end_date = self.cleaned_data.get('employment_end_date')

    if status == 'former' and not end_date:
        raise forms.ValidationError(
            "Employment end date required for former staff"
        )

    if status == 'current' and end_date:
        raise forms.ValidationError(
            "Current staff should not have an end date"
        )

    return end_date

```

1.8.2 Custom Validation

```

def clean(self):
    """Cross-field validation"""
    cleaned_data = super().clean()

    start_date = cleaned_data.get('start_date')
    end_date = cleaned_data.get('end_date')

    if start_date and end_date:
        if end_date < start_date:
            raise forms.ValidationError(
                "End date must be after start date"
            )

    return cleaned_data

```

1.9 □ Templates and Frontend

1.9.1 Template Inheritance

```
base.html                      # Base template
└── registration/
    ├── login.html          # Auth templates
    └── password_reset.html
└── core/
    ├── dashboard.html      # Extends base
    ├── colleague_list.html # Extends base
    └── ...
...
```

1.9.2 Base Template (base.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>{% block title %}REF Manager{% endblock %}</title>

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist

    <!-- Font Awesome -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.4/css/all.min.css">

    <!-- Custom CSS -->
    {% load static %}
    <link rel="stylesheet" href="{% static 'css/style.css' %}">

    {% block extra_css %}{% endblock %}
</head>
<body>
    <!-- Navigation -->
    <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
        <a class="navbar-brand" href="{% url 'core:dashboard' %}">
            <i class="fas fa-graduation-cap"></i> REF Manager
        </a>

        <button class="navbar-toggler" type="button" data-
toggle="collapse" data-target="#navbarNav">
            <span class="navbar-toggler-icon"></span>
        </button>

        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav mr-auto">
```

```

        <li class="nav-item">
    <a class="nav-link" href="{% url 'core:colleague_list' %}">
        <i class="fas fa-users"></i> Colleagues
    </a>
</li>
<li class="nav-item">
    <a class="nav-link" href="{% url 'core:output_list' %}">
        <i class="fas fa-book"></i> Outputs
    </a>
</li>
<!-- v2.0 additions --&gt;
&lt;li class="nav-item"&gt;
    &lt;a class="nav-link" href="{% url 'core:internal_panel_list' %}"&gt;
        &lt;i class="fas fa-user-shield"&gt;&lt;/i&gt; Internal Panel
    &lt;/a&gt;
&lt;/li&gt;
&lt;li class="nav-item"&gt;
    &lt;a class="nav-link" href="{% url 'core:task_list' %}"&gt;
        &lt;i class="fas fa-tasks"&gt;&lt;/i&gt; Tasks
    &lt;/a&gt;
&lt;/li&gt;
&lt;li class="nav-item"&gt;
    &lt;a class="nav-link" href="{% url 'core:export_assignments' %}"&gt;
        &lt;i class="fas fa-download"&gt;&lt;/i&gt; Export
    &lt;/a&gt;
&lt;/li&gt;
&lt;/ul&gt;

&lt;ul class="navbar-nav"&gt;
    {% if user.is_authenticated %}
        &lt;li class="nav-item dropdown"&gt;
            &lt;a class="nav-link dropdown-toggle" href="#" id="userDropdown" data-
toggle="dropdown"&gt;
                &lt;i class="fas fa-user"&gt;&lt;/i&gt; {{ user.username }}
            &lt;/a&gt;
            &lt;div class="dropdown-menu dropdown-menu-right"&gt;
                &lt;a class="dropdown-item" href="{% url 'admin:index' %}"&gt;
                    &lt;i class="fas fa-cog"&gt;&lt;/i&gt; Admin
                &lt;/a&gt;
                &lt;div class="dropdown-divider"&gt;&lt;/div&gt;
                &lt;a class="dropdown-item" href="{% url 'logout' %}"&gt;
                    &lt;i class="fas fa-sign-out-alt"&gt;&lt;/i&gt; Logout
                &lt;/a&gt;
            &lt;/div&gt;
        &lt;/li&gt;
    {% else %}
        &lt;li class="nav-item"&gt;
            &lt;a class="nav-link" href="{% url 'login' %}"&gt;
                &lt;i class="fas fa-sign-in-alt"&gt;&lt;/i&gt; Login
            &lt;/a&gt;
        &lt;/li&gt;
    </pre>

```

```

        {% endif %}
    </ul>
</div>
</nav>


{% if messages %}


{% for message in messages %}
        <div class="alert alert-{{ message.tags }} alert-dismissible fade show" role="alert">
            {{ message }}
            <button type="button" class="close" data-dismiss="alert">
                <span>&times;</span>
            </button>
        </div>
    {% endfor %}
</div>
{% endif %}



{% block content %}{% endblock %}


<footer class="mt-5 py-4 bg-light">
    <div class="container text-center">
        <p class="text-muted mb-0">
            REF Manager v2.0 &copy; 2025 University of York
        </p>
    </div>
</footer>


<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>


<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.bundle.js"></script>


<script src="{% static 'js/script.js' %}"></script>

    {% block extra_js %}{% endblock %}
</body>
</html>


```

1.9.3 Template Tags and Filters

1.9.3.1 Custom Template Filters ([core/templatetags/custom_filters.py](#))

```

from django import template

register = template.Library()

@register.filter
def get_item(dictionary, key):
    """Get item from dictionary"""
    return dictionary.get(key)

@register.filter
def multiply(value, arg):
    """Multiply value by arg"""
    try:
        return float(value) * float(arg)
    except (ValueError, TypeError):
        return ''

@register.filter
def badge_class(status):
    """Return Bootstrap badge class for status"""
    badges = {
        'pending': 'badge-warning',
        'in_progress': 'badge-info',
        'completed': 'badge-success',
        'cancelled': 'badge-secondary',
    }
    return badges.get(status, 'badge-secondary')

```

Usage in templates:

```

{% load custom_filters %}

<span class="badge {{ task.status|badge_class }}">
    {{ task.get_status_display }}
</span>

```

1.10 □ Static Files and Media

1.10.1 Static Files Structure

```

static/
└── css/
    └── style.css          # Custom styles
└── js/
    └── script.js          # Custom JavaScript
└── img/
    └── logo.png           # Application logo

```

1.10.2 Media Files

```
media/
└─ pdfs/                               # Uploaded research papers
    ├─ output_1_paper.pdf
    ├─ output_2_paper.pdf
    └─ ...
```

1.10.3 Static Files Configuration

```
# settings.py

STATIC_URL = '/static/'
STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, 'static'),
]

MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

1.10.4 Collecting Static Files

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

# Production (in deployment script)
python manage.py collectstatic --noinput --clear
```

11 Security Considerations

11.1 Authentication and Authorization

```
# Use login_required decorator
from django.contrib.auth.decorators import login_required

@login_required
def protected_view(request):
    # Only authenticated users can access
    pass

# Permission checks
from django.contrib.auth.decorators import permission_required

@permission_required('core.add_output')
def create_output(request):
```

```
pass
```

1.11.2 CSRF Protection

Django's CSRF protection is enabled by default:

```
<!-- In forms -->
<form method="post">
    {% csrf_token %}
    <!-- form fields -->
</form>
```

1.11.3 SQL Injection Prevention

Django ORM automatically escapes queries:

```
# Safe - parameterized
Colleague.objects.filter(user__username=username)

# Avoid raw SQL unless necessary
# If using raw SQL, always use parameters
Colleague.objects.raw('SELECT * FROM core_colleague WHERE id = %s', [id])
```

1.11.4 XSS Prevention

Django templates auto-escape by default:

```
<!-- Automatically escaped -->
{{ user_input }}

<!-- If you need raw HTML (be careful!) -->
{{ trusted_html|safe }}
```

1.11.5 File Upload Security

```
# Validate file types
def validate_pdf(file):
    if not file.name.endswith('.pdf'):
        raise ValidationError('Only PDF files allowed')

    # Check file size (20MB limit)
    if file.size > 20 * 1024 * 1024:
        raise ValidationError('File too large (max 20MB)')

    return file

# In models
pdf_file = models.FileField(
    upload_to='pdfs/ ',
```

```

        validators=[validate_pdf],
        null=True, blank=True
    )

```

1.11.6 Production Security Settings

```

# settings.py (production)

DEBUG = False
ALLOWED_HOSTS = ['your-domain.com', 'www.your-domain.com']

# HTTPS/Security
SECURE_SSL_REDIRECT = True
SESSION_COOKIE_SECURE = True
CSRF_COOKIE_SECURE = True
SECURE_BROWSER_XSS_FILTER = True
SECURE_CONTENT_TYPE_NOSNIFF = True
X_FRAME_OPTIONS = 'DENY'

# HSTS
SECURE_HSTS_SECONDS = 31536000
SECURE_HSTS_INCLUDE_SUBDOMAINS = True
SECURE_HSTS_PRELOAD = True

```

1.12 □ Testing

1.12.1 Test Structure

```

# core/tests.py

from django.test import TestCase, Client
from django.contrib.auth.models import User
from .models import Colleague, Output

class ColleagueModelTest(TestCase):
    """Test Colleague model"""

    def setUp(self):
        """Set up test data"""
        self.user = User.objects.create_user(
            username='testuser',
            email='test@example.com',
            password='testpass123'
        )
        self.colleague = Colleague.objects.create(
            user=self.user,
            fte=1.0,
            employment_status='current',

