



<http://fsd08-twitterclone.azurewebsites.net/>



# TWITTER/X CLONE

Meredith White | Shiyuan Xu | Xiaoxing Pan

# AGENDA

Introduction

User Features

Technology

Database

Challenges & Solutions

Future Work

Summary

# INTRODUCTION

Application Y attempts to clone the core features of the social media platform X, formerly known as Twitter.

X is a microblogging platform where users publish short posts, and interact with other users and posts via likes, follows, comments, retweets, and messages.

X has many advanced features for searches, trends, suggestions, and moderation.



# USER FEATURES

## Non-admin user

- Register, login, logout
- Post, edit, delete a tweet
- View, retweet with or without added content, reply, like
- Follow/unfollow users
- Search for tweets and users
- View user profiles
- Edit profile, upload avatar, reset password
- Send and receive messages
- Report tweets to admin

## Admin

- View reports
- Suspend tweets

# TECHNOLOGIES



- ASP.NET Core (Razor Pages)
- SQL Server
- Entity Framework Core
- View components, Partial Views
- ASP.NET Core Identity
- Azure Blob Storage & hosting
- Web API
- AJAX
- JQuery
- Bootstrap
- Rich text editor: CKEditor



# APPROACH

## Xiaoxing

- Search
- Notifications
- Follows
- Trends
- Views & components

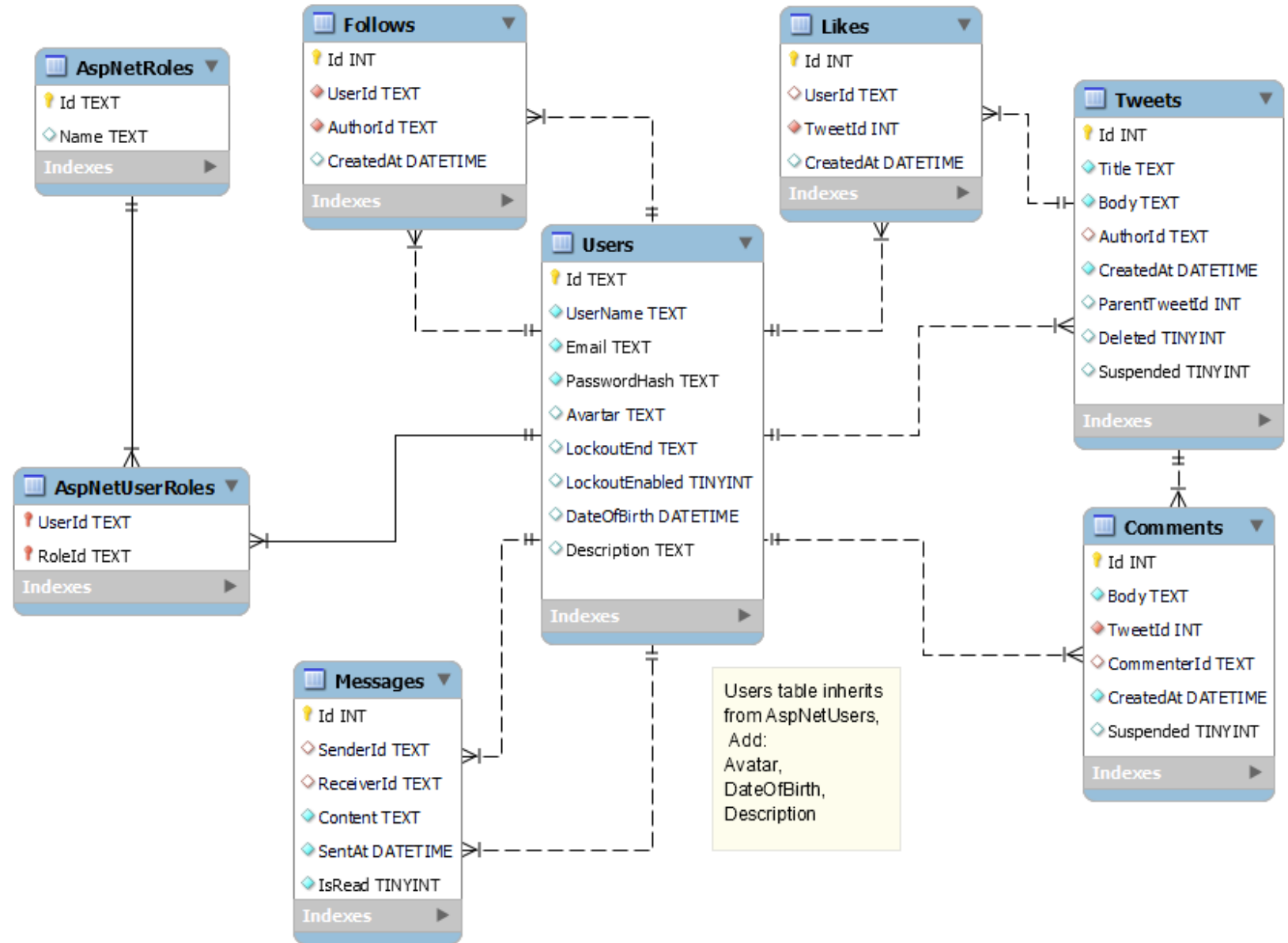
## Shiyuan

- Database & models
- Tweets, retweets, replies
- Likes & follows
- Admin operations
- Reset password

## Meredith

- Login & registration
- User settings
- Messages
- Image uploads
- Azure hosting

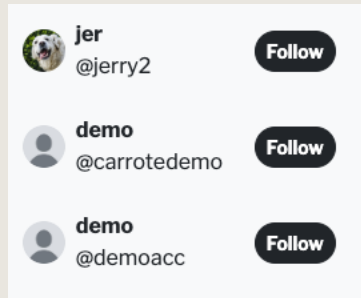
# DATABASE



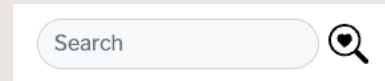
# CHALLENGES & SOLUTIONS

## BREAK DOWN TO SMALL PIECES & MAKE IT REUSABLE

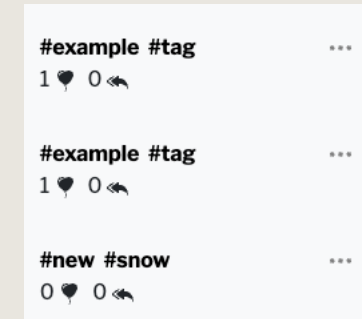
Partial View



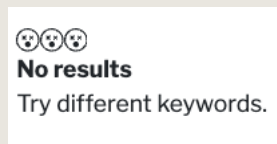
- ❖ Right sidebar
- ❖ Search Result
- ❖ Follow Suggestion



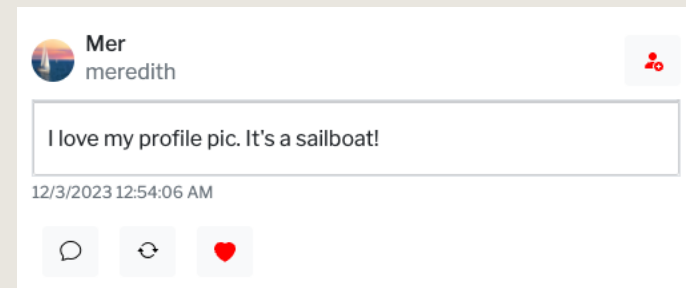
- ❖ Right sidebar
- ❖ Search Result
- ❖ Show more Trend



