# COMP 019 - Applications Development and Emerging Technologies

*Full-Stack Python Development*

## SESSION 8: CLOUD DEPLOYMENT

### ACTIVITY 8.1: PREPARING DJANGO FOR PRODUCTION

**Topic:** Production-ready Django configuration

**Description:** Configure Django application for production deployment

**INSTRUCTIONS:**

Production settings:
• DEBUG = False
• ALLOWED_HOSTS configuration
• Secure SECRET_KEY from environment
• Configure STATIC_ROOT and MEDIA_ROOT
Static files handling:
• pip install whitenoise
• Add WhiteNoise middleware
• python manage.py collectstatic
Security settings:
• SECURE_SSL_REDIRECT = True
• SESSION_COOKIE_SECURE = True
• CSRF_COOKIE_SECURE = True
• SECURE_HSTS_SECONDS
Create requirements.txt:
• pip freeze > requirements.txt
• Ensure all dependencies listed
Create Procfile (for some hosts):
• web: gunicorn myproject.wsgi

| | |
|---|---|
| **Deliverables:** Submit production settings.py and requirements.txt | **25 Points** |

### ACTIVITY 8.2: DEPLOYING TO CLOUD PLATFORM

**Topic:** Deploying Django to Railway/Render/Heroku

**Description:** Deploy your complete Django application to the cloud

**INSTRUCTIONS:**

Choose platform (Railway recommended):
• Create account on railway.app
• Connect GitHub repository
• Create new project from repo
Configure environment variables:

• DATABASE_URL (auto-configured with Railway PostgreSQL)
• SECRET_KEY
• DEBUG=False
• ALLOWED_HOSTS
Add PostgreSQL database:
• Add PostgreSQL plugin/service
• Database URL auto-injected
Deploy and verify:
• Trigger deployment
• Check build logs
• Run migrations (railway run python manage.py migrate)
• Create superuser
• Test all functionality

| | |
|---|---|
| **Deliverables:** Submit live deployed URL and working app screenshots | **40 Points** |

## ACTIVITY 8.3: CONNECTING KIVY TO CLOUD API

**Topic:** Mobile app consuming cloud REST API

**Description:** Connect your Kivy mobile app to the deployed Django API

### INSTRUCTIONS:

Install requests library:
• pip install requests
Create API client module:
• Base URL configuration
• Functions for each API endpoint
• Handle authentication token
• Error handling for network issues
Implement in Kivy app:
• Login screen → get token from API
• Store token securely
• Fetch data from API on screen load
• Create/Update/Delete via API calls
• Handle offline scenarios gracefully
Use threading for network calls:
• Don't block UI during API calls
• Use Clock.schedule_once for UI updates
• Show loading indicators
Test complete flow: Kivy → API → PostgreSQL

| | |
|---|---|
| **Deliverables:** Submit Kivy app connected to cloud API with demo | **35 Points** |

**TOTAL POINTS: 100**