UPG Management System - Deployment & Handover Documentation

System Overview

Application: Ultra Poor Graduation (UPG) Management Information System **Version:** 1.0 **Framework:** Django 6.0.2 **Purpose:** Track and manage beneficiaries, business groups, savings groups, training, grants, and mentoring activities for the UPG program.

Technical Stack

Backend

Component	Version	Purpose
Python	3.12.3	Runtime
Django	6.0.2	Web Framework
MySQL	8.0.45	Database
Gunicorn	25.0.1	WSGI Server
Nginx	1.24.0	Reverse Proxy

Frontend

Component	Purpose
Bootstrap 5	CSS Framework
Font Awesome	Icons
Chart.js	Data Visualization
jQuery	JavaScript utilities

Key Python Packages

```
Django==6.0.2
django-crispy-forms==2.5
crispy-bootstrap5==2025.6
mysqlclient==2.2.7
pandas==3.0.0
openpyxl==3.1.5
pillow==12.1.0
reportlab==4.4.9
gunicorn==25.0.1
python-decouple==3.8
```

Server Requirements (Production)

Minimum Requirements

Resource	Staging (Current)	Production (Recommended)
CPU	2 vCPU	4 vCPU
RAM	4 GB	8 GB
Storage	20 GB	50 GB SSD
OS	Ubuntu 24.04 LTS	Ubuntu 24.04 LTS

Recommended AWS Instance Types

- **Staging:** t3.small or t3.medium
- **Production:** t3.large or t3.xlarge
- **Database:** RDS MySQL (db.t3.medium) for production

Application Architecture

Django Apps

```
upg_system/
├── accounts/          # User authentication & roles
├── core/              # Core models (Village, County, BM Cycles,
ESR Import)
├── dashboard/         # Role-based dashboards
├── households/        # Beneficiary management
├── business_groups/   # Business group management
├── savings_groups/    # BSG and savings tracking
├── training/          # Training sessions & mentoring
├── upg_grants/        # Grant applications & disbursements
├── programs/          # Program management
├── forms/             # KoBoToolbox integration
├── settings_module/   # System settings & audit logs
└── upg_system/        # Project configuration
```

User Roles

Role	Access Level
ICT Admin	Full system access
Program Manager	Program-wide access, edit savings
M&E Staff	Monitoring & reporting
Field Associate	Supervises mentors, village-level access
Mentor	Assigned households only
County Executive	Dashboard & reports (read-only)
Beneficiary	Limited self-service

Deployment Steps

1. Server Setup

```
# Update system
sudo apt update && sudo apt upgrade -y

# Install dependencies
sudo apt install -y python3.12 python3.12-venv python3.12-dev \
    mysql-server mysql-client libmysqlclient-dev \
    nginx git build-essential

# Create application user
sudo useradd -m -s /bin/bash upg_user
```

2. Database Setup

```
# Secure MySQL
sudo mysql_secure_installation

# Create database and user
sudo mysql -e "CREATE DATABASE upg_production CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci;"
sudo mysql -e "CREATE USER 'upg_user'@'localhost' IDENTIFIED BY
'SECURE_PASSWORD';"
sudo mysql -e "GRANT ALL PRIVILEGES ON upg_production.* TO
'upg_user'@'localhost';"
sudo mysql -e "FLUSH PRIVILEGES;"
```

3. Application Setup

```
# Clone repository
cd /var/www
sudo git clone <repository_url> upg_system
cd upg_system

# Create virtual environment
python3.12 -m venv venv
source venv/bin/activate

# Install dependencies
pip install -r requirements.txt

# Create .env file
cat > .env << EOF
DEBUG=False
SECRET_KEY=<generate-secure-key>
DATABASE_NAME=upg_production
DATABASE_USER=upg_user
DATABASE_PASSWORD=SECURE_PASSWORD
DATABASE_HOST=localhost
DATABASE_PORT=3306
ALLOWED_HOSTS=your-domain.com,www.your-domain.com
EOF

# Run migrations
python manage.py migrate

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

```
# Create superuser
python manage.py createsuperuser
```

4. Gunicorn Configuration

```
# /etc/systemd/system/gunicorn.service
[Unit]
Description=gunicorn daemon for UPG System
After=network.target

[Service]
User=www-data
Group=www-data
WorkingDirectory=/var/www/upg_system
ExecStart=/var/www/upg_system/venv/bin/gunicorn \
    --workers 3 \
    --bind unix:/var/www/upg_system/gunicorn.sock \
    upg_system.wsgi:application