- ❖ Right sidebar
- ❖ Show more Trend



- ❖ Search user/tweet
- ❖ Search Trend

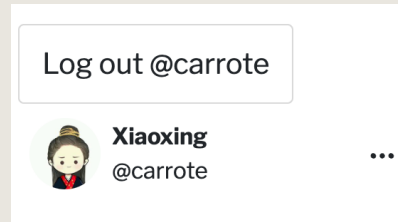


- ❖ !Anywhere a tweet is displayed



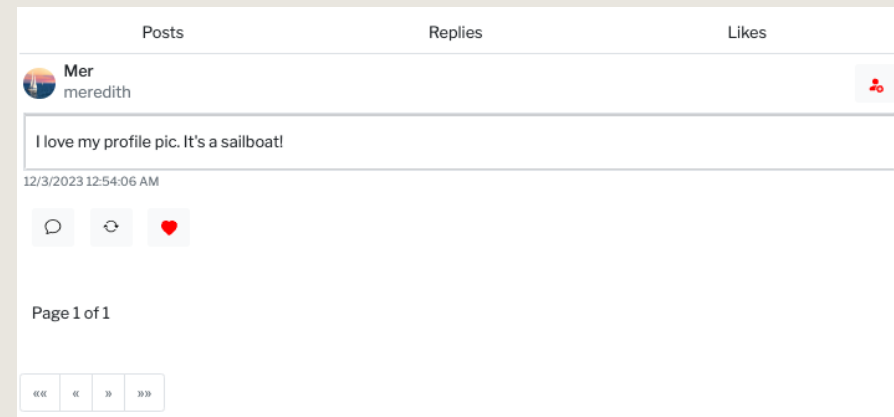
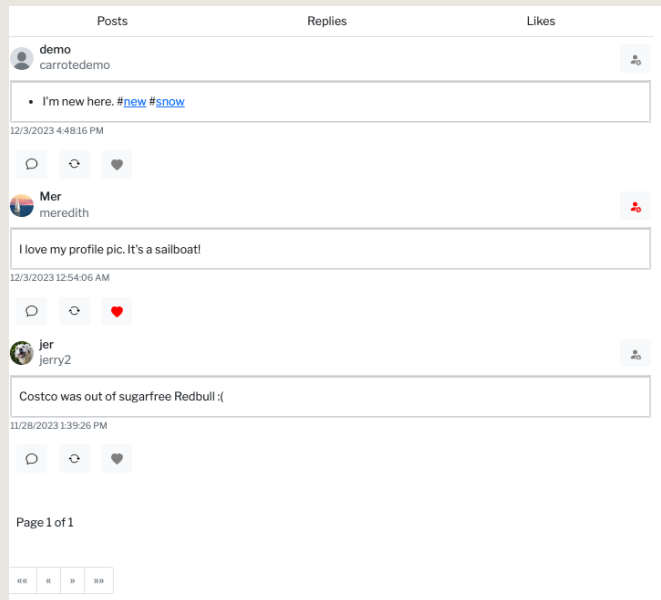
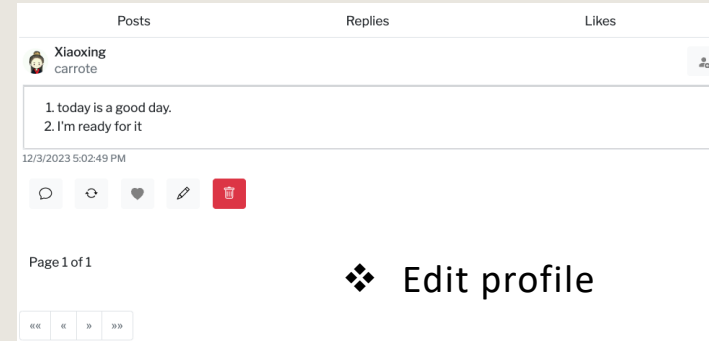
# CHALLENGES & SOLUTIONS

BREAK DOWN TO SMALL PIECES & MAKE IT REUSABLE



❖ Left navbar/\_layout

View Component



# CHALLENGES & SOLUTIONS

- PARTIAL VIEW CONTROL

## Parent Page

```
ViewData["ShowDescription"] = Model.ShowDescription;  
  
public int ShowDescription { get; set; } = 0; // 0: hide, 1: show
```

