

# Deliverables: Code Documentation

---

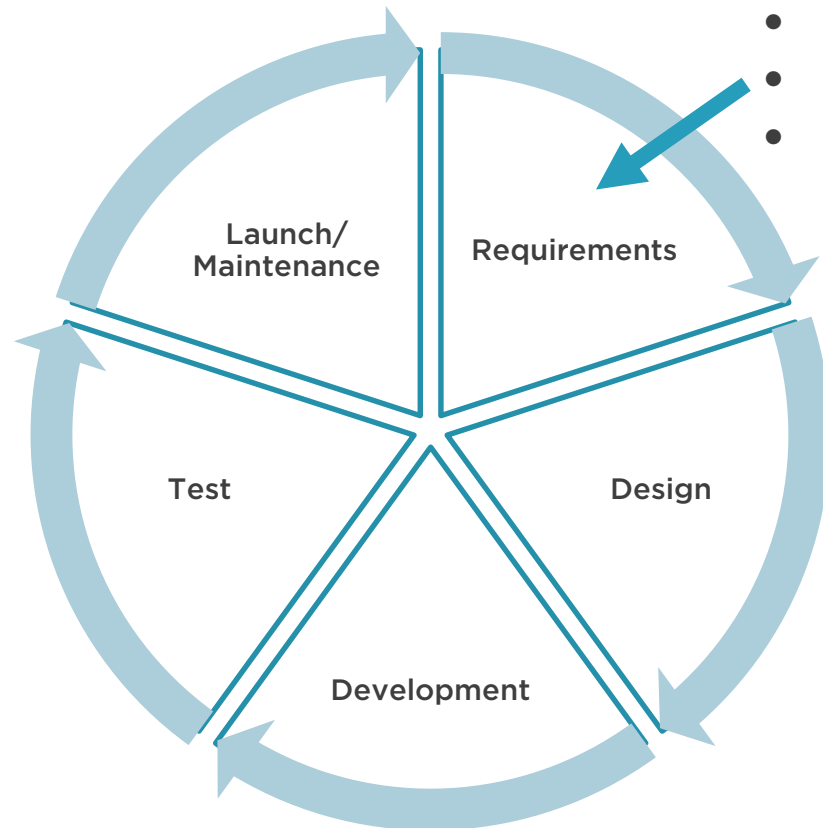


**Amber Israelsen**

DEVELOPER, AUTHOR, TRAINER

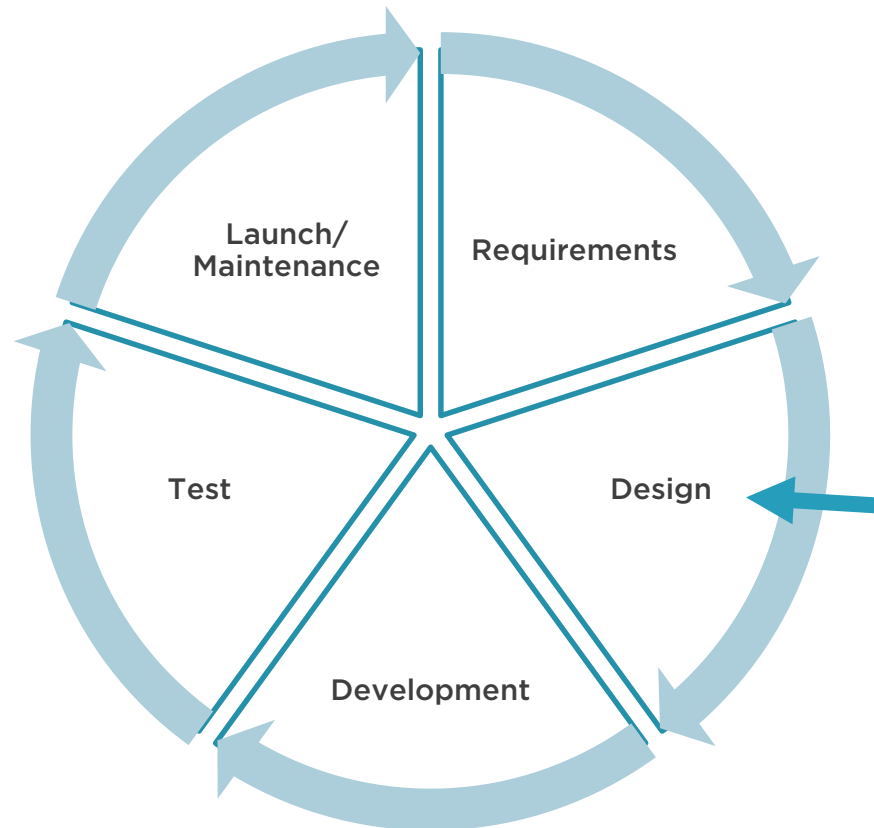
[www.amberisraelsen.com](http://www.amberisraelsen.com)