```

```

        colleague_category='independent'
    )

def test_colleague_creation(self):
    """Test colleague is created correctly"""
    self.assertEqual(self.colleague.user.username, 'testuser')
    self.assertEqual(self.colleague.fte, 1.0)

def test_required_outputs(self):
    """Test required outputs calculation"""
    self.assertEqual(self.colleague.required_outputs, 2.5)

def test_string_representation(self):
    """Test __str__ method"""
    self.assertEqual(str(self.colleague), self.user.get_full_name())

class OutputModelTest(TestCase):
    """Test Output model"""

    def setUp(self):
        user = User.objects.create_user('testuser', 'test@example.com', 'pass')
        self.colleague = Colleague.objects.create(user=user, fte=1.0)
        self.output = Output.objects.create(
            title='Test Output',
            all_authors='Author 1, Author 2',
            publication_venue='Test Journal',
            publication_type='article',
            publication_date='2025-01-01',
            colleague=self.colleague,
            unit_of_assessment='UoA 27',
            quality_rating=4
        )

    def test_output_creation(self):
        """Test output is created"""
        self.assertEqual(self.output.title, 'Test Output')
        self.assertEqual(self.output.quality_rating, 4)

class DashboardViewTest(TestCase):
    """Test dashboard view"""

    def setUp(self):
        self.client = Client()
        self.user = User.objects.create_user('testuser', 'test@example.com', 'pass')
        self.client.login(username='testuser', password='pass')

    def test_dashboard_loads(self):
        """Test dashboard page loads"""
        response = self.client.get('/')

```

```

        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'core/dashboard.html')

def test_dashboard_requires_login(self):
    """Test dashboard requires authentication"""
    self.client.logout()
    response = self.client.get('/')
    self.assertEqual(response.status_code, 302) # Redirect to login

```

1.12.2 Running Tests

```

# Run all tests
python manage.py test

# Run specific app tests
python manage.py test core

# Run specific test class
python manage.py test core.tests.ColleagueModelTest

# Run with verbosity
python manage.py test --verbosity=2

# Keep test database
python manage.py test --keepdb

# Run with coverage
coverage run --source='.' manage.py test
coverage report
coverage html

```

1.13 ↘ Performance Optimization

1.13.1 Database Optimization

1. Use `select_related()` and `prefetch_related()`

```

# Without optimization - N+1 queries
colleagues = Colleague.objects.all()
for colleague in colleagues:
    print(colleague.user.username) # Extra query each time

# With optimization - 2 queries total
colleagues = Colleague.objects.select_related('user').all()
for colleague in colleagues:
    print(colleague.user.username) # No extra queries

```

2. Use `annotate()` for aggregations

```

from django.db.models import Count

# Count in Python - many queries
colleagues = Colleague.objects.all()
for colleague in colleagues:
    output_count = colleague.outputs.count() # Query per colleague

# Count in database - single query
colleagues = Colleague.objects.annotate(
    output_count=Count('outputs')
).all()

```

3. Use only() and defer()

```

# Load only needed fields
colleagues = Colleague.objects.only('id', 'user__username', 'fte')

# Defer large fields
colleagues = Colleague.objects.defer('research_interests')

```

4. Database indexing

```

# In models.py
class Output(models.Model):
    title = models.CharField(max_length=500, db_index=True)
    quality_rating = models.IntegerField(db_index=True)

    class Meta:
        indexes = [
            models.Index(fields=['publication_date', 'quality_rating']),
            models.Index(fields=['colleague', 'quality_rating']),
        ]

```

1.13.2 Caching

```

from django.core.cache import cache
from django.views.decorators.cache import cache_page

# Cache view for 15 minutes
@cache_page(60 * 15)
def report_view(request):
    # Expensive computation
    return render(request, 'report.html', context)

# Manual caching
def get_quality_statistics():
    stats = cache.get('quality_stats')
    if stats is None:
        stats = compute_expensive_statistics()
        cache.set('quality_stats', stats, 60 * 60) # 1 hour
    return stats

```

1.13.3 Query Optimization Tips

```
# Bad - loads all into memory
all_outputs = list(Output.objects.all())

# Good - use iterator for large datasets
for output in Output.objects.iterator(chunk_size=100):
    process_output(output)

# Bad - multiple queries
for colleague in Colleague.objects.all():
    if colleague.outputs.filter(quality_rating=4).exists():
        # process

# Good - single query with annotation
colleagues_with_4star = Colleague.objects.annotate(
    has_4star=Count('outputs', filter=Q(outputs__quality_rating=4))
).filter(has_4star__gt=0)
```

1.14 Deployment Guide

See main README.md for complete deployment instructions.

Quick reference:

```
# 1. Set up environment
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt

# 2. Configure settings
cp .env.example .env
# Edit .env with production values

# 3. Database
python manage.py migrate
python manage.py createsuperuser

# 4. Static files
python manage.py collectstatic --noinput

# 5. Run with Gunicorn
gunicorn ref_manager.wsgi:application --bind 0.0.0.0:8000

# 6. Set up Nginx reverse proxy
# (see production deployment section in README)
```

1.15 Development Workflow

1.15.1 Setting Up Development Environment

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

# Create virtual environment
python3 -m venv venv
source venv/bin/activate

# Install dependencies
pip install -r requirements.txt

# Install development dependencies
pip install django-debug-toolbar coverage black flake8

# Set up database
python manage.py migrate

# Create superuser
python manage.py createsuperuser

# Run development server
python manage.py runserver
```

1.15.2 Code Style

Python: - PEP 8 compliance - Use Black for formatting: `black .` - Use flake8 for linting: `flake8`

JavaScript: - ES6+ syntax - Semicolons required - 2-space indentation

HTML: - Django template syntax - 4-space indentation - Semantic HTML5

1.15.3 Git Workflow

```
# Create feature branch
git checkout -b feature/new-feature

# Make changes and commit
git add .
git commit -m "Add new feature"

# Push to remote
git push origin feature/new-feature

# Create pull request on GitHub
# After review and approval, merge to main
```

1.15.4 Adding New Features

1. Create model (if needed)

```
# models.py
class NewModel(models.Model):
    # fields
    pass
```

2. Create migration

```
python manage.py makemigrations
python manage.py migrate
```

3. Create forms

```
# forms.py
class NewModelForm(forms.ModelForm):
    class Meta:
        model = NewModel
        fields = '__all__'
```

4. Create views

```
# views.py
@login_required
def new_model_list(request):
    # implementation
    pass
```

5. Add URLs

```
# urls.py
path('newmodel/', views.new_model_list, name='new_model_list'),
```

6. Create templates

```
<!-- templates/core/newmodel_list.html -->
{% extends 'base.html' %}
{% block content %}
<!-- content -->
{% endblock %}
```

7. Update navigation

```
<!-- base.html -->
- New Feature

```

8. Write tests

```
# tests.py
class NewModelTest(TestCase):
    def test_creation(self):
        # test code
        pass
```

9. Run tests

python manage.py test

10. Update documentation

1.16 □ Additional Resources

Django Documentation: - Official Docs: <https://docs.djangoproject.com/> - Tutorial: <https://docs.djangoproject.com/en/4.2/intro/tutorial01/>

REF Information: - REF 2029: <https://www.ref.ac.uk/>

Python Resources: - PEP 8: <https://pep8.org/> - Python Docs: <https://docs.python.org/3/>

Frontend: - Bootstrap 4: <https://getbootstrap.com/docs/4.6/> - Font Awesome: <https://fontawesome.com/>

Version: 2.0.0

Last Updated: November 3, 2025

Maintained by: George Tsoulas

Institution: Department of Language and Linguistic Science, University of York

For more information, see the complete documentation suite.

Contents

1 REF Manager v2.0 - Troubleshooting Guide	2
1.1 <input type="checkbox"/> Quick Problem Finder	2
1.2 <input type="checkbox"/> Installation Issues	2
1.2.1 Python Version Issues	2
1.2.2 Package Installation Failures	2
1.2.3 Virtual Environment Errors	3
1.3 <input type="checkbox"/> Runtime Errors	4
1.3.1 Server Won't Start	4
1.3.2 Database Errors	4
1.3.3 Permission Denied Errors	5
1.4 <input type="checkbox"/> Feature-Specific Issues	6
1.4.1 CSV Import Issues	6
1.4.2 Excel Export Issues	6
1.4.3 PDF Upload Issues	7
1.4.4 Authentication Issues	8
1.5 ⚡ Performance Issues	8
1.5.1 Slow Loading	8
1.5.2 Slow Database Queries	9
1.6 <input type="checkbox"/> Data Issues	10
1.6.1 Category Field Inconsistencies	10
1.6.2 Missing Dropdowns	11
1.7 <input type="checkbox"/> Systemd Service Issues	11
1.8 <input type="checkbox"/> Template Errors	12
1.9 <input type="checkbox"/> Development Tools	13
1.9.1 Django Shell Issues	13
1.9.2 Missing Admin Styling	13
1.10 <input type="checkbox"/> Getting Additional Help	13
1.10.1 Information to Include When Seeking Help	13
1.10.2 Diagnostic Commands	14
1.11 <input type="checkbox"/> Common Error Messages Index	14

1 REF Manager v2.0 - Troubleshooting Guide

Problem Solving and Solutions

Version: 2.0.0

Last Updated: November 3, 2025

1.1 Quick Problem Finder

Installation Issues - Python version problems - Package installation failures - Virtual environment errors

Runtime Errors - Server won't start - Database errors - Permission denied

Feature-Specific - CSV import failures - Excel export problems - PDF upload issues
- Login problems

Performance - Slow loading - Database queries

1.2 Installation Issues

1.2.1 Python Version Issues

Problem: Wrong Python version or Python 3.13 compatibility errors

Error Messages:

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

Solution: 1. Check Python version: bash python3 --version Should be 3.10 or higher.

2. If you have Python 3.13, the decimal issue is fixed in v2.0. Update to latest code.

3. For decimal errors in required_outputs:

```
# In models.py, change:  
return self.fte * 2.5  
# To:  
return float(self.fte) * 2.5
```

1.2.2 Package Installation Failures

Problem: pip install fails

Error Messages:

ERROR: Could not install packages due to an EnvironmentError

Solutions:**For Python 3.13:**

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

For permission errors:

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

For specific package failures:

```
# Update pip first  
pip install --upgrade pip
```

```
# Install packages one by one to identify problem  
pip install Django==4.2.0  
pip install django-crispy-forms==2.0  
# etc.
```

For PostgreSQL adapter issues:

```
# Install PostgreSQL development files first  
sudo apt-get install libpq-dev python3-dev  
  
# Then install psycopg2  
pip install psycopg2-binary
```

1.2.3 Virtual Environment Errors

Problem: Virtual environment not activating

Solutions:**Linux/macOS:**

```
# Create fresh venv  
rm -rf venv  
python3 -m venv venv  
source venv/bin/activate
```

Windows:

