# Pilot Task – ASP.NET MVC 5 Developer Evaluation

We are inviting shortlisted candidates to complete a technical pilot task to assess your development skills in ASP.NET MVC 5, SQL Server, and n-tier architecture. This will help us evaluate your coding discipline, architecture understanding, and attention to detail before considering long-term engagement.

**Objective:**

Build a functional ticket management module (3–5 pages) in ASP.NET MVC 5 that allows users to submit, view, edit, and list customer service tickets. You will design the backend SQL Server schema, use stored procedures exclusively for data operations, and follow an n-tier architecture. All input must be properly validated, and error handling should be implemented at all levels.

**Required Pages (3–5 total):**

1. Create Ticket Page

- Fields:

- Full Name (required)

- Mobile Number (required)

- Email Address (required, must be valid)

- Issue Type (dropdown: from DB)

- Description (multiline, required)

- Priority (dropdown: Low / Medium / High)

- Created Date & Time (auto-populated server-side)

- Function:

- Submit saves ticket using a stored procedure

- Data validation must occur on both client and server

2. List Tickets Page

- Display tickets in a table/grid with columns:

- Ticket ID, Customer Name, Issue Type, Priority, Status, Created Date

- Features:

- Basic filters for Issue Type and Priority

- Option to view or edit each ticket

3. View Ticket Details Page

- Shows full details of a specific ticket

- Read-only mode

- Includes Created Date in full format (with time)

4. Edit Ticket Page

- Load existing data for editing

- Fields same as in Create Page (except Created Date)

- Allow changing all fields except Ticket ID and Created Date

- Save using stored procedure

- Full validation again required

5. (Optional) Home/Navigation Page

- Simple landing page with links to create, list, and search tickets (optional bonus)

**Database Requirements (SQL Server):**

- Create 2 tables:

- CustomerTickets

- TicketID (Primary Key, Identity)

- FullName

- MobileNumber

- Email

- IssueTypeID (FK)

- Description

- Priority

- Status (default: “Open”)

- CreatedDate (datetime)

- IssueTypes

- IssueTypeID (Primary Key)

- IssueTypeName (e.g., Technical, Billing, Complaint, Other)

- Use Stored Procedures ONLY for:

- Insert, Update, Select (List & Single), and Filter

**Architecture Guidelines:**

Use a proper N-Tier Architecture, structured as:

- Presentation Layer: MVC Controllers and Views

- Business Logic Layer (BLL): Handle input/output, validation logic

- Data Access Layer (DAL): Call stored procedures, return DTOs

**Error Handling Requirements:**

- Implement structured error handling:

- Try-catch in DAL, BLL, and Controllers

- Log or show meaningful messages for database or system errors

- Handle validation issues with clear user messages

**Other Notes:**

- Use proper naming conventions for methods and variables

- Avoid using auto-generated scaffolding – create views and logic manually

- Use ViewModels if needed

- Prioritize code readability and maintainability

**Deliverables:**

1. Zipped Visual Studio solution

2. SQL script (tables, stored procedures, and seed data)

3. README or Word file describing:

- Architecture and components

- How to run the application

- Notes or assumptions

**Estimated Time to Complete:**

4–6 hours

Deadline: Within 2 days of receiving this task