## Partial View

```
@model List<TwitterClone.Models.User>  
  
@{  
    ViewData["Title"] = "Follow Suggestions";  
    int showDescription = (int)ViewData["ShowDescription"];  
}
```

```
@if (showDescription == 1)  
{  
    <div>@user.Description</div>  
}
```

```
ViewData["IndexDropDown"] = Model.IndexDropDown;
```

```
public int IndexDropDown { get; set; } = 1; // 1:keyword, 2:tag
```

```
@{  
    int indexDropDown = (int)ViewData["IndexDropDown"];  
}
```

```
@if (indexDropDown == 1)  
{  
    <p class="fs-6 p-1 fw-lighter mb-0">Try searching for people, lists,  
    or keywords</p>  
}  
else  
{  
    <p class="fs-6 p-1 fw-lighter mb-0">Try searching for tags</p>  
}
```

# CHALLENGES & SOLUTIONS

## • PARTIAL VIEW CONTROL

Button

```
<div class="ms-auto">
  <button class="btnFollow btn btn-sm rounded rounded-pill
    fw-bold" data-user-id="@user.Id">Follow</button>
</div>
```



JavaScript

```
// Attach click event to follow buttons
$('.btnFollow').each(function () {
  var button = $(this);
  var userId = $(this).data('user-id');

  // Make an asynchronous request to the server to get follow status
  fetch(`/api/Follow/getFollowStatus?userId=${userId}`, {
    method: 'GET',
    headers: {
      'Content-Type': 'application/json',
    }
  })
})
```



Controller

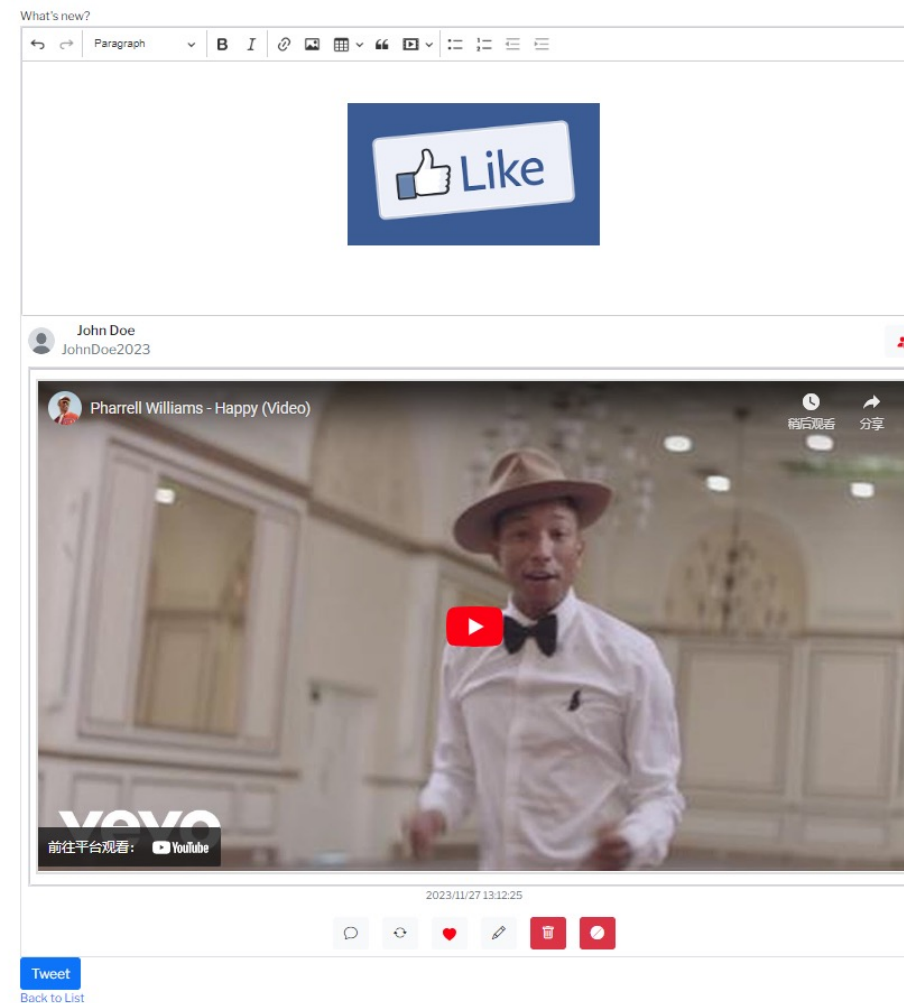
```
[HttpPost]
[Route("HandleFollow")]
0 references
public async Task<IActionResult> HandleFollow([FromBody] string userId) ...
}
```