```
# Create fresh venv  
rmdir /s venv  
python -m venv venv  
venv\Scripts\activate
```

If activation still fails:

```
# Check Python venv module is installed  
python3 -m pip install virtualenv
```

```
# Use virtualenv instead  
virtualenv venv  
source venv/bin/activate
```

1.3 □ Runtime Errors

1.3.1 Server Won't Start

Problem: python manage.py runserver fails

Error 1: Port already in use

Error: That port is already in use.

Solution:

```
# Find process using port 8000
lsof -i :8000
```

```
# Kill the process
kill -9 PID
```

```
# Or use different port
python manage.py runserver 8001
```

Error 2: Import errors

ModuleNotFoundError: No module named 'crispy_forms'

Solution:

```
# Ensure virtual environment is activated
source venv/bin/activate
```

```
# Reinstall requirements
pip install -r requirements.txt
```

Error 3: Settings errors

django.core.exceptions.ImproperlyConfigured: The SECRET_KEY setting must not be empty

Solution:

```
# Check .env file exists
ls -la .env
```

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

```
# Add to .env file
echo "SECRET_KEY=generated-key-here" >> .env
```

1.3.2 Database Errors

Problem: Database migrations fail

Error: No such table

django.db.utils.OperationalError: no such table: core_colleague

Solution:

```
# Run migrations
python manage.py migrate

# If that fails, try:
python manage.py migrate --run-syncdb
```

Error: Migration conflicts

```
django.db.migrations.exceptions.InconsistentMigrationHistory
```

Solution:

```
# Show migrations
python manage.py showmigrations

# Fake problematic migration
python manage.py migrate core --fake 0001

# Then run normally
python manage.py migrate
```

Error: PostgreSQL connection refused

```
FATAL: password authentication failed for user "ref_manager_user"
```

Solution:

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

# Start if needed
sudo systemctl start postgresql

# Reset password
sudo -u postgres psql
ALTER USER ref_manager_user WITH PASSWORD 'new_password';
\q

# Update .env file with new password
```

1.3.3 Permission Denied Errors

Problem: Can't access files or database

Error:

```
PermissionError: [Errno 13] Permission denied: '/path/to/file'
```

Solutions:

For database file:

```
# SQLite database
chmod 664 db.sqlite3
chmod 775 .
```

For media files:

```
chmod -R 755 media/
```

For logs:

```
mkdir -p logs  
chmod 775 logs
```

1.4 □ Feature-Specific Issues

1.4.1 CSV Import Issues

Problem: CSV import fails or gives errors

Error 1: Encoding problems

UnicodeDecodeError: 'utf-8' codec can't decode byte

Solution:

```
# Save CSV as UTF-8 in Excel:  
# File → Save As → CSV UTF-8 (Comma delimited) (.csv)
```

```
# Or convert existing file:  
iconv -f ISO-8859-1 -t UTF-8 oldfile.csv > newfile.csv
```

Error 2: Missing required columns

ValidationError: Missing required columns: title, all_authors

Solution: - Download CSV template from import page - Ensure all required columns present: - title - all_authors - publication_venue - publication_type - publication_date - quality_rating - author_email - uoa - ref_eligible

Error 3: Date format errors

ValidationError: Invalid date format

Solution: - Use YYYY-MM-DD format: 2025-01-15 - Not DD/MM/YYYY or MM/DD/YYYY

Error 4: Email not matching colleague

Warning: No colleague found with email: x@example.com

Solution: - Ensure colleague exists in system first - Check email spelling matches exactly - Or add colleague before import

Error 5: Quality rating invalid

ValidationError: Quality rating must be 0, 1, 2, 3, or 4

Solution: - Use numbers only: 0, 1, 2, 3, or 4 - 0 = Unclassified - 4 = 4* (World-Leading)

1.4.2 Excel Export Issues

Problem: Excel export fails or links don't work

Error 1: Export button does nothing

Check browser console for JavaScript errors

Solution:

```
# Reinstall openpyxl  
pip install --force-reinstall openpyxl  
  
# Check version  
pip show openpyxl  
# Should be 3.1.2+
```

Error 2: Links not clickable

Solution: - Ensure PDFs are uploaded for outputs - Check PDF files are accessible - Verify file paths in media directory

Error 3: File download fails

Solution:

```
# Check disk space  
df -h  
  
# Check permissions  
ls -la media/pdfs/  
  
# Check media URL in settings  
python manage.py shell  
>>> from django.conf import settings  
>>> print(settings.MEDIA_ROOT)  
>>> print(settings.MEDIA_URL)
```

1.4.3 PDF Upload Issues

Problem: Can't upload PDF files

Error 1: File too large

RequestDataTooBig: The request's size exceeds the maximum allowed

Solution:

```
# In settings.py, add:  
DATA_UPLOAD_MAX_MEMORY_SIZE = 20971520 # 20MB  
FILE_UPLOAD_MAX_MEMORY_SIZE = 20971520 # 20MB
```

For Nginx:

```
# In nginx config  
client_max_body_size 20M;
```

Error 2: Invalid file type

ValidationError: Only PDF files allowed

Solution: - Ensure file extension is .pdf - Check file is actually PDF (not renamed) - Maximum 20MB per file

Error 3: Upload directory not writable

```
PermissionError: [Errno 13] Permission denied: '/path/to/media/pdfs/'
```

Solution:

```
mkdir -p media/pdfs  
chmod -R 755 media  
chown -R your-username:your-username media
```

1.4.4 Authentication Issues

Problem: Can't log in

Error 1: Forgot password

Solution:

```
# Reset password via command line  
python manage.py changepassword username
```

Error 2: User doesn't exist

Solution:

```
# Create new user  
python manage.py createsuperuser
```

```
# Or create via shell  
python manage.py shell  
>>> from django.contrib.auth.models import User  
>>> User.objects.create_user('username', 'email@example.com', 'password')
```

Error 3: Session expired

Solution: - Just log in again - To extend sessions, in settings.py:

```
SESSION_COOKIE_AGE = 86400 # 24 hours
```

1.5 ↘ Performance Issues

1.5.1 Slow Loading

Problem: Pages load slowly

Diagnosis:

```
# Enable Django Debug Toolbar  
pip install django-debug-toolbar
```

```
# Add to INSTALLED_APPS in settings.py  
INSTALLED_APPS = [  
    # ...  
    'debug_toolbar',  
]
```

```

# Add to MIDDLEWARE
MIDDLEWARE = [
    # ...
    'debug_toolbar.middleware.DebugToolbarMiddleware',
]

# Add to urls.py
import debug_toolbar
urlpatterns = [
    # ...
    path('__debug__/', include(debug_toolbar.urls)),
]

# Set INTERNAL_IPS
INTERNAL_IPS = ['127.0.0.1']

```

Solutions:

1. Database query optimization:

```

# Use select_related for foreign keys
colleagues = Colleague.objects.select_related('user').all()

# Use prefetch_related for reverse relationships
colleagues = Colleague.objects.prefetch_related('outputs').all()

```

2. Add database indexes:

```

python manage.py makemigrations
python manage.py migrate

```

3. Enable caching:

```

# settings.py
CACHES = {
    'default': {
        'BACKEND': 'django.core.cache.backends.locmem.LocMemCache',
        'LOCATION': 'unique-snowflake',
    }
}

```

1.5.2 Slow Database Queries

Problem: Database queries taking too long

Diagnosis:

```

# Enable query logging
# settings.py
LOGGING = {
    'version': 1,
    'handlers': {
        'console': {
            'class': 'logging.StreamHandler',

```

```

        },
    },
    'loggers': {
        'django.db.backends': {
            'handlers': ['console'],
            'level': 'DEBUG',
        },
    },
}

```

Solutions:

1. Reduce number of queries:

```

# Bad - N+1 queries
for colleague in Colleague.objects.all():
    print(colleague.user.username)

# Good - 2 queries
for colleague in Colleague.objects.select_related('user'):
    print(colleague.user.username)

```

2. Use aggregation:

```

from django.db.models import Count

# Bad
for colleague in Colleague.objects.all():
    count = colleague.outputs.count()

# Good
colleagues = Colleague.objects.annotate(
    output_count=Count('outputs')
)

```

3. Switch to PostgreSQL for large datasets

1.6 □ Data Issues

1.6.1 Category Field Inconsistencies

Problem: Categories stored with both hyphens and underscores

Error:

Colleague shows 'non-independent' in database but 'non_independent' in code

Solution:

```

# Run data migration
python manage.py shell

from core.models import Colleague

```

```

# Fix hyphenated entries
colleagues      =     Colleague.objects.filter(colleague_category='non-
independent')
for c in colleagues:
    c.colleague_category = 'non_independent'
    c.save()

print(f"Fixed {colleagues.count()} colleagues")

```

1.6.2 Missing Dropdowns

Problem: Colleague categories or status not showing in forms

Solution:

1. Check model choices are correct:

```

# In models.py
COLLEAGUE_CATEGORY_CHOICES = [
    ('non_independent', 'Non-Independent Researcher'),
    # Ensure underscore, not hyphen
]

```

2. Run migrations:

```

python manage.py makemigrations
python manage.py migrate

```

3. Clear browser cache: - Ctrl+Shift+R (hard refresh) - Or clear all cache

4. Collect static files:

```

python manage.py collectstatic --clear --noinput

```

1.7 System Service Issues

Problem: Systemd service won't start

Error:

```

Job for ref-manager.service failed

```

Diagnosis:

```

# Check service status
sudo systemctl status ref-manager

```

```

# View logs
sudo journalctl -u ref-manager -n 50

```

Common Issues:

1. Wrong username in service file

```
# /etc/systemd/system/ref-manager.service
[Service]
User=CORRECT_USERNAME # Must match your actual username
```

2. Wrong paths

```
WorkingDirectory=/home/CORRECT_USERNAME/ref-project-app/ref-manager
ExecStart=/home/CORRECT_USERNAME/ref-project-app/ref-manager/venv/bin/gunicorn .
```

3. Virtual environment not activated

```
Environment="PATH=/home/USERNAME/ref-project-app/ref-manager/venv/bin"
```

Fix and reload:

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

```
# Reload systemd
sudo systemctl daemon-reload
```

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

```
# Check status
sudo systemctl status ref-manager
```

1.8 □ Template Errors

Problem: Template syntax errors

Error:

```
TemplateSyntaxError: Invalid block tag
```

Common causes:

1. Unbalanced tags:

```
<!-- Bad -->
{% if condition %}
    <div>Content</div>
<!-- Missing {% endif %} -->
```

```
<!-- Good -->
{% if condition %}
    <div>Content</div>
{% endif %}
```

2. Missing template tag load:

```
<!-- Bad -->
{{ value|custom_filter }}
```

```
<!-- Good -->
```

```
{% load custom_filters %}  
{{ value|custom_filter }}
```

3. Wrong quote types:

```
<!-- Bad -->  
href="{% url 'core:view_name" %}"
```

```
<!-- Good -->  
href="{% url 'core:view_name' %}"
```

1.9 □ Development Tools

1.9.1 Django Shell Issues

Problem: Can't access Django shell

Solution:

```
# Standard shell  
python manage.py shell
```

```
# Enhanced shell (install ipython)  
pip install ipython  
python manage.py shell
```

1.9.2 Missing Admin Styling

Problem: Admin panel has no CSS

Solution:

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

```
# Check STATIC_ROOT setting  
python manage.py shell  
>>> from django.conf import settings  
>>> print(settings.STATIC_ROOT)
```

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

1.10 □ Getting Additional Help

1.10.1 Information to Include When Seeking Help

When reporting issues, include:

1. System Information:

```
# Python version  
python3 --version  
  
# Django version  
python manage.py version  
  
# OS information  
uname -a
```

2. Error Message: - Full error traceback - When error occurs - Steps to reproduce

3. Recent Changes: - What was changed before error - New packages installed - Configuration changes

4. Logs:

```
# Application logs  
tail -n 100 logs/django.log  
  
# System service logs  
sudo journalctl -u ref-manager -n 100  
  
# Nginx logs (if applicable)  
sudo tail -n 100 /var/log/nginx/error.log
```

1.10.2 Diagnostic Commands

```
# Check Django installation  
python manage.py check  
  
# Check deployment settings  
python manage.py check --deploy  
  
# Show migrations  
python manage.py showmigrations  
  
# Show database state  
python manage.py dbshell  
.tables # (in SQLite)  
\dt      # (in PostgreSQL)  
  
# Test database connection  
python manage.py shell  
>>> from django.db import connection  
>>> connection.ensure_connection()  
>>> print("Connected!")
```

1.11 □ Common Error Messages Index

Error	Section
TypeError: Decimal and float	Python 3.13
Port already in use	Server won't start
No such table	Database errors
Permission denied	Permission errors
UnicodeDecodeError	CSV import
File too large	PDF upload
Invalid date format	CSV import
TemplateSyntaxError	Template errors
Service failed	Systemd
Missing static files	Missing admin styling

Version: 2.0.0

Last Updated: November 3, 2025

Maintained by: George Tsoulas

For additional support, see complete documentation suite or contact system administrator.

Contents

1 Changelog	1
1.1 [2.0.0] - 2025-11-03	2
1.1.1 Added	2
1.1.2 Changed	5
1.1.3 Fixed	5
1.2 [1.0.0] - 2025-10-21	5
1.2.1 Added	5
1.3 [Unreleased]	7
1.3.1 Planned Features	7
1.4 Version History Summary	8
1.5 Upgrade Notes	8
1.5.1 Upgrading from 1.0.0 to 2.0.0	8
1.6 Breaking Changes	10
1.6.1 Version 2.0.0	10
1.7 Security Updates	10
1.7.1 Version 2.0.0	10
1.7.2 Version 1.0.0	10
1.8 Performance Improvements	10
1.8.1 Version 2.0.0	10
1.8.2 Version 1.0.0	10
1.9 Documentation Updates	11
1.9.1 Version 2.0.0	11
1.9.2 Version 1.0.0	11
1.10 Acknowledgments	11
1.11 Support	11

1 Changelog

All notable changes to REF Manager will be documented in this file.

The format is based on [Keep a Changelog](#), and this project adheres to [Semantic Versioning](#).

1.1 [2.0.0] - 2025-11-03

1.1.1 Added

1.1.1.1 Employment Status Tracking

- Added employment_status field to Colleague model (current/former)
- Added employment_end_date field to track when employment ended
- Filtering in forms to show only current employees in dropdowns
- Badge display for employment status in colleague lists
- “Mark as former” functionality with confirmation page
- Employment status statistics in dashboard
- Historical data preservation for former staff

1.1.1.2 Enhanced Colleague Categories

- Added colleague_category field with 9 distinct types:
 - Independent Researcher
 - **Non-Independent Researcher** (for post-docs) □ KEY FEATURE
 - Post-Doctoral Researcher
 - Research Assistant
 - Academic Staff
 - Support Staff
 - Current Employee
 - Former Employee
 - Co-author (External)
- Category counts displayed on colleague list page
- Filtering by category in colleague views
- Category-specific badges and visual indicators
- Auto-categorization of external co-authors in CSV import

1.1.1.3 Internal Panel System □ MAJOR FEATURE

- New InternalPanelMember model for internal reviewers
- New InternalPanelAssignment model for internal evaluations
- Four panel member roles: Chair, Member, Specialist, External Liaison
- Complete CRUD operations for panel members
- Assignment tracking with four status types (assigned, in_progress, completed, deferred)
- Rating recommendations aligned with REF quality ratings (4, 3, 2, 1)
- Dashboard widget showing internal panel statistics:
 - Total panel members
 - Active assignments
 - Completed reviews
 - Quality rating distribution
 - Workload distribution
- Separate from Critical Friends (external) system
- Internal panel views, forms, and templates
- Expertise area tracking for panel members
- Availability toggle for panel members

1.1.1.4 Task Management System □ MAJOR FEATURE

- New Task model for general task tracking
- Seven task categories:
 - Administrative
 - Submission
 - Review
 - Meeting
 - Documentation
 - Deadline
 - Other
- Four priority levels: Low, Medium, High, Urgent
- Four status types: Pending, In Progress, Completed, Cancelled
- User assignment and creation tracking
- Due date and start date management
- Task dashboard widget showing:
 - Urgent tasks (due within 7 days)
 - Overdue tasks
 - Tasks by status
 - Tasks by priority
- Complete task list and detail views
- Task creation, editing, and completion functionality
- Filter and search capabilities
- Quick task completion from list view

1.1.1.5 CSV Import Functionality □ MAJOR FEATURE

- Bulk import for research outputs
- Bulk import for colleague data
- Upload interface with file validation
- CSV template download
- Comprehensive data validation:
 - Required field checking
 - Date format validation
 - Quality rating validation
 - Email matching to colleagues
- Duplicate detection by title and DOI
- Automatic linking to existing colleagues
- Auto-categorization of external co-authors
- Import summary and error reporting
- Batch processing support

1.1.1.6 Excel Export with Clickable Links □ MAJOR FEATURE

- Export review assignments to Excel
- Clickable hyperlinks to paper PDF files
- Color-coded formatting:
 - Light blue background for Internal Panel assignments
 - Light yellow background for Critical Friends assignments
- Professional Excel formatting:
 - Frozen header row

- Bold headers
- Adjusted column widths
- Border styling
- Comprehensive filtering options:
 - By reviewer type (Internal Panel / Critical Friends)
 - By specific reviewer
 - By output author
 - By assignment status
- Complete assignment information:
 - Reviewer name and email
 - Output title and author
 - Quality rating
 - Review status
 - Due date and priority
 - Direct PDF links
 - Notes
- Automatic filename generation with timestamp
- CSV export alternative also available

1.1.1.7 Request Management Enhancements

- “Mark as completed” functionality with confirmation page
- Completion timestamp tracking
- “Delete request” functionality with warning confirmation
- Visual status indicators with badges
- Enhanced action buttons in list view:
 - View (blue)
 - Edit (yellow)
 - Complete (green) - hidden for already completed requests
 - Delete (red)
- Success messages after completion or deletion
- Conditional button display based on request status

1.1.1.8 Dashboard Improvements

- New Internal Panel Statistics widget showing:
 - Panel member count by role
 - Active assignments count
 - Completed reviews count
 - Average quality rating from internal reviews
 - Workload distribution
- New Task Overview widget showing:
 - Urgent tasks count
 - Overdue tasks count
 - Tasks by status breakdown
 - Tasks by priority breakdown
- Employment status distribution in Staff Summary
- Category breakdown in colleague statistics
- Enhanced Recent Activity section

1.1.2 Changed

- Enhanced colleague model with new status and category fields
- Updated colleague list view with category filters and statistics
- Improved colleague detail view with employment status display
- Updated dashboard to show Internal Panel and Task statistics
- Enhanced colleague forms with category and status dropdowns
- Modified dropdown filtering to respect employment status (current only)
- Updated colleague queryset filtering in assignment forms
- Improved admin interface with new filters for status and category
- Updated URL patterns for new views and features
- Enhanced navigation menu with new sections

1.1.3 Fixed

- **Category field naming issue:** Resolved inconsistency between hyphenated ("non-independent") and underscore ("non_independent") versions in database
 - **Python 3.13 compatibility:** Fixed Decimal * float multiplication error in required_outputs calculation by converting Decimal to float
 - **Template syntax errors:** Fixed unbalanced if/endif blocks in request templates
 - **URL routing errors:** Fixed missing URL patterns for request_detail, request_complete, request_delete
 - **Form field references:** Fixed incorrect field path references in assignment forms (e.g., colleague_user_last_name)
 - **Static file issues:** Resolved Django admin panel styling problems with collectstatic
 - **Dropdown filtering:** Fixed filtering to show only current employees in assignment forms
-

1.2 [1.0.0] - 2025-10-21

1.2.1 Added

1.2.1.1 Initial Release

- Complete REF submission management system
- Django 4.2+ based web application
- SQLite database (development) with PostgreSQL support (production)
- User authentication and authorization
- Multi-user access with permissions

