PostgreSQL Database Configuration Fix Summary

Issues Identified

1. **SQLite fallback in production**: The application was falling back to SQLite instead of using PostgreSQL in Azure
2. **Missing Stripe module**: The error log showed "ModuleNotFoundError: No module named 'stripe'" but this was already resolved in requirements.txt

Changes Made

1. Enhanced Database Configuration (`app/config.py`)

- **Added comprehensive environment variable detection** for PostgreSQL connection:
  - `DATABASE_URL` (standard format)
  - `APPSETTING_DATABASE_URL` (Azure App Service format)
  - `AZURE_POSTGRESQL_CONNECTIONSTRING` (Azure-specific)
  - `POSTGRESQL_URL` and `POSTGRES_URL` (alternative formats)
  - Individual components: `POSTGRES_HOST`, `POSTGRES_USER`, `POSTGRES_PASSWORD`, `POSTGRES_DB`, `POSTGRES_PORT`
- **Added PostgreSQL URL construction** from individual environment variables if full connection string isn't available
- **Added production validation** - raises error if DATABASE_URL cannot be determined in non-development environments
- **Enhanced error reporting** - shows available database-related environment variables for debugging

2. Enhanced Database Engine Creation (`app/db/base.py`)

- **Added detailed logging** showing database type being used
- **Added environment variable debugging** for production troubleshooting
- **Added warnings** when SQLite is used in production scenarios

3. Created Test Script (`test_db_config.py`)

- Tests Stripe module import (✅ confirmed working)
- Tests database URL construction under various scenarios
- Shows current environment variables for debugging
- Provides guidance for Azure deployment configuration

Azure Deployment Requirements

For the application to use PostgreSQL in Azure, **at least one** of these environment variables must be set:

Option 1: Full Connection String

```
DATABASE_URL=postgresql://username:password@hostname:5432/database?
sslmode=require
# OR
APPSETTING_DATABASE_URL=postgresql://username:password@hostname:5432/dat
abase?sslmode=require
```

Option 2: Individual Components

```
POSTGRES_HOST=your-azure-postgres-host.postgres.database.azure.com
POSTGRES_USER=your-username
POSTGRES_PASSWORD=your-password
POSTGRES_DB=your-database-name
POSTGRES_PORT=5432
```

Testing Results

✅ **Stripe module**: Successfully imports (resolves the original error)
✅ **PostgreSQL URL construction**: Works correctly when proper environment variables are provided
✅ **Fallback behavior**: Appropriately falls back to SQLite in development only
✅ **Error reporting**: Provides detailed debugging information

Local vs Azure Behavior

- **Local development**: Uses SQLite as configured in `.env` file
- **Azure deployment**: Will use PostgreSQL when proper environment variables are configured
- **Error handling**: Provides clear error messages if PostgreSQL connection cannot be established in production

Next Steps for Azure Deployment

1. **Configure PostgreSQL service** in Azure if not already done
2. **Set environment variables** in Azure App Service configuration:
    - Go to Azure Portal → App Service → Configuration → Application Settings
    - Add one of the database configuration options above
3. **Test deployment** - the enhanced logging will show which database is being used
4. **Monitor logs** - detailed environment variable information is now logged for troubleshooting

Testing Locally

To test PostgreSQL connection locally, temporarily set environment variables:

```
export DATABASE_URL="postgresql://user:pass@localhost:5432/testdb"
python test_db_config.py
```

The application now provides comprehensive database configuration options and detailed error reporting to resolve the SQLite/PostgreSQL deployment issues.