[Install]
WantedBy=multi-user.target

sudo systemctl enable gunicorn
sudo systemctl start gunicorn
```

5. Nginx Configuration

```
# /etc/nginx/sites-available/upg_system
server {
    listen 80;
    server_name your-domain.com www.your-domain.com;

    location /static/ {
        alias /var/www/upg_system/static/;
    }

    location /media/ {
        alias /var/www/upg_system/media/;
    }

    location / {
        include proxy_params;
        proxy_pass http://unix:/var/www/upg_system/gunicorn.sock;
    }
}

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

6. SSL Certificate (Let's Encrypt)

```
sudo apt install certbot python3-certbot-nginx
sudo certbot --nginx -d your-domain.com -d www.your-domain.com
```

Environment Variables (.env)

Variable	Description	Example
DEBUG	Debug mode (False for production)	False
SECRET_KEY	Django secret key	
DATABASE_NAME	MySQL database name	upg_production
DATABASE_USER	MySQL username	upg_user
DATABASE_PASSWORD	MySQL password	
DATABASE_HOST	Database host	localhost
DATABASE_PORT	Database port	3306
ALLOWED_HOSTS	Comma-separated domains	domain.com,www.domain.cor

Backup & Recovery

Database Backup

```
# Daily backup script
mysqldump -u upg_user -p upg_production | gzip > /backups/upg_$(date
+%Y%m%d).sql.gz

# Automated backup (cron)
0 2 * * * /var/www/upg_system/scripts/backup.sh
```

Restore Database

```
gunzip < backup.sql.gz | mysql -u upg_user -p upg_production
```

Media Files Backup

```
tar -czf media_backup.tar.gz /var/www/upg_system/media/
```

Monitoring & Maintenance

Log Files

Log	Location
Django/Gunicorn	`journalctl -u gunicorn`
Nginx Access	`/var/log/nginx/access.log`
Nginx Error	`/var/log/nginx/error.log`

MySQL	`/var/log/mysql/error.log`

Health Checks

```
# Check services
sudo systemctl status gunicorn nginx mysql

# Check disk space
df -h

# Check memory
free -h

# Test application
curl -I http://localhost/
```

Restart Services

```
sudo systemctl restart gunicorn
sudo systemctl restart nginx
sudo systemctl restart mysql
```

Security Checklist

- ☐ DEBUG=False in production
- ☐ Strong SECRET_KEY (50+ characters)
- ☐ HTTPS enabled with valid SSL certificate
- ☐ Database password is secure
- ☐ Firewall configured (UFW)
- ☐ Regular security updates applied
- ☐ Database backups automated
- ☐ Admin URL changed from default
- ☐ Rate limiting configured
- ☐ CSRF protection enabled (default in Django)

Firewall Setup

```
sudo ufw allow 22/tcp    # SSH
sudo ufw allow 80/tcp    # HTTP
sudo ufw allow 443/tcp   # HTTPS
sudo ufw enable
```

Key URLs

URL	Description
`/`	Main dashboard
`/accounts/login/`	User login
`/admin/`	Django admin
`/households/`	Beneficiary management
`/business-groups/`	Business groups
`/savings-groups/`	Savings groups

`/training/`	Training sessions
`/grants/`	Grant management
`/settings/`	System settings

Support Contacts

Role	Contact
Project Manager	Derick Otieno
Lead Developer	Derick Joseph
Document Prepared By	CHASP

