Configuration Guide for Course Platform Microservices

This guide explains how to configure all the credentials, API keys, and environment-specific settings for the course platform microservices architecture.

What Was Changed

All configuration files have been annotated with clear instructional comments using the format:

```
# CHANGE THIS TO: [Description of what to change] ([Where to get the value])
```

Files That Were Updated

Environment Files (.env)

- `.env.microservices.example` - Main microservices environment configuration
- `app/.env.example` - Main website environment variables
- `services/admin-backend/.env.example` - Admin backend service configuration
- `services/blog-backend/.env.example` - Blog backend service configuration
- `services/payment-service/.env.example` - Payment service configuration

Docker Compose Files

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

Nginx Configuration Files

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

Deployment Scripts

- `scripts/deploy-microservices.sh` - Microservices deployment script
- `scripts/deploy.sh` - Main deployment script
- `scripts/deploy_staging.sh` - Staging deployment script
- `scripts/deploy_dev.sh` - Development deployment script
- `scripts/server_init.sh` - Server initialization script
- `scripts/backup.sh` - Database backup script

Database Configuration

- `scripts/init-db.sql` - Database initialization script

- `scripts/postgres.conf` - PostgreSQL configuration
- `scripts/redis.conf` - Redis configuration

Cloud Configuration

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

Required Configurations

1. Database Credentials

```
# Generate a strong PostgreSQL password
openssl rand -base64 32

# Update these in your .env files:
POSTGRES_PASSWORD=your_generated_password_here
POSTGRES_USER=postgres
POSTGRES_DB=course_platform
```

2. Security Secrets

```
# Generate strong secrets for JWT and sessions
openssl rand -base64 64

# Update these in your .env files:
JWT_SECRET=your_generated_jwt_secret_here
SESSION_SECRET=your_generated_session_secret_here
NEXTAUTH_SECRET=your_generated_nextauth_secret_here
INTERNAL_WEBHOOK_SECRET=your_generated_webhook_secret_here
```

3. YooKassa Payment Configuration

1. Sign up at YooKassa (https://yookassa.ru/)
2. Go to Shop Integration Settings (https://yookassa.ru/my/shop/integration)
3. Get your Shop ID and Secret Key
4. Set up webhooks and get the webhook secret

```
YOOKASSA_SHOP_ID=123456
YOOKASSA_SECRET_KEY=live_your_secret_key_here
YOOKASSA_WEBHOOK_SECRET=your_webhook_secret_here
```

4. Email Configuration (SMTP)

For Gmail:

1. Enable 2-factor authentication
2. Generate an App Password at Google Account Settings (https://myaccount.google.com/apppasswords)

```
SMTP_HOST=smtp.gmail.com
SMTP_PORT=587
SMTP_USER=your_email@gmail.com
SMTP_PASS=your_app_password_here
SMTP_FROM=noreply@yourdomain.com
```

For Outlook/Hotmail:

```
SMTP_HOST=smtp-mail.outlook.com
SMTP_PORT=587
SMTP_USER=your_email@outlook.com
SMTP_PASS=your_password_here
```

5. Domain Configuration

Update all domain references to your actual domain:

```
DOMAIN=yourdomain.com
NEXTAUTH_URL=https://yourdomain.com
MAIN_SITE_URL=https://yourdomain.com
ADMIN_FRONTEND_URL=https://admin.yourdomain.com
BLOG_FRONTEND_URL=https://blog.yourdomain.com
```

6. Analytics and Tracking

Google Analytics 4:

1. Create a GA4 property at Google Analytics (https://analytics.google.com/)
2. Get your Measurement ID (format: G-XXXXXXXXXX)

```
GOOGLE_ANALYTICS_ID=G-XXXXXXXXXX
```

Facebook Pixel:

1. Create a pixel at Facebook Events Manager (https://business.facebook.com/events_manager)
2. Get your Pixel ID

```
FACEBOOK_PIXEL_ID=123456789012345
```

Microsoft Clarity:

1. Create a project at Microsoft Clarity (https://clarity.microsoft.com/)
2. Get your Project ID

```
CLARITY_ID=abcdefghij
```

7. SSL Configuration

For Let's Encrypt SSL certificates:

```
SSL_EMAIL=admin@yourdomain.com
```

8. Redis Configuration

For local Redis (default):

```
REDIS_URL=redis://localhost:6379
REDIS_PASSWORD=  # Leave empty for no password
```

For Redis with authentication:

```
REDIS_URL=redis://username:password@host:port
REDIS_PASSWORD=your_redis_password
```

Deployment Steps

1. Copy Environment Files

```
# Copy and customize the main environment file
cp .env.microservices.example .env

# Copy environment files for each service
cp app/.env.example app/.env
cp services/admin-backend/.env.example services/admin-backend/.env
cp services/blog-backend/.env.example services/blog-backend/.env
cp services/payment-service/.env.example services/payment-service/.env
```

2. Update All Configuration Values

Go through each `.env` file and update all values marked with “CHANGE THIS TO” comments.

3. Update Domain Names in Nginx

Update the `server_name` directives in nginx configuration files:
- `nginx/microservices.conf`
- `nginx/prod.conf`
- `nginx/staging.conf`

4. Update SSL Certificate Paths

If using custom SSL certificates, update the paths in nginx configuration files:

```
ssl_certificate /path/to/your/certificate.crt;
ssl_certificate_key /path/to/your/private.key;
```

5. Deploy the Application

```
# For microservices architecture
./scripts/deploy-microservices.sh

# For single-container production
./scripts/deploy.sh

# For staging environment
./scripts/deploy_staging.sh
```

Security Checklist

- [ ] All default passwords changed
- [ ] Strong secrets generated for JWT and sessions
- [ ] YooKassa credentials configured

- [ ] SMTP credentials configured
- [ ] Domain names updated
- [ ] SSL certificates configured
- [ ] CORS origins properly set
- [ ] Database access restricted
- [ ] Redis password set (if needed)
- [ ] Monitoring email configured

Testing Your Configuration

1. Test Database Connection

```
# Test PostgreSQL connection
docker-compose exec postgres psql -U postgres -d course_platform -c "SELECT version();"
```

2. Test Redis Connection

```
# Test Redis connection
docker-compose exec redis redis-cli ping
```

3. Test Email Configuration

Check the application logs for successful email sending or use the admin panel to send a test email.

4. Test Payment Integration

Use YooKassa test credentials to verify payment processing works correctly.

5. Test SSL Certificates

```
# Check SSL certificate
openssl s_client -connect yourdomain.com:443 -servername yourdomain.com
```

Support

If you encounter issues:

1. Check the application logs: `docker-compose logs [service-name]`
2. Verify all environment variables are set correctly
3. Ensure all external services (YooKassa, SMTP) are properly configured
4. Check firewall and network settings

Backup Information

Original configuration files have been backed up to `/tmp/config-backups/` before modification.

To restore original files if needed:

```
# Restore a specific file
cp /tmp/config-backups/filename.backup ./path/to/filename

# List all backup files
ls -la /tmp/config-backups/
```

Environment-Specific Notes

Development

- Use `localhost` for domain names
- Use `http://` instead of `https://`
- YooKassa test credentials are acceptable
- Email testing can use services like MailHog

Staging

- Use staging subdomain (e.g., `staging.yourdomain.com`)
- Use staging credentials for external services
- Test SSL certificates

Production

- Use production domain names
- Use production credentials for all services
- Ensure SSL certificates are valid
- Set up monitoring and alerting
- Configure regular backups

Remember to never commit actual credentials to version control. Always use environment variables and keep your `.env` files secure!