Creating a Custom C# MVC API for Managing Patient Prescriptions with Twilio Integration

In today's healthcare landscape, effective communication and efficient management of patient prescriptions are paramount. In this article, I'll walk you through the process of creating a custom C# MVC API for managing patient prescriptions, while also integrating Twilio for SMS, email, and voice communication. This project not only enhances patient engagement but also streamlines pharmacy operations.

Setting Up the MVC API

1. Project Initialization

Start by creating a new ASP.NET Core MVC project. You can do this using Visual Studio or the .NET CLI.

```
dotnet new mvc -n PatientPrescriptionAPI

cd PatientPrescriptionAPI
```

2. Define the Model

Define a Prescription model that represents the prescription details.

```
public class Prescription
{
  public int patid { get; set; }
  public string PatientName { get; set; }
  public string Medication { get; set; }
  public string Dosage { get; set; }
  public DateTime DatePrescribed { get; set; }
}
```

3. Create the API Controller

Next, create an API controller for handling prescription-related operations.

```
[ApiController]
[Route("api/[controller]")]
public class PrescriptionsController : ControllerBase
{
  private readonly IPrescriptionService _prescriptionService;

  public PrescriptionsController(IPrescriptionService prescriptionService)
  {
    _prescriptionService = prescriptionService;
  }

  [HttpGet("{id}")]
  public ActionResult<Prescription> GetPrescription(int id)
  {
    var prescription = _prescriptionService.GetById(id);
    if (prescription == null) return NotFound();
    return Ok(prescription);
  }

  [HttpPost]
  public ActionResult<Prescription> CreatePrescription([FromBody]
Prescription prescription)
  {
    if (prescription == null) return BadRequest();
    _prescriptionService.Add(prescription);
    return CreatedAtAction(nameof(GetPrescription), new { id =
prescription.Id }, prescription);
  }
}
```

Integrating for Twilio Communication

4. Setting up Twilio

To integrate Twilio, first, install the Twilio NuGet package:

```
dotnet add package Twilio
```

Next, configure your Twilio credentials in appsettings.json. For enhanced security, consider storing the credentials in an encrypted SQL table instead.

```
{
  "Twilio": {
    "AccountSid": "your_account_sid",
    "AuthToken": "your_auth_token",
    "FromNumber": "your_twilio_number"
  }
}
```

5. Sending SMS Notifications

Implement a service to send SMS notifications using Twilio.

```
public class TwilioService
{
  private readonly IConfiguration _configuration;

  public TwilioService(IConfiguration configuration)
  {
    // read from appsettings.json
    _configuration = configuration;
    TwilioClient.Init(
      _configuration["Twilio:AccountSid"],
      _configuration["Twilio:AuthToken"]
    );
  }

  public async Task SendSmsAsync(string to, string message)
  {
    var messageOptions = new CreateMessageOptions(new PhoneNumber(to))
    {
      From = new PhoneNumber(_configuration["Twilio:FromNumber"]),
      Body = message,
    };

    await MessageResource.CreateAsync(messageOptions);
  }
}
```

6. Integrating Email and Voice

For email and voice communications, you can extend the TwilioService to include methods for sending email using the Twilio SendGrid API and the Twilio CallResource API for making voice calls using Twilio's programmable voice capabilities.

Conclusion

By combining a custom C# MVC API with Twilio's communication tools, you can create a robust system for managing patient prescriptions and enhancing patient engagement through SMS, email, and voice notifications. This approach not only streamlines pharmacy operations but also ensures that patients receive timely updates regarding their medications.