# SOLUTION: CKEDITOR

One of the most popular Richtext editor

- Text format, Images, Links, Tables...
- Customized interface
- Cloud platform & EasyImage
- Edit mode & readonly mode

## AddEditTweet





# SOLUTION: CASCADING IN SQL SERVER

- A table cannot appear more than once in a cascading tree

Table name

Columns Primary Key Foreign Keys Check Constraints Indexes General

+ New Foreign Key

Name	Foreign Table	Remove
FK_Tweets_AspNetUsers_AuthorId	dbo.AspNetUsers	
FK_Tweets_Tweets_ParentTweetId	dbo.Tweets	

```
await context.Database.  
ExecuteSqlInterpolatedAsync($"UPDATE Tweets  
SET ParentTweetId = NULL WHERE ParentTweetId  
= {tweet.Id}");  
context.Tweets.Remove(tweet);  
await context.SaveChangesAsync();
```

Foreign Key Properties

General

Name

Description

Foreign Table

On Update Action

On Delete Action

Is Enabled ☒

Not For Replication ☐

# USING MVC IN RAZOR PAGE

## Razor pages

organizing the application by feature

URL structure directly related to the physical location and filename of the page.

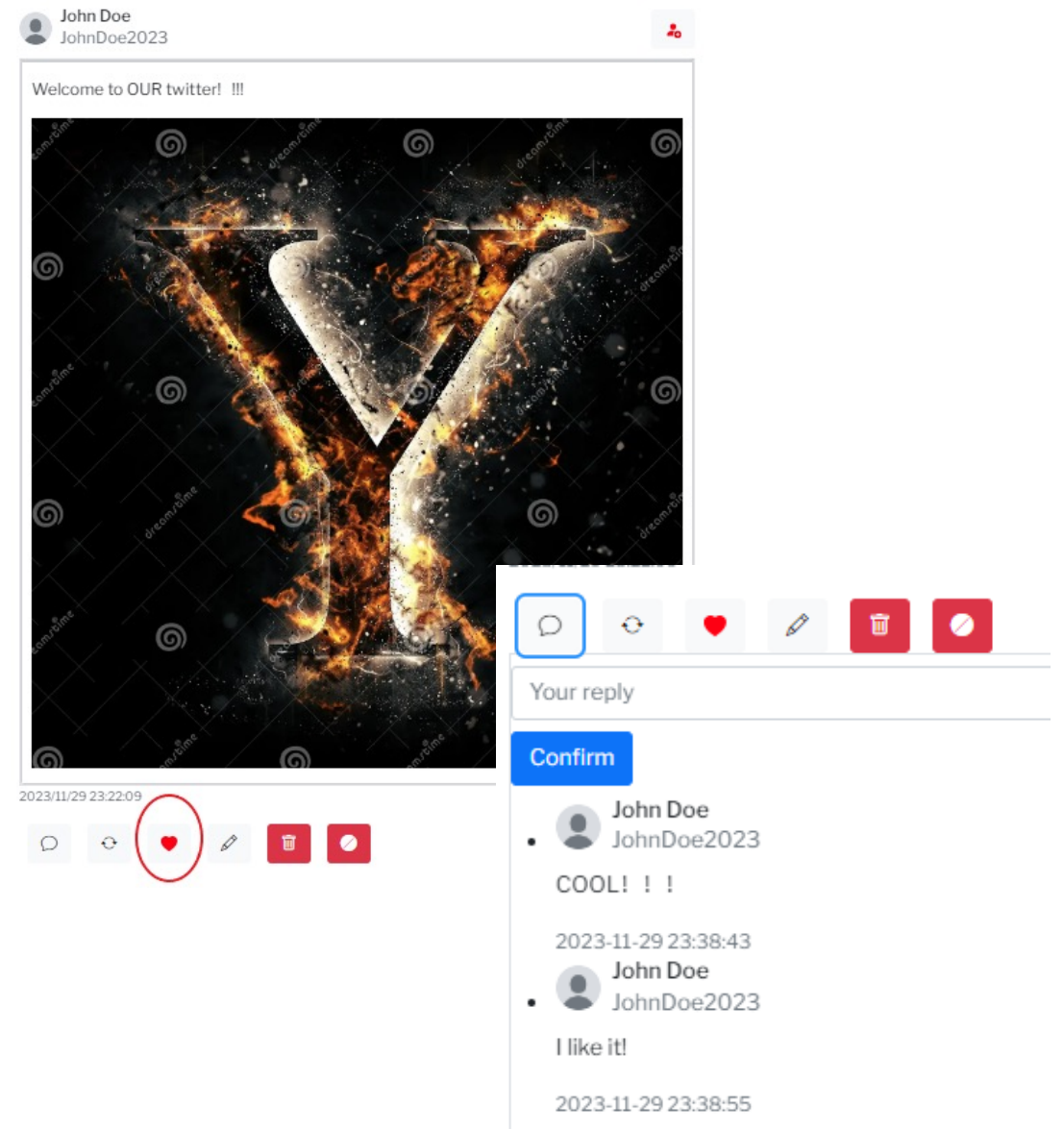
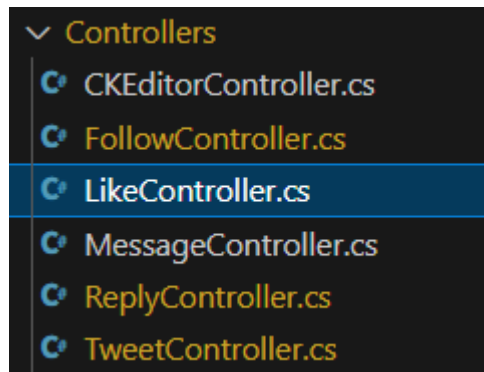
Suitable for small to medium-sized applications

## Asp.net MVC

MVC architecture

Requires configuring routing rules

