Configuration Annotation Summary

Task Completed Successfully

All configuration files in the microservices setup have been annotated with clear instructional comments for credentials, API keys, domain names, and other environment-specific settings.

Statistics

- **Total annotations added**: 254 “CHANGE THIS TO” comments
- **Files annotated**: 19 configuration files
- **Backup files created**: 28 original files backed up to `/tmp/config-backups/`

Breakdown by File Type:

- **Environment files (.env)**: 106 annotations
- **Docker Compose files**: 58 annotations
- **Nginx configuration files**: 16 annotations
- **Shell scripts**: 18 annotations
- **Other config files**: 56 annotations

Files That Were Annotated

Environment Configuration Files

1. `.env` - Main environment file
2. `.env.microservices.example` - Microservices environment template
3. `app/.env` - Main website environment
4. `app/.env.example` - Main website environment template
5. `services/admin-backend/.env.example` - Admin backend configuration
6. `services/blog-backend/.env.example` - Blog backend configuration
7. `services/payment-service/.env.example` - Payment service configuration

Docker Compose Files

1. `docker-compose.microservices.yml` - Production microservices setup
2. `docker-compose.prod.yml` - Production single-container setup
3. `docker-compose.staging.yml` - Staging environment setup
4. `docker-compose.dev.yml` - Development environment setup

Nginx Configuration Files

1. `nginx/microservices.conf` - Microservices reverse proxy
2. `nginx/prod.conf` - Production nginx configuration
3. `nginx/staging.conf` - Staging nginx configuration
4. `nginx/dev.conf` - Development nginx configuration
5. `nginx/common.conf` - Common nginx settings

Deployment and Database Scripts

1. `scripts/deploy-microservices.sh` - Microservices deployment
2. `scripts/deploy.sh` - Main deployment script
3. `scripts/deploy_staging.sh` - Staging deployment
4. `scripts/deploy_dev.sh` - Development deployment
5. `scripts/server_init.sh` - Server initialization
6. `scripts/backup.sh` - Database backup script
7. `scripts/db_migrate.sh` - Database migration script
8. `scripts/rollback.sh` - Rollback script
9. `scripts/init-db.sql` - Database initialization
10. `scripts/init-multiple-databases.sh` - Multi-database setup
11. `scripts/postgres.conf` - PostgreSQL configuration
12. `scripts/redis.conf` - Redis configuration

Cloud Configuration

1. `cloud-init/cloud-init.yaml` - Cloud server initialization

Types of Settings Annotated

1. Database Configuration

• PostgreSQL connection strings and credentials
• Database names and user accounts
• Connection parameters

2. Security Secrets

• JWT secret keys
• Session secrets
• NextAuth secrets
• Internal webhook secrets

3. Payment Integration

• YooKassa Shop ID and Secret Key
• YooKassa Webhook Secret
• Payment service URLs

4. Email Configuration

• SMTP server settings (host, port, security)
• Email credentials and app passwords
• From addresses and sender configuration

5. Domain and URL Configuration

• Main domain names
• Subdomain configurations (admin, blog)
• API endpoints and service URLs
• CORS allowed origins

6. Analytics and Tracking

- Google Analytics 4 Measurement IDs
- Facebook Pixel IDs
- Microsoft Clarity Project IDs

7. SSL and Security

- SSL certificate paths
- Certificate email addresses
- Security headers configuration

8. Redis Configuration

- Redis connection URLs
- Redis authentication passwords
- Cache configuration

9. File Upload Settings

- Maximum file sizes
- Allowed file types
- Upload directories

10. Monitoring and Logging

- Health check intervals
- Monitoring email addresses
- Log levels and retention

Tools Created

1. Configuration Guide (`CONFIGURATION_GUIDE.md`)

Comprehensive guide explaining:
- How to obtain each type of credential
- Step-by-step configuration instructions
- Environment-specific considerations
- Security best practices
- Testing procedures

2. Verification Script (`verify_config.py`)

Automated script that:
- Checks for unchanged default values
- Identifies missing critical configurations
- Validates domain names and URLs
- Provides actionable feedback
- Generates configuration reports

Example Annotations Added

Environment Variables

```
# CHANGE THIS TO: A strong PostgreSQL password (generate with: openssl rand -base64 32)
POSTGRES_PASSWORD=secure_postgres_password_change_this

# CHANGE THIS TO: Your YooKassa Shop ID (get from: https://yookassa.ru/my/shop/integration)
YOOKASSA_SHOP_ID=your_yookassa_shop_id

# CHANGE THIS TO: Your email app password (Gmail: generate at https://myaccount.google.com/apppasswords)
SMTP_PASS=your_app_password
```

Nginx Configuration

```
# CHANGE THIS TO: Your actual domain names
server_name localhost courseplatform.local;

# CHANGE THIS TO: Path to your SSL certificate file
ssl_certificate /etc/nginx/ssl/yourdomain.com.crt;
```

Docker Compose

```
# CHANGE THIS TO: Use strong password from .env file
POSTGRES_PASSWORD: ${POSTGRES_PASSWORD:-postgres123}

# CHANGE THIS TO: Your main website URL (with https://)
- NEXTAUTH_URL=${NEXTAUTH_URL:-http://localhost:3000}
```

Next Steps for Users

1. **Review the Configuration Guide**: Read `CONFIGURATION_GUIDE.md` for detailed instructions
2. **Copy Environment Files**: Copy `.example` files to actual `.env` files
3. **Update All Values**: Replace all values marked with “CHANGE THIS TO” comments
4. **Run Verification**: Use `python3 verify_config.py` to check your configuration
5. **Test Configuration**: Deploy to staging environment first
6. **Deploy to Production**: Once verified, deploy to production

Security Notes

- All original files backed up before modification
- No actual credentials were added (only instructional comments)
- Users must generate their own secure secrets
- Verification script helps ensure no defaults remain
- Guide includes security best practices

Support

If users encounter issues:
1. Check the verification script output
2. Review the configuration guide
3. Examine application logs
4. Verify external service configurations
5. Test individual components

The annotation system provides clear, actionable guidance for every configuration value that needs to be changed, making the setup process much more user-friendly and reducing the chance of configuration errors.