1.2.1.2 Core Features

- **Output Management:**
 - Add, edit, view, delete research outputs

- Multiple publication types (Article, Book, Chapter, etc.)
- Quality rating system (4, 3, 2, 1, Unclassified)
- PDF file upload for publications
- DOI and URL linking
- REF eligibility tracking
- Unit of Assessment assignment
- Keywords and abstracts
- Submission notes

- **Colleague Management:**

- Staff information tracking
- FTE (Full-Time Equivalent) management
- Unit of Assessment assignment
- Required output calculation
- Output count tracking
- Contact information
- Research interests

- **Critical Friends System:**

- External reviewer management
- Institution and expertise tracking
- Assignment to outputs
- Review status tracking
- Due dates and priorities
- Rating and feedback recording
- Availability management
- Completion tracking

- **Request Management:**

- REF-related request tracking
- Multiple request types
- Priority levels
- Status tracking (Pending, In Progress)
- Due dates
- Output linking
- Assignment to colleagues
- Request history

- **Dashboard:**

- Submission statistics
- Quality profile visualization
- Staff summary
- Review progress tracking
- Recent activity feed
- Quick access to common tasks

- **Report Generation:**

- LaTeX template-based reports
- Multiple report types:
 - * Submission Overview
 - * Quality Profile Analysis
 - * Staff Progress Report
 - * Review Status Report
 - * Comprehensive Report (all combined)
- Three output formats:
 - * LaTeX source (.tex)

- * Article format
- * Report format
- * Beamer presentation
- Professional formatting with charts
- Executive summaries
- Detailed analysis sections

1.2.1.3 Administrative Features

- Django admin panel integration
- User management
- Bulk operations
- Data filtering and search
- Advanced querying
- System configuration

1.2.1.4 Technical Features

- Responsive Bootstrap 4 interface
 - AJAX-based interactions
 - Form validation
 - File upload handling
 - PDF viewing
 - DOI linking
 - Search and filter functionality
 - Pagination
 - Crispy forms integration
-

1.3 [Unreleased]

1.3.1 Planned Features

1.3.1.1 High Priority

- Email notification system for:
 - Task reminders
 - Review due dates
 - Request updates
 - System announcements
- Advanced search across all entities
- Export to Word/DOCX format
- API endpoints for external integrations
- Mobile app or responsive mobile views
- Batch operations for outputs and assignments

1.3.1.2 Medium Priority

- Advanced analytics and visualizations

- Customizable dashboard widgets
- Team collaboration features
- Version control for outputs
- Comment system for discussions
- File attachment support for requests
- Integration with institutional repositories
- Impact case study tracking
- Environment statement management

1.3.1.3 Low Priority

- Machine learning for quality prediction
 - Automated reviewer matching
 - Citation tracking integration
 - Grant linkage
 - Conference presentation tracking
 - Teaching exemption calculation
 - Workload modeling
-

1.4 Version History Summary

Version	Date	Description
2.0.0	2025-11-03	Major update with Internal Panel, Tasks, CSV Import, Excel Export, Enhanced Categories
1.0.0	2025-10-21	Initial release with core REF management features

1.5 Upgrade Notes

1.5.1 Upgrading from 1.0.0 to 2.0.0

Database Migrations Required: Yes

Steps:

1. Backup database:

```
cp db.sqlite3 db.sqlite3.backup.$(date +%Y%m%d)
```

2. Pull/download new code:

```
git pull origin main
# or download and extract new version
```

3. Update dependencies:

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

4. Run migrations:

```
python manage.py makemigrations  
python manage.py migrate
```

5. Collect static files:

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

6. Restart server:

```
# If using systemd:  
sudo systemctl restart ref-manager
```

```
# If using development server:  
# Press Ctrl+C and restart  
python manage.py runserver
```

Data Migration Notes:

- **Colleague Categories:** Existing colleagues will have colleague_category set to None. Update manually or via admin panel.
- **Employment Status:** Existing colleagues will default to “current”. Update as needed.
- **No data loss:** All existing data preserved.
- **New tables created:** Task, InternalPanelMember, InternalPanelAssignment.
- **Backward compatible:** Old data works with new system.

Configuration Changes:

No changes to .env required, but you may want to add:

```
# Task Settings (optional)  
DEFAULT_TASK_PRIORITY=medium  
TASK_OVERDUE_THRESHOLD_DAYS=7
```

```
# Excel Export Settings (optional)  
MAX_EXPORT_ROWS=10000
```

Testing After Upgrade:

1. Log in successfully
 2. View dashboard (new widgets should appear)
 3. Create a colleague with category
 4. Create a task
 5. Create an internal panel member
 6. Try CSV import
 7. Try Excel export
 8. Test existing features still work
-

1.6 Breaking Changes

1.6.1 Version 2.0.0

None - v2.0.0 is fully backward compatible with v1.0.0.

All existing functionality preserved. New features are additions only.

1.7 Security Updates

1.7.1 Version 2.0.0

- Updated Django to 4.2+ (latest security patches)
- Enhanced form validation
- Improved file upload security
- CSV upload validation and sanitization

1.7.2 Version 1.0.0

- Initial security implementation
 - Django built-in authentication
 - CSRF protection
 - SQL injection prevention
 - XSS protection
-

1.8 Performance Improvements

1.8.1 Version 2.0.0

- Optimized database queries with select_related() and prefetch_related()
- Improved dashboard loading with query optimization
- Better pagination for large datasets
- Efficient Excel export with streaming
- CSV import batch processing

1.8.2 Version 1.0.0

- Initial performance optimization
 - Database indexing
 - Query optimization
 - Static file caching
-

1.9 Documentation Updates

1.9.1 Version 2.0.0

- Complete documentation rewrite for v2.0
- Updated README with new features
- Enhanced Quick Start Guide
- Comprehensive User Guide covering all v2.0 features
- Updated Technical Documentation
- Enhanced Troubleshooting Guide
- This Changelog

1.9.2 Version 1.0.0

- Initial documentation suite
 - Installation guide
 - User manual
 - Technical reference
-

1.10 Acknowledgments

Version 2.0.0 Contributors: - George Tsoulas - Feature design and testing - Department of Language and Linguistic Science staff - User feedback - Claude (Anthropic) - Development assistance

Version 1.0.0 Contributors: - George Tsoulas - Initial concept and requirements - Department staff - Testing and feedback

1.11 Support

For questions about changes in specific versions: - Email: [support-email] - Documentation: Complete docs in project folder - GitHub Issues: [if using GitHub]

Note: This changelog follows [Keep a Changelog](#) principles and [Semantic Versioning](#).

Version Format: MAJOR.MINOR.PATCH - MAJOR: Incompatible API changes - MINOR: New functionality (backward compatible) - PATCH: Bug fixes (backward compatible)

Last Updated: November 3, 2025

Maintained by: George Tsoulas

Project: REF Manager

Institution: Department of Language and Linguistic Science, University of York

Contents

1 REF Manager v2.0 - Documentation Index	2
1.1 □ Documentation Overview	2
1.1.1 What's Included	2
1.2 □ Documentation Map	3
1.2.1 By User Type	3
1.3 □ Document Descriptions	3
1.3.1 1. REF-MANAGER-README.md	3
1.3.2 2. QUICK-START-GUIDE.md	4
1.3.3 3. USER-GUIDE.md	6
1.3.4 4. CHANGELOG.md	8
1.3.5 5. TROUBLESHOOTING.md	8
1.3.6 6. DOCUMENTATION-INDEX.md	9
1.4 □ Where to Find Information	9
1.4.1 Common Questions → Documentation Location	9
1.5 □ Reading Order by Goal	10
1.5.1 Goal: Get System Running Quickly	10
1.5.2 Goal: Become a Proficient User	10
1.5.3 Goal: Learn v2.0 New Features	10
1.5.4 Goal: Set Up Production System	11
1.5.5 Goal: Troubleshoot Issues	11
1.5.6 Goal: Understand Changes in v2.0	11
1.6 □ What's New in v2.0 - Quick Reference	11
1.6.1 1. Employment Status Tracking	12
1.6.2 2. Enhanced Colleague Categories	12
1.6.3 3. Internal Panel System	12
1.6.4 4. Task Management	12
1.6.5 5. CSV Import	12
1.6.6 6. Excel Export	12
1.6.7 7. Request Enhancements	12
1.6.8 8. Dashboard Updates	13
1.7 □ Documentation Statistics	13
1.8 □ Documentation Maintenance	13
1.8.1 Update Schedule	13

1.8.2 Contributing to Documentation	13
1.9 □ Documentation Support	14
1.10□ File Locations	14
1.11□ Additional Learning Resources	14
1.12□ Documentation Checklist	14
1.12.1For New Users	14
1.12.2For Administrators	15
1.12.3For Upgrading from v1.0	15
1.13□ Documentation Best Practices	15
1.14□ Version Roadmap	16

1 REF Manager v2.0 - Documentation Index

Complete Documentation Suite

Version: 2.0.0

Last Updated: November 3, 2025

Total Documents: 6 comprehensive guides

1.1 □ Documentation Overview

This documentation suite provides complete coverage of the REF Manager v2.0 system, from installation to advanced usage. Whether you're a new user getting started, an experienced user leveraging new features, or an administrator managing the system, you'll find the guidance you need here.