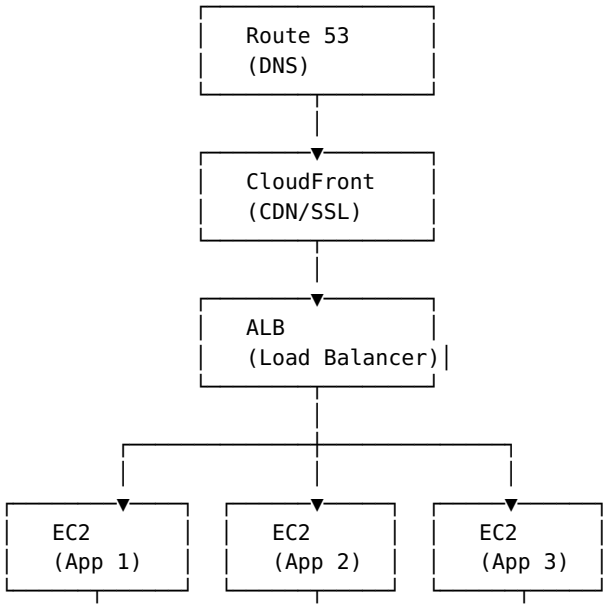
Recommended Hosting Settings

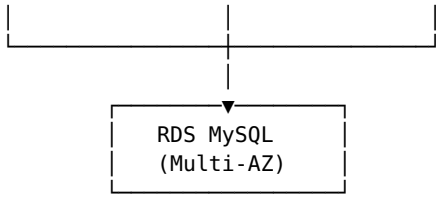
AWS Infrastructure Recommendations

Instance Selection by User Load

Expected Users	Instance Type	vCPU	RAM	Est. Monthly Cost
< 50 concurrent	t3.small	2	2 GB	~$15
50-200 concurrent	t3.medium	2	4 GB	~$30
200-500 concurrent	t3.large	2	8 GB	~$60
500+ concurrent	t3.xlarge	4	16 GB	~$120

Recommended AWS Architecture (Production)





Storage Recommendations

Storage Type	Size	Use Case
Root EBS (gp3)	30 GB	OS & Application
Data EBS (gp3)	50-100 GB	Media uploads
S3 Bucket	Unlimited	Backups, static files

Optimized Gunicorn Configuration

```
# /etc/systemd/system/gunicorn.service
[Unit]
Description=gunicorn daemon for UPG System
Requires=gunicorn.socket
After=network.target

[Service]
User=www-data
Group=www-data
WorkingDirectory=/var/www/upg_system
RuntimeDirectory=gunicorn
Environment="PATH=/var/www/upg_system/venv/bin"

# Production settings
ExecStart=/var/www/upg_system/venv/bin/gunicorn \
    --workers 4 \
    --worker-class gthread \
    --threads 2 \
    --worker-connections 1000 \
    --max-requests 5000 \
    --max-requests-jitter 500 \
    --timeout 120 \
    --graceful-timeout 30 \
    --keep-alive 5 \
    --bind unix:/var/www/upg_system/gunicorn.sock \
    --access-logfile /var/log/gunicorn/access.log \
    --error-logfile /var/log/gunicorn/error.log \
    --capture-output \
    --log-level info \
    upg_system.wsgi:application

Restart=always
RestartSec=3

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

Workers Calculation Formula:

```
workers = (2 × CPU cores) + 1
threads = 2-4 per worker
```

```
Example for 2 vCPU: workers=5, threads=2 (total 10 concurrent)
Example for 4 vCPU: workers=9, threads=2 (total 18 concurrent)
```

Optimized Nginx Configuration

```
# /etc/nginx/nginx.conf - Main settings
user www-data;
worker_processes auto;
pid /run/nginx.pid;
worker_rlimit_nofile 65535;

events {
    worker_connections 4096;
    use epoll;
    multi_accept on;
}

http {
    # Basic Settings
    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 65;
    types_hash_max_size 2048;
    server_tokens off;

    # Buffer Settings
    client_body_buffer_size 10K;
    client_header_buffer_size 1k;
    client_max_body_size 50M;
    large_client_header_buffers 4 32k;

    # Timeouts
    client_body_timeout 12;
    client_header_timeout 12;
    send_timeout 10;

    # Gzip Compression
    gzip on;
    gzip_vary on;
    gzip_proxied any;
    gzip_comp_level 6;
    gzip_types text/plain text/css text/xml application/json
application/javascript
               application/xml application/xml+rss text/javascript
image/svg+xml;
    gzip_min_length 1000;

    # Rate Limiting
    limit_req_zone $binary_remote_addr zone=login:10m rate=5r/s;
    limit_req_zone $binary_remote_addr zone=api:10m rate=30r/s;
    limit_conn_zone $binary_remote_addr zone=addr:10m;

    include /etc/nginx/mime.types;
    include /etc/nginx/conf.d/*.conf;
    include /etc/nginx/sites-enabled/*;
}

# /etc/nginx/sites-available/upg_system - Site configuration
```

```
upstream upg_backend {
    server unix:/var/www/upg_system/gunicorn.sock fail_timeout=0;
    keepalive 32;
}

server {
    listen 80;
    server_name your-domain.com www.your-domain.com;
    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl http2;
    server_name your-domain.com www.your-domain.com;

    # SSL Configuration
    ssl_certificate /etc/letsencrypt/live/your-
domain.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/your-
domain.com/privkey.pem;
    ssl_session_timeout 1d;
    ssl_session_cache shared:SSL:50m;
    ssl_session_tickets off;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-
SHA256;
    ssl_prefer_server_ciphers off;