Suitable for large, complex applications



# SOLUTION: RESET PASSWORD BY EMAIL

```
using(var smtp = new SmtpClient())
{
    var credential = new NetworkCredential
    {
        UserName = smtpUserName,
        Password = smtpPassword
    };
    smtp.Credentials = credential;
    smtp.Host = smtpHost;
    smtp.Port = smtpPort;
    smtp.EnableSsl = enableSsl;
    var message = new MailMessage();
    message.To.Add(email);
    message.Subject = "Y - Reset Password";
    message.Body = $"Dear {user.UserName} Click <a href=\"{resetLink}\">here</a> to reset your password.";
    message.IsBodyHtml = true;
    message.From = new MailAddress("jac2340575@gmail.com");
    await smtp.SendMailAsync(message);
}
```

- SMTP service
- Token
- Url Safe

```
var resetToken = await _userManager.
GenerateUserTokenAsync(user, TokenOptions.
DefaultProvider, "ResetPassword");

var resetLink = $"{_httpContextAccessor.
HttpContext.Request.Scheme}://
{_httpContextAccessor.HttpContext.Request.
Host}/ResetPassword?token={WebUtility.
UrlEncode(resetToken)}";
```

Abstract geometric lines in the top-left corner of the slide, consisting of several thin, black, overlapping lines that form a complex, non-representational shape.