1.1.1 What's Included

1. **Main README** - Complete system documentation (~60 pages)
2. **Quick Start Guide** - Get running in 10 minutes (~15 pages)
3. **User Guide** - Comprehensive feature guide (~50 pages)
4. **Changelog** - Version history and changes (~10 pages)
5. **Troubleshooting** - Problem solving reference (~15 pages)
6. **This Index** - Documentation navigation

Total: ~155 pages of comprehensive documentation

1.2 □ Documentation Map

1.2.1 By User Type

1.2.1.1 □ New Users Start here: 1. Quick Start Guide (10 min read) 2. README - Installation section 3. User Guide - Dashboard Overview

Then explore: - User Guide - Colleague Management - User Guide - Output Management - README - Features section

1.2.1.2 □ Regular Users Daily reference: - User Guide (keep handy) - Quick Start - Quick Commands Reference - README - Features section

For specific tasks: - User Guide - Import/Export □ NEW - User Guide - Task Management □ NEW - User Guide - Internal Panel □ NEW

1.2.1.3 □ Administrators Installation & Setup: - README - Installation - README - Configuration - README - Production Deployment

Maintenance: - README - Maintenance & Backup - Troubleshooting Guide - README - Update Procedures

System Management: - README - Running the Application - Technical Documentation - Troubleshooting Guide

1.2.1.4 □ Developers Development: - Technical Documentation - README - System Architecture - README - File Structure

Contributing: - Changelog - Version History - README - Technology Stack

1.3 □ Document Descriptions

1.3.1 1. REF-MANAGER-README.md

Primary Documentation - Start Here

Length: ~60 pages

Reading Time: 2 hours (comprehensive), 30 min (skim)

Update Frequency: With each release

Contents:

- **Overview**
 - What is REF Manager
 - Key benefits
 - System architecture
- **What's New in v2.0 □**
 - Employment status tracking
 - Enhanced colleague categories
 - Internal Panel system

- Task management
- CSV import & Excel export
- Dashboard improvements
- **Installation**
 - System requirements
 - Quick start (10 steps)
 - Detailed installation
 - Database setup
 - Environment configuration
- **Features**
 - Complete feature list
 - Dashboard widgets
 - Colleague management
 - Output tracking
 - Review systems (Internal & External)
 - Task management ☐ NEW
 - Request handling
 - Import/Export ☐ ENHANCED
 - Report generation
- **Running the Application**
 - Development server
 - Background service (systemd)
 - Using Screen
- **Production Deployment**
 - Gunicorn + Nginx setup
 - SSL with Let's Encrypt
 - Docker deployment (optional)
- **Maintenance & Backup**
 - Backup procedures
 - Restore procedures
 - Log management
 - Update procedures
- **Troubleshooting**
 - Common issues
 - Debug mode
 - Getting help
- **Appendix**
 - Technology stack
 - File structure
 - Common commands

Best for: - Initial system understanding - Installation and setup - Complete feature reference - Production deployment - System administration

1.3.2 2. QUICK-START-GUIDE.md

Get Running in 10 Minutes

Length: ~15 pages

Reading Time: 10-20 minutes

Update Frequency: With major releases

Contents:

- **Prerequisites**
 - System requirements check
 - Quick verification
- **Installation** (6 steps, 10 minutes)
 - Download & setup
 - Virtual environment
 - Dependencies
 - Environment config
 - Database initialization
 - Server start
- **First Steps**
 - Log in
 - Explore dashboard
 - Add first colleague
 - Add first output
 - Try new features □
- **Daily Workflow**
 - Morning routine
 - Common tasks
 - Weekly activities
- **Running as Background Service**
 - Screen method (simple)
 - systemd method (production)
- **Quick Commands Reference**
 - Daily commands
 - Admin tasks
 - Database operations
- **Common Tasks**
 - Export assignments □ NEW
 - Import outputs □ NEW
 - Mark staff as former □ NEW
 - Manage internal panel □ NEW
 - Task management □ NEW
- **Quick Fixes**
 - Login issues
 - CSS not loading
 - Database errors
 - Python 3.13 compatibility
 - Import/export problems
- **What's New in v2.0**
 - Major additions checklist
 - Feature highlights
- **Checklist**
 - Installation verification
 - Initial setup
 - Testing v2.0 features

Best for: - First-time installation - Quick reference - Common tasks - Daily commands - Troubleshooting basics

1.3.3 3. USER-GUIDE.md

Comprehensive Feature Guide

Length: ~50 pages

Reading Time: 2-3 hours (complete), 15 min (specific sections)

Update Frequency: With feature updates

Contents:

1.3.3.1 Introduction

- What's new in v2.0
- Who should use this guide
- Version highlights

1.3.3.2 Core Features (Detailed)

1. Dashboard Overview

- All dashboard widgets explained
- How to interpret statistics
- Best practices

2. Colleague Management

- Adding colleagues
- Understanding categories □ NEW
- Employment status □ NEW
- Former staff handling □ NEW
- Required outputs calculation
- Best practices

3. Research Output Management

- Adding outputs
- Quality ratings explained
- Bulk operations □ NEW
- CSV import □ NEW
- Filtering and searching
- Best practices

4. Critical Friends System

- About Critical Friends
- Adding reviewers
- Assigning outputs
- Export assignments □ NEW
- Managing reviews
- Best practices

5. Internal Panel System □ NEW

- About Internal Panel
- Setting up panel
- Panel member roles
- Assigning reviews
- Recording reviews

- Dashboard widget
- Internal vs External workflow
- Best practices

6. Task Management □ NEW

- About tasks
- Creating tasks
- Categories and priorities explained
- Managing tasks
- Dashboard widget
- Best practices

7. Request Management

- About requests
- Creating requests
- Status management
- Mark as completed □ NEW
- Delete requests □ NEW
- Best practices

8. Data Import/Export □ ENHANCED

- CSV import (detailed)
- Excel export with links □ NEW
- Other export options
- Best practices

9. Report Generation

- Available reports
- Report formats
- Generating reports
- Customization
- Best practices

1.3.3.3 Additional Sections

- **Best Practices**
 - Data quality
 - Workflow efficiency
 - Quality assurance
 - Team coordination
 - Data management
 - Security
- **Frequently Asked Questions**
 - General questions
 - Output management
 - Import/export □
 - Tasks □
 - Reviews
 - Technical
 - Troubleshooting

Best for: - Daily reference - Learning specific features - Understanding workflows - Best practice guidance - FAQ lookup

1.3.4 4. CHANGELOG.md

Version History and Changes

Length: ~10 pages

Reading Time: 15-30 minutes

Update Frequency: With each release

Contents:

- **Version 2.0.0** (Current)
 - All new features detailed
 - Changed functionality
 - Fixed issues
 - Breaking changes (none)
 - Migration notes
- **Version 1.0.0** (Initial Release)
 - Core features
 - Initial implementation
- **Unreleased / Planned**
 - Future features
 - Roadmap
- **Upgrade Notes**
 - Migration procedures
 - Testing checklist
 - Configuration changes
- **Version History Summary**
 - Quick reference table
- **Security Updates**
 - Per version
- **Performance Improvements**
 - Per version

Best for: - Understanding what changed - Upgrade planning - Feature history - Release notes - Migration guidance

1.3.5 5. TROUBLESHOOTING.md

Problem Solving Reference

Length: ~15 pages (separate document, not yet created)

Reading Time: As needed

Update Frequency: Ongoing

Contents (when created):

- Installation issues
- Runtime errors
- Database problems
- Import/export failures
- Performance issues
- Security concerns

- Network problems
- Browser compatibility
- Mobile issues
- Integration problems

Best for: - Solving specific problems - Error message lookup - Performance optimization - Security hardening

Note: For now, see README - Troubleshooting section and Quick Start Guide - Quick Fixes.

1.3.6 6. DOCUMENTATION-INDEX.md

This Document - Navigation Guide

Length: ~5 pages

Reading Time: 5-10 minutes

Update Frequency: With documentation updates

Purpose: - Navigate documentation suite - Find right document for your needs - Understand documentation structure - Reading order recommendations

1.4 □ Where to Find Information

1.4.1 Common Questions → Documentation Location

“How do I install REF Manager?” → Quick Start Guide OR README - Installation

“What’s new in version 2.0?” → README - What’s New OR Changelog

“How do I add a colleague?” → User Guide - Colleague Management

“What do the colleague categories mean?” □ NEW → User Guide - Colleague Management - Understanding Categories

“How do I import many outputs at once?” □ NEW → User Guide - Data Import/Export - CSV Import

“How do I send papers to reviewers?” □ NEW → User Guide - Data Import/Export - Excel Export

“What’s the difference between Internal Panel and Critical Friends?” □ NEW → User Guide - Internal Panel System - About Internal Panel

“How do I track my tasks?” □ NEW → User Guide - Task Management

“How do I run the server in the background?” → Quick Start Guide - Running as Background Service OR README - Running the Application

“The server won’t start, what do I do?” → Quick Start Guide - Quick Fixes OR README - Troubleshooting

“How do I back up my data?” → README - Maintenance & Backup

“What does ‘Python 3.13 compatibility error’ mean?” → Quick Start Guide - Quick Fixes OR Changelog - Fixed

“Can I delete former staff?” → User Guide - FAQ OR User Guide - Colleague Management

“What’s the required number of outputs per person?” → User Guide - Colleague Management - Required Outputs Calculation

“How do I generate a report?” → User Guide - Report Generation

“What changed in the latest version?” → Changelog

“How do I contribute?” → README - Support OR (future) CONTRIBUTING.md

“What technology does it use?” → README - Appendix - Technology Stack

“Where are the files stored?” → README - Appendix - File Structure

1.5 □ Reading Order by Goal

1.5.1 Goal: Get System Running Quickly

1. Quick Start Guide (full read - 10-20 min)
2. README - Overview (5 min)
3. README - What’s New in v2.0 (5 min)
4. User Guide - Dashboard Overview (5 min)

Total Time: 25-40 minutes

1.5.2 Goal: Become a Proficient User

1. Quick Start Guide (full read - 15 min)
2. User Guide - Introduction (10 min)
3. User Guide - Dashboard Overview (10 min)
4. User Guide - Colleague Management (20 min)
5. User Guide - Output Management (20 min)
6. User Guide - Task Management □ NEW (15 min)
7. User Guide - Internal Panel □ NEW (20 min)
8. User Guide - Data Import/Export □ NEW (15 min)
9. User Guide - Best Practices (15 min)

Total Time: ~2.5 hours

1.5.3 Goal: Learn v2.0 New Features

1. README - What’s New in v2.0 (10 min)
2. Changelog - Version 2.0.0 (15 min)

3. User Guide - Understanding Categories (10 min)
4. User Guide - Internal Panel System (20 min)
5. User Guide - Task Management (15 min)
6. User Guide - CSV Import (10 min)
7. User Guide - Excel Export (10 min)
8. Quick Start Guide - Common Tasks (10 min)