    # Security Headers
    add_header X-Frame-Options "SAMEORIGIN" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header X-XSS-Protection "1; mode=block" always;
    add_header Strict-Transport-Security "max-age=31536000;
includeSubDomains" always;
    add_header Referrer-Policy "strict-origin-when-cross-origin"
always;

    # Static Files (with caching)
    location /static/ {
        alias /var/www/upg_system/staticfiles/;
        expires 30d;
        add_header Cache-Control "public, immutable";
        access_log off;
    }

    # Media Files
    location /media/ {
        alias /var/www/upg_system/media/;
        expires 7d;
        add_header Cache-Control "public";
    }

    # Rate limit login attempts
    location /accounts/login/ {
        limit_req zone=login burst=5 nodelay;
        limit_conn addr 10;
        include proxy_params;
        proxy_pass http://upg_backend;
    }
```

```
    # Main application
    location / {
        limit_req zone=api burst=20 nodelay;
        include proxy_params;
        proxy_pass http://upg_backend;
        proxy_http_version 1.1;
        proxy_set_header Connection "";
        proxy_connect_timeout 60s;
        proxy_send_timeout 60s;
        proxy_read_timeout 60s;
    }

    # Health check endpoint
    location /health/ {
        access_log off;
        return 200 "healthy\n";
        add_header Content-Type text/plain;
    }
}
```

MySQL/Database Optimization

```
        # /etc/mysql/mysql.conf.d/mysqld.cnf

        [mysqld]
        # General
        user = mysql
        pid-file = /var/run/mysqld/mysqld.pid
        socket = /var/run/mysqld/mysqld.sock
        datadir = /var/lib/mysql
        bind-address = 127.0.0.1

        # Character Set
        character-set-server = utf8mb4
        collation-server = utf8mb4_unicode_ci

        # InnoDB Settings (adjust based on available RAM)
        innodb_buffer_pool_size = 1G          # 50-70% of available RAM
        innodb_buffer_pool_instances = 4
        innodb_log_file_size = 256M
        innodb_log_buffer_size = 16M
        innodb_flush_log_at_trx_commit = 2    # Better performance (1 for
strict durability)
        innodb_flush_method = O_DIRECT
        innodb_file_per_table = 1

        # Connection Settings
        max_connections = 200
        max_connect_errors = 100000
        wait_timeout = 600
        interactive_timeout = 600

        # Query Cache (MySQL 8.0+ uses different approach)
        # Use query result caching at application level instead

        # Logging
        slow_query_log = 1
        slow_query_log_file = /var/log/mysql/slow.log
        long_query_time = 2
```

```
# Temporary Tables
tmp_table_size = 64M
max_heap_table_size = 64M

# Thread Settings
thread_cache_size = 16
table_open_cache = 4000
table_definition_cache = 2000
```

Memory Allocation Guide: | Server RAM | innodb_buffer_pool_size | max_connections | |————|———————-|————-| | 2 GB | 512 MB | 100 | | 4 GB | 1.5 GB | 150 | | 8 GB | 4 GB | 200 | | 16 GB | 10 GB | 300 |

Django Production Settings

Add these to `settings.py` for production:

```
# Production optimizations
if not DEBUG:
    # Database connection pooling
    DATABASES['default']['CONN_MAX_AGE'] = 600  # 10 minutes
    DATABASES['default']['CONN_HEALTH_CHECKS'] = True

    # Caching (Redis recommended)
    CACHES = {
        'default': {
            'BACKEND':
'django.core.cache.backends.redis.RedisCache',
            'LOCATION': 'redis://127.0.0.1:6379/1',
            'OPTIONS': {
                'CLIENT_CLASS': 'django_redis.client.DefaultClient',
            }
        }
    }

    # Session storage in cache
    SESSION_ENGINE = 'django.contrib.sessions.backends.cache'
    SESSION_CACHE_ALIAS = 'default'

    # Logging
    LOGGING = {
        'version': 1,
        'disable_existing_loggers': False,
        'formatters': {
            'verbose': {
                'format': '{levelname} {asctime} {module}
{message}',
                'style': '{',
            },
        },
        'handlers': {
            'file': {
                'level': 'WARNING',
                'class': 'logging.handlers.RotatingFileHandler',
                'filename': '/var/log/upg_system/django.log',
                'maxBytes': 10485760,  # 10MB
                'backupCount': 5,
                'formatter': 'verbose',
            },
```

