Invoice System Project Documentation

Overview

The Invoice System is an enterprise-ready ASP.NET WebForms application designed to manage invoices, invoice items, and customers. The system features:

- **Invoice Management:** Create, view, update, and delete invoices.
- **Invoice Item Management:** Manage invoice items that reference a master Items table.
- **Customer Management:** Add, update, and delete customer records.
- **Data Access Layer:** All database interactions are handled through repository classes that call stored procedures.
- **UI:** A responsive user interface using a Master Page with Bootstrap.

Architecture

1. **Presentation Layer:**
    - ASP.NET WebForms pages (Default.aspx, InvoiceDetails.aspx, AddInvoice.aspx, Customers.aspx, InvoiceItems.aspx) using a common Master Page (Site.Master) with Bootstrap.
2. **Business/Data Access Layer:**
    - Repository classes (e.g., InvoiceRepository, CustomerRepository, InvoiceItemRepository) encapsulate all CRUD operations by calling stored procedures in SQL Server.
3. **Database Layer:**
    - SQL Server database (InvoiceDB) with normalized tables:
        - **Customers:** Stores customer information.
        - **Invoices:** Stores invoice headers and references Customers.
        - **Items:** Stores master list of items (name and unit price).
        - **InvoiceItems:** Associates invoices with items and stores quantities.

Prerequisites

- **Development Environment:** Visual Studio 2019 or later.
- **Framework:** .NET Framework 4.7.2 (or later recommended).
- **Database:** SQL Server Express or full SQL Server.

- **Tools:** SQL Server Management Studio (SSMS) for database scripts.

Setup Instructions

1. **Project Creation:**
   - Create a new ASP.NET WebForms project in Visual Studio.
   - Add a Master Page (Site.Master) with Bootstrap references for a consistent UI.
   - Add content pages (Default.aspx, InvoiceDetails.aspx, AddInvoice.aspx, Customers.aspx, InvoiceItems.aspx) that reference the Master Page.
2. **Database Setup:**
   - Create a new database (e.g., InvoiceDB) in SQL Server.
   - Run the provided T-SQL scripts (see the "Database Schema" and "Stored Procedures" sections below) to create tables and stored procedures.
   - Use the test data script to populate the tables.
3. **Configure Connection String:**
   - Update the web.config file with your connection string (e.g., using Windows Authentication or SQL authentication).
4. **Data Access Layer:**
   - Add repository classes (InvoiceRepository, CustomerRepository, InvoiceItemRepository) in a DataAccess folder.
   - Ensure these classes call the correct stored procedures for all CRUD operations.
5. **(Optional) Authentication Setup:**
   - Install NuGet packages for ASP.NET Identity and OWIN.
   - Create ApplicationUser, ApplicationDbContext, and ApplicationUserManager classes.
   - Configure OWIN in Startup.cs.
6. **Build and Run:**
   - Build the project in Visual Studio.
   - Run the application locally and verify each feature (invoices, invoice items, and customer management).

Database Schema

Tables

- **Customers:**
  - CustomerID (INT, PK, IDENTITY)
  - Name (NVARCHAR(100))
  - Address (NVARCHAR(255))
  - ContactInfo (NVARCHAR(100))
- **Invoices:**
  - InvoiceID (INT, PK, IDENTITY)
  - CustomerID (INT, FK to Customers)
  - InvoiceDate (DATETIME)
  - TotalAmount (DECIMAL(18,2))
- **Items:**
  - ItemID (INT, PK, IDENTITY)
  - Name (NVARCHAR(255))
  - UnitPrice (DECIMAL(18,2))
- **InvoiceItems:**
  - InvoiceItemID (INT, PK, IDENTITY)
  - InvoiceID (INT, FK to Invoices)
  - ItemID (INT, FK to Items)
  - Quantity (INT)

Note: Foreign key constraints enforce referential integrity. Optionally, you may enable cascade deletes on child records.

Stored Procedures

Below are examples of stored procedures for key operations:

Invoice Stored Procedures

- spGetAllInvoices
- spGetInvoiceById
- spAddInvoice

- spUpdateInvoice
- spDeleteInvoice

InvoiceItems Stored Procedures

- spGetInvoiceItems
- spAddInvoiceItem
- spUpdateInvoiceItem
- spDeleteInvoiceItem

Customer Stored Procedures

- spGetAllCustomers
- spGetCustomerById
- spAddCustomer
- spUpdateCustomer
- spDeleteCustomer

Data Access Layer

The data access layer consists of repository classes that encapsulate all database interactions. Examples include:

- **InvoiceRepository:** Uses stored procedures to get all invoices, get invoice by ID, add, update, and delete invoices.
- **CustomerRepository:** Handles CRUD operations for customers.
- **InvoiceItemRepository:** Manages invoice items using stored procedures for adding, updating, and deleting invoice items.

Each repository uses ADO.NET with SqlConnection, SqlCommand, and CommandType.StoredProcedure to perform operations securely and efficiently.

UI and Code-Behind

- **Default.aspx:** Lists all invoices in a GridView using InvoiceRepository.GetAllInvoices.
- **AddInvoice.aspx:** Allows the addition of new invoices.
- **Customers.aspx:** Enables managing customers with add, update, and delete functionality similar to the InvoiceItems screen.

- **InvoiceItems.aspx:** Provides an interface to add, update, and delete invoice items for a selected invoice.

Each page interacts with the data access layer, ensuring a separation of concerns and maintainable code.

Deployment

1. **Build the Project:**
   - Build the solution in Visual Studio.
2. **Deploy to IIS:**
   - Publish the project to IIS (or your hosting environment) and ensure the web.config connection string is updated appropriately.
3. **Database Deployment:**
   - Run all stored procedure and table creation scripts on the production SQL Server.
4. **Security & Configuration:**
   - Verify firewall settings, security configurations, and authentication mechanisms before going live.

Troubleshooting & Future Enhancements

- **Troubleshooting:**
  - Ensure the database and stored procedures exist before running the application.
  - Check connection string settings if database connectivity issues occur.
  - Verify the cascade delete settings or manually delete child records if foreign key conflicts occur.
- **Future Enhancements:**
  - Improve error handling and logging.
  - Expand ASP.NET Identity for user and role management.
  - Enhance UI and add client-side validation.
  - Consider an ORM (like Entity Framework) for advanced data access.

Conclusion

This documentation outlines the architecture, setup, and components of the Invoice System project. By following the provided instructions and scripts, you can build a robust, enterprise-ready application that cleanly separates the presentation, business, and data access layers using stored procedures, repository classes, and ASP.NET WebForms with Bootstrap.