Total Time: ~1.5 hours

1.5.4 Goal: Set Up Production System

1. README - System Requirements (5 min)
2. README - Installation (20 min)
3. README - Configuration (15 min)
4. README - Production Deployment (30 min)
5. README - Maintenance & Backup (20 min)
6. Quick Start Guide - Background Service (10 min)

Total Time: ~1.5 hours + setup time

1.5.5 Goal: Troubleshoot Issues

1. Quick Start Guide - Quick Fixes (5 min)
2. README - Troubleshooting (15 min)
3. User Guide - FAQ (10 min)
4. Changelog - Fixed issues (5 min)

Total Time: 35 minutes

1.5.6 Goal: Understand Changes in v2.0

1. Changelog - Version 2.0.0 (full read - 15 min)
2. README - What's New (10 min)
3. README - Features (skim new features - 10 min)
4. User Guide - New v2.0 sections (skim - 20 min)

Total Time: ~1 hour

1.6 □ What's New in v2.0 - Quick Reference

Where to find v2.0 information:

1.6.1 1. Employment Status Tracking

- **README:** What's New → Employment Status Tracking
- **User Guide:** Colleague Management → Employment Status Management
- **Changelog:** Added → Employment Status Tracking

1.6.2 2. Enhanced Colleague Categories

- **README:** What's New → Enhanced Colleague Categories
- **User Guide:** Colleague Management → Understanding Colleague Categories
- **Quick Start:** First Steps → Add Your First Colleague
- **Changelog:** Added → Enhanced Colleague Categories

1.6.3 3. Internal Panel System

- **README:** What's New → Internal Panel System
- **User Guide:** Internal Panel System (complete section)
- **Quick Start:** First Steps → Set Up Internal Panel
- **Changelog:** Added → Internal Panel System

1.6.4 4. Task Management

- **README:** What's New → Task Management System
- **User Guide:** Task Management (complete section)
- **Quick Start:** First Steps → Create a Task
- **Changelog:** Added → Task Management System

1.6.5 5. CSV Import

- **README:** What's New → CSV Import Functionality
- **User Guide:** Data Import/Export → CSV Import
- **Quick Start:** Common Tasks → Try CSV Import
- **Changelog:** Added → CSV Import Functionality

1.6.6 6. Excel Export

- **README:** What's New → Excel Export with Clickable Links
- **User Guide:** Data Import/Export → Excel Export
- **Quick Start:** Common Tasks → Export Assignments
- **Changelog:** Added → Excel Export with Clickable Links

1.6.7 7. Request Enhancements

- **README:** What's New → Request Management Enhancements
- **User Guide:** Request Management → Managing Requests

- **Changelog:** Added → Request Management Enhancements

1.6.8 8. Dashboard Updates

- **README:** What's New → Dashboard Improvements
 - **User Guide:** Dashboard Overview → Dashboard Widgets
 - **Changelog:** Added → Dashboard Improvements
-

1.7 □ Documentation Statistics

Total Pages: ~155 **Total Words:** ~85,000 **Reading Time (complete):** 6-8 hours
Last Updated: November 3, 2025 **Version:** 2.0.0 **Languages:** English **Format:** Markdown (.md) **Maintenance:** Active

Document Breakdown: - README: 60 pages, 33,000 words - User Guide: 50 pages, 28,000 words - Quick Start: 15 pages, 8,000 words - Changelog: 10 pages, 5,500 words - Troubleshooting: 15 pages, 8,000 words (to be created) - Index: 5 pages, 2,500 words

1.8 □ Documentation Maintenance

1.8.1 Update Schedule

Continuous: - Bug fixes and corrections - Clarifications - Small improvements

With Releases: - Feature documentation - Changelog updates - Version numbers - Screenshots (if any)

Quarterly: - Comprehensive review - Reorganization if needed - Best practices updates - FAQ updates

1.8.2 Contributing to Documentation

Found an error? - Email maintainer with correction - Include document name and section - Suggest improvement

Have a suggestion? - Submit via GitHub Issues (if using GitHub) - Email with suggestion - Discuss at team meetings

Want to contribute? - Contact maintainer - Follow documentation style - Submit pull request or send document

1.9 □ Documentation Support

Questions about documentation? - Email: [support-email] - Office: [location] - Hours: [schedule]

Can't find what you need? - Check User Guide - FAQ - Email support - Request documentation update

Documentation feedback: - Very welcome! - Helps improve for everyone - Email or GitHub Issues

1.10 □ File Locations

All documentation files:

```
ref-manager/
└── REF-MANAGER-README.md           ← Main documentation
└── QUICK-START-GUIDE.md            ← Quick start
└── USER-GUIDE.md                  ← User manual
└── CHANGELOG.md                   ← Version history
└── TROUBLESHOOTING.md              ← Problem solving (TBD)
└── DOCUMENTATION-INDEX.md          ← This file
```

Viewing documentation: - Text editor (any .md file) - GitHub (automatic rendering) - Markdown viewer (recommended) - Convert to PDF (using pandoc)

1.11 □ Additional Learning Resources

Video Tutorials (if available): - Getting Started (5 min) - Adding Outputs (3 min) - Managing Reviews (5 min) - v2.0 New Features (10 min)

Training Materials: - User training slides - Quick reference cards - Cheat sheets - Workshop materials

Community: - User group meetings - Discussion forums - Email list - Department collaboration

1.12 □ Documentation Checklist

Use this checklist to ensure you've covered the essentials:

1.12.1 For New Users

- Read Quick Start Guide
- Follow installation steps

- Review What's New in v2.0
- Explore dashboard
- Try adding colleague
- Try adding output
- Try new v2.0 features
- Bookmark User Guide

1.12.2 For Administrators

- Read README completely
- Understand system requirements
- Review production deployment
- Set up backup procedures
- Test disaster recovery
- Review security section
- Configure monitoring
- Train users

1.12.3 For Upgrading from v1.0

- Read Changelog completely
 - Review What's New
 - Check upgrade notes
 - Backup database
 - Follow migration steps
 - Test new features
 - Update workflows
 - Train team on v2.0
-

1.13 Documentation Best Practices

How to use this documentation effectively:

1. **Start with Index:** This document - understand structure
 2. **Skim First:** Quick overview before deep dive
 3. **Bookmark:** Keep frequently used docs handy
 4. **Search:** Use Ctrl+F to find specific topics
 5. **Follow Links:** Documentation is interconnected
 6. **Take Notes:** Personal annotations helpful
 7. **Test Examples:** Try examples as you read
 8. **Ask Questions:** Don't hesitate to seek help
 9. **Provide Feedback:** Help improve documentation
 10. **Share Knowledge:** Help colleagues learn
-

1.14 □ Version Roadmap

Current: v2.0.0 - November 2025 **Next:** v2.1.0 - Planned Q1 2026 (email notifications, advanced search) **Future:** v3.0.0 - Planned 2026 (API, mobile app, advanced analytics)

Documentation will be updated with each release.

Documentation Maintained By: George Tsoulas

Institution: Department of Language and Linguistic Science, University of York

Last Updated: November 3, 2025

Version: 2.0.0

Status: Complete and Current

Thank you for using REF Manager!

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

Happy REF Managing! □

Contents

1 REF Manager Documentation Tools	2
1.1 Overview	2
1.2 What Gets Built	2
1.3 Prerequisites	3
1.3.1 Required Software	3
1.4 Quick Start	3
1.4.1 Basic Usage	3
1.5 Build Process Details	4
1.5.1 What the Script Does	4
1.5.2 Build Options	4
1.6 Output Structure	4
1.7 Customizing the Build	5
1.7.1 Editing the Script	5
1.7.2 Skip Certain Files	5
1.8 Troubleshooting Build Issues	6
1.8.1 Pandoc Not Found	6
1.8.2 XeLaTeX Not Found	6
1.8.3 Font Errors	6
1.8.4 Build Fails on Specific File	6
1.8.5 LaTeX Compilation Errors	7
1.8.6 Combined PDF Fails	7
1.9 Manual PDF Generation	7
1.9.1 Convert Single Markdown File	7
1.9.2 Compile Single LaTeX File	7
1.9.3 Combine Multiple PDFs	7
1.10 Advanced Usage	8
1.10.1 Generate HTML Instead	8
1.10.2 Generate DOCX	8
1.10.3 Generate EPUB	8
1.11 Build Script Reference	8
1.11.1 Command Line Options	8
1.11.2 Exit Codes	8
1.11.3 Environment Variables	9

1.12	Best Practices	9
1.12.1	Before Building	9
1.12.2	After Building	9
1.12.3	For Distribution	9
1.13	Integration with Git	10
1.13.1	Pre-commit Hook	10
1.13.2	GitHub Actions	10
1.14	Scheduled Builds	11
1.14.1	Daily Build with Cron	11
1.14.2	Weekly Summary Email	11
1.15	Tips and Tricks	11
1.16	Support	11
1.17	Summary	12

1 REF Manager Documentation Tools

Building PDF Documentation from Markdown

Version: 2.0.0

Last Updated: November 3, 2025

1.1 Overview

This guide explains how to build PDF documentation from the markdown source files using the provided `build_docs.sh` script.

1.2 What Gets Built

The build script converts these markdown files to PDF:

1. `REF-MANAGER-README.md` - Main documentation
2. `QUICK-START-GUIDE.md` - Installation guide
3. `USER-GUIDE.md` - Feature guide
4. `TECHNICAL-DOCUMENTATION.md` - Developer reference
5. `TROUBLESHOOTING.md` - Problem solving
6. `CHANGELOG.md` - Version history
7. `DOCUMENTATION-INDEX.md` - Navigation guide
8. `README-TOOLS.md` - This file
9. `00-START-HERE.md` - Overview

Plus compiles LaTeX source files: - QUICK-START-GUIDE.tex - LaTeX version

1.3 Prerequisites

1.3.1 Required Software

1. Pandoc - Document converter

```
# Ubuntu/Debian  
sudo apt-get install pandoc  
  
# macOS  
brew install pandoc
```

```
# Check installation  
pandoc --version
```

2. XeLaTeX - LaTeX engine

```
# Ubuntu/Debian (full install recommended)  
sudo apt-get install texlive-xetex texlive-fonts-recommended texlive-latex-extra
```

```
# macOS  
brew install --cask mactex
```

```
# Check installation  
xelatex --version
```

3. PDF Tools (optional, for combined PDF)

```
# Ubuntu/Debian  
sudo apt-get install poppler-utils
```

```
# macOS  
brew install poppler
```

1.4 Quick Start

1.4.1 Basic Usage

```
# Make script executable  
chmod +x build_docs.sh
```

```
# Run the build script  
./build_docs.sh
```

That's it! PDFs will be generated in documentation/pdf/

1.5 Build Process Details

1.5.1 What the Script Does

1. **Checks dependencies** - Verifies pandoc and xelatex are installed
2. **Creates directories** - Makes documentation/pdf/ and documentation/latex/
3. **Converts markdown** - Uses pandoc to convert .md files to PDF
4. **Compiles LaTeX** - Compiles .tex sources with xelatex
5. **Combines PDFs** - Creates single combined PDF (if pdfunite available)
6. **Cleans up** - Removes auxiliary files

1.5.2 Build Options

Pandoc options used:

```
--pdf-engine=xelatex          # Use XeLaTeX for better Unicode support
--toc                          # Include table of contents
--toc-depth=3                  # Three levels in TOC
--number-sections              # Number all sections
-V geometry:margin=2.5cm       # Set margins
-V fontsize=11pt                # Set font size
-V linkcolor:blue              # Blue links
-V papersize:a4                # A4 paper
--highlight-style=tango         # Code syntax highlighting
-V mainfont="DejaVu Sans"        # Main font
-V monofont="DejaVu Sans Mono"   # Monospace font
```

1.6 Output Structure

```
documentation/
└── pdf/                                # Generated PDFs
    ├── REF-MANAGER-README.pdf
    ├── QUICK-START-GUIDE.pdf
    ├── USER-GUIDE.pdf
    ├── TECHNICAL-DOCUMENTATION.pdf
    ├── TROUBLESHOOTING.pdf
    ├── CHANGELOG.pdf
    ├── DOCUMENTATION-INDEX.pdf
    ├── README-TOOLS.pdf
    └── 00-START-HERE.pdf
        └── REF-Manager-Complete-Documentation.pdf  # Combined (optional)

└── latex/                               # LaTeX intermediate files
    └── QUICK-START-GUIDE.tex
```

```
└── README.txt # Info file
```

1.7 Customizing the Build

1.7.1 Editing the Script

The script can be customized by editing `build_docs.sh`:

Add/remove markdown files:

```
# Find this section in the script
MD_FILES=(
    "REF-MANAGER-README.md"
    "QUICK-START-GUIDE.md"
    # Add your files here
    "YOUR-NEW-FILE.md"
)
```

Add/remove LaTeX files:

```
# Find this section
TEX_FILES=(
    "QUICK-START-GUIDE.tex"
    # Add your .tex files here
)
```

Change PDF options:

```
# Modify the pandoc command
pandoc "$md_file" -o "documentation/pdf/$pdf_name" \
    --pdf-engine=xelatex \
    -V geometry:margin=3cm \      # Change margin
    -V fontsize=12pt \           # Change font size
    # Add more options...
```

1.7.2 Skip Certain Files

Comment out files you don't want to build:

```
MD_FILES=(
    "REF-MANAGER-README.md"
    # "TECHNICAL-DOCUMENTATION.md" # Commented out
)
```

1.8 Troubleshooting Build Issues

1.8.1 Pandoc Not Found

Error:

```
x Error: pandoc not found
```

Solution:

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

```
# Verify  
pandoc --version
```

1.8.2 XeLaTeX Not Found

Error:

```
x Error: xelatex not found
```

Solution:

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

```
# Or minimal install  
sudo apt-get install texlive-xetex texlive-fonts-recommended
```

```
# Verify  
xelatex --version
```

1.8.3 Font Errors

Error:

```
! Font \TU/DejaVuSans(0)/m/n/10=DejaVu Sans at 10.0pt not loadable
```

Solution:

```
# Install DejaVu fonts  
sudo apt-get install fonts-dejavu
```

```
# Or change font in script  
-V mainfont="Liberation Sans"
```

1.8.4 Build Fails on Specific File

Check: 1. File exists and is readable 2. File is valid markdown 3. No special characters causing issues 4. Check pandoc can read it: bash pandoc YOUR-FILE.md -o test.pdf

1.8.5 LaTeX Compilation Errors

Error:

```
! LaTeX Error: ...
```

Solution: 1. Check .tex file syntax 2. Ensure all packages are installed: bash
sudo apt-get install texlive-latex-extra 3. Try compiling manually: bash
xelatex QUICK-START-GUIDE.tex

1.8.6 Combined PDF Fails

Error:

```
pdfunite: command not found
```

Solution:

```
# Install poppler-utils
sudo apt-get install poppler-utils
```

1.9 Manual PDF Generation

1.9.1 Convert Single Markdown File

```
pandoc INPUT.md -o OUTPUT.pdf \
--pdf-engine=xelatex \
--toc \
--number-sections \
-V geometry:margin=2.5cm \
-V fontsize=11pt
```

1.9.2 Compile Single LaTeX File

```
xelatex FILE.tex
xelatex FILE.tex # Run twice for TOC
```

1.9.3 Combine Multiple PDFs

```
pdfunite file1.pdf file2.pdf file3.pdf output.pdf
```

1.10 Advanced Usage

1.10.1 Generate HTML Instead

```
# Modify script or run manually
pandoc INPUT.md -o OUTPUT.html \
--toc \
--standalone \
--css=style.css
```

1.10.2 Generate DOCX

```
pandoc INPUT.md -o OUTPUT.docx \
--toc \
--reference-doc=template.docx
```

1.10.3 Generate EPUB

```
pandoc INPUT.md -o OUTPUT.epub \
--toc \
--epub-cover-image=cover.jpg
```

1.11 Build Script Reference

1.11.1 Command Line Options

The script takes no arguments currently, but you can modify it to accept options:

```
# Example modifications you could add:
```

```
# Clean only (no build)
./build_docs.sh --clean

# Build specific file
./build_docs.sh --file USER-GUIDE.md

# Verbose output
./build_docs.sh --verbose

# Custom output directory
./build_docs.sh --output /path/to/output
```

1.11.2 Exit Codes

- **0:** Success
- **1:** Dependency missing or build error

1.11.3 Environment Variables

You can set these before running:

```
# Use different LaTeX engine  
export PDF_ENGINE=pdflatex  
.build_docs.sh  
  
# Change output directory  
export DOC_OUTPUT_DIR=/custom/path  
.build_docs.sh
```

1.12 Best Practices

1.12.1 Before Building

1. **Update all markdown files** with latest changes
2. **Check markdown syntax** - use a markdown linter
3. **Test manually** - convert one file first to test
4. **Check disk space** - PDFs can be large

1.12.2 After Building

1. **Verify PDFs** - open each to check formatting
2. **Check page numbers** - ensure TOC is correct
3. **Test links** - click internal links
4. **Review combined PDF** - check page order

1.12.3 For Distribution

1. **Compress PDFs** if needed:

```
gs -sDEVICE=pdfwrite -dCompatibilityLevel=1.4 \  
-dPDFSETTINGS=/ebook -dNOPAUSE -dQUIET -dBATCH \  
-sOutputFile=output-compressed.pdf input.pdf
```

2. **Create ZIP archive:**

```
cd documentation  
zip -r REF-Manager-Documentation.zip pdf/
```

3. **Upload to documentation site** or share folder
-

1.13 Integration with Git

1.13.1 Pre-commit Hook

Create .git/hooks/pre-commit:

```
#!/bin/bash
# Auto-build docs on commit

echo "Building documentation..."
./build_docs.sh

# Add PDFs to commit
git add documentation/pdf/*.pdf

echo "Documentation built and staged"
```

1.13.2 GitHub Actions

Create .github/workflows/build-docs.yml:

```
name: Build Documentation

on:
  push:
    paths:
      - '**.md'
      - 'build_docs.sh'

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: Install dependencies
        run: |
          sudo apt-get update
          sudo apt-get install -y pandoc texlive-xetex
      - name: Build documentation
        run: ./build_docs.sh
      - name: Upload PDFs
        uses: actions/upload-artifact@v2
        with:
          name: documentation
          path: documentation/pdf/
```

1.14 Scheduled Builds

1.14.1 Daily Build with Cron

```
# Edit crontab  
crontab -e  
  
# Add line for daily 2 AM build  
0 2 * * * cd /path/to/ref-manager && ./build_docs.sh >> /path/to/build.log 2>&1
```

1.14.2 Weekly Summary Email

```
#!/bin/bash  
# weekly-doc-build.sh  
  
cd /path/to/ref-manager  
./build_docs.sh  
  
# Email results  
mail -s "Weekly Documentation Build" admin@example.com < build.log
```

1.15 Tips and Tricks

1. **Preview before building:** Use a markdown preview tool to check formatting

2. **Keep builds fast:** Comment out files you're not changing

3. **Version PDFs:**

```
cp documentation/pdf/USER-GUIDE.pdf \  
documentation/archive/USER-GUIDE-v2.0.pdf
```

4. **Watermark drafts:** Add draft watermark to PDFs in development

5. **Check file sizes:**

```
ls -lh documentation/pdf/
```

Large files may indicate embedded images that could be optimized

1.16 Support

Script issues? - Check dependencies are installed - Read error messages carefully
- Try building one file manually - Check file permissions

PDF quality issues? - Adjust pandoc options - Try different font - Check source markdown - Review LaTeX output

Still stuck? - Check TROUBLESHOOTING.md - Contact system administrator - Submit issue on GitHub (if using)

1.17 Summary

To build all documentation:

`./build_docs.sh`

Output location:

`documentation/pdf/`

Requirements: - pandoc - xelatex (texlive) - poppler-utils (optional)

Build time: ~2-5 minutes depending on system

Version: 2.0.0

Last Updated: November 3, 2025

Script Author: George Tsoulas

For complete documentation, see the generated PDFs in `documentation/pdf/`.