```
        },
        'root': {
            'handlers': ['file'],
            'level': 'WARNING',
        },
    }
```

Monitoring Setup

CloudWatch Metrics (AWS)

```
# Install CloudWatch agent
wget https://s3.amazonaws.com/amazoncloudwatch-agent/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb
sudo dpkg -i amazon-cloudwatch-agent.deb
```

Key Metrics to Monitor

Metric	Warning Threshold	Critical Threshold
CPU Usage	> 70%	> 90%
Memory Usage	> 75%	> 90%
Disk Usage	> 70%	> 85%
DB Connections	> 80% of max	> 95% of max
Response Time	> 2s	> 5s
Error Rate	> 1%	> 5%

Health Check Script

```
#!/bin/bash
# /var/www/upg_system/scripts/health_check.sh

# Check Gunicorn
if ! systemctl is-active --quiet gunicorn; then
    echo "CRITICAL: Gunicorn is not running"
    sudo systemctl restart gunicorn
fi

# Check Nginx
if ! systemctl is-active --quiet nginx; then
    echo "CRITICAL: Nginx is not running"
    sudo systemctl restart nginx
fi

# Check MySQL
if ! systemctl is-active --quiet mysql; then
    echo "CRITICAL: MySQL is not running"
    sudo systemctl restart mysql
fi

# Check disk space
DISK_USAGE=$(df -h / | awk 'NR==2 {print $5}' | sed 's/%//')
if [ $DISK_USAGE -gt 85 ]; then
    echo "WARNING: Disk usage is ${DISK_USAGE}%"
fi

# Check application response
```

```
        HTTP_CODE=$(curl -s -o /dev/null -w "%{http_code}"
http://localhost/)
        if [ $HTTP_CODE -ne 200 ]; then
            echo "WARNING: Application returned HTTP $HTTP_CODE"
        fi

        echo "Health check completed at $(date)"
```

Backup Strategy

Automated Backup Script

```
        #!/bin/bash
        # /var/www/upg_system/scripts/backup.sh

        BACKUP_DIR="/var/backups/upg_system"
        DATE=$(date +%Y%m%d_%H%M%S)
        RETENTION_DAYS=30

        # Create backup directory
        mkdir -p $BACKUP_DIR

        # Database backup
        mysqldump -u upg_user -p'PASSWORD' upg_production | gzip >
$BACKUP_DIR/db_$DATE.sql.gz

        # Media files backup
        tar -czf $BACKUP_DIR/media_$DATE.tar.gz /var/www/upg_system/media/

        # Upload to S3 (optional)
        aws s3 cp $BACKUP_DIR/db_$DATE.sql.gz s3://your-bucket/backups/
        aws s3 cp $BACKUP_DIR/media_$DATE.tar.gz s3://your-bucket/backups/

        # Clean old backups
        find $BACKUP_DIR -type f -mtime +$RETENTION_DAYS -delete

        echo "Backup completed: $DATE"
```

Cron Schedule

```
        # /etc/cron.d/upg_backup
        # Daily database backup at 2 AM
        0 2 * * * root /var/www/upg_system/scripts/backup.sh >>
/var/log/upg_backup.log 2>&1

        # Weekly full backup on Sunday at 3 AM
        0 3 * * 0 root /var/www/upg_system/scripts/full_backup.sh >>
/var/log/upg_backup.log 2>&1

        # Health check every 5 minutes
        */5 * * * * root /var/www/upg_system/scripts/health_check.sh >>
/var/log/upg_health.log 2>&1
```

Cost Optimization Tips

1. **Use Reserved Instances** - Save 30-60% for long-term workloads
2. **Enable Auto Scaling** - Scale down during off-hours
3. **Use S3 for Static Files** - Cheaper than EBS for static content

4. **Enable CloudFront** - Reduces origin requests and bandwidth
5. **Right-size RDS** - Start small, scale as needed
6. **Use Spot Instances** - For non-critical background tasks

Estimated Monthly Costs (AWS Kenya Region)

Component	Staging	Production
EC2 (t3.medium)	$30	$60 (t3.large)
RDS MySQL	-	$50 (db.t3.small)
EBS Storage (50GB)	$5	$10
S3 (10GB)	$1	$2
Data Transfer	$5	$20
CloudFront	-	$10
Route 53	$1	$1
Total	**~$42**	**~$153**

Support Contacts

Role	Contact
Project Manager	Derick Otieno
Lead Developer	Derick Joseph
Document Prepared By	CHASP

Version History

Version	Date	Changes
1.0	Feb 2026	Initial release

Document generated: February 5, 2026