# CHALLENGE:

Combining server-side rendering with client-side features



# SOLUTION: ASP.NET WEB API

```
namespace TwitterClone.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
```

Controllers

- CKEditorController.cs
- FollowController.cs
- LikeController.cs
- MessageController.cs
- ReplyController.cs
- TweetController.cs

```
[HttpGet("Check")]
0 references
public async Task<ActionResult<string>> checkMsgs()
{
    try
    {
        var currentUser = await userManager.GetUserAsync(HttpContext.User);
        if (currentUser == null)
        {
            return StatusCode(401, "You must be logged in to check messages.");
        }
        var id = currentUser.Id;
        int unread = db.Messages
            .Where(m => m.Receiver.Id == id && m.IsRead == false).Count();
        return Json(unread);
    }
    catch
    {
        return StatusCode(500, new { err = "Something went wrong." });
    }
}
```

# SOLUTION: ASP.NET WEB API

```
async function checkMsgs() {  
  
    $.ajax({  
        url: '/api/Message/Check',  
        type: 'GET',  
        headers: {  
            'Content-Type': 'application/json',  
        },  
        error: function (jqxhr, status, errorThrown) {  
            console.log(jqxhr.responseText);  
        }  
    }).done(function (data) {  
        if (data > 0) {  
            $('#msgCount').html(data);  
            $('#msgCount').show();  
        } else {  
            $('#msgCount').hide();  
        }  
    });  
}
```

```
$(document).ready(function () {  
  
    setInterval(checkMsgs, 3000);  
});
```



Messages

Name	×	Headers	Preview	Response
Check	1	1		
Check				
log_event?alt=json&key=AlzaSyAO...				
Check				

# CHALLENGE: ACCURATE DATETIMES ACROSS DIFFERENT TIMEZONES

hey

2h

When you really think about it, what even is time?

29m

# SOLUTION: RTFM (OR RATFM)

Starting with the .NET Framework version 2.0, the value returned by the `ToUniversalTime` method is determined by the `Kind` property of the current `DateTime` object. The following table describes the possible results.

Kind	Results
Utc	No conversion is performed.
Local	The current <code>DateTime</code> object is converted to UTC.
Unspecified	The current <code>DateTime</code> object is assumed to be a local time, and the conversion is performed as if <code>Kind</code> were <code>Local</code> .

```
var newMsg = new Message
{
    Sender = currentUser,
    Receiver = receiver,
    Content = msg.Content,
    SentAt = DateTime.Now.ToUniversalTime()
};
```

```
function getTime(datetime) {
    var diff = new Date().getTimezoneOffset();
    var newDate = new Date(datetime);
    newDate = new Date(newDate.getTime() - (diff * 60000));
    var millisec = Date.now() - newDate;
```

~~`c.Msg.SentAt = c.Msg.SentAt.ToUniversalTime();`~~

1 \_\_\_\_\_ Suspend Tweet/User list, suspend user

2 \_\_\_\_\_ Notification

3 \_\_\_\_\_ Threads

4 \_\_\_\_\_ Expanded user settings

## FUTURE WORK



# SUMMARY

- We achieved a likeness to the real X with ASP.NET Core
- User accounts replicate most of the core features
- There are many microblogging features to interact with:
  - Post a tweet in rich text
  - Reply, like, retweet others' tweets
  - Report tweets to admin
  - Follow and unfollow other users, view other user's profiles
  - Send and receive messages
  - Suggestions, trends, advanced search
  - Functional admin controls to suspend/unsuspend tweet.

A series of white, thin, overlapping geometric lines and polygons on a black background, located on the left side of the slide. The lines form various shapes, including triangles and quadrilaterals, some of which are nested or intersecting.

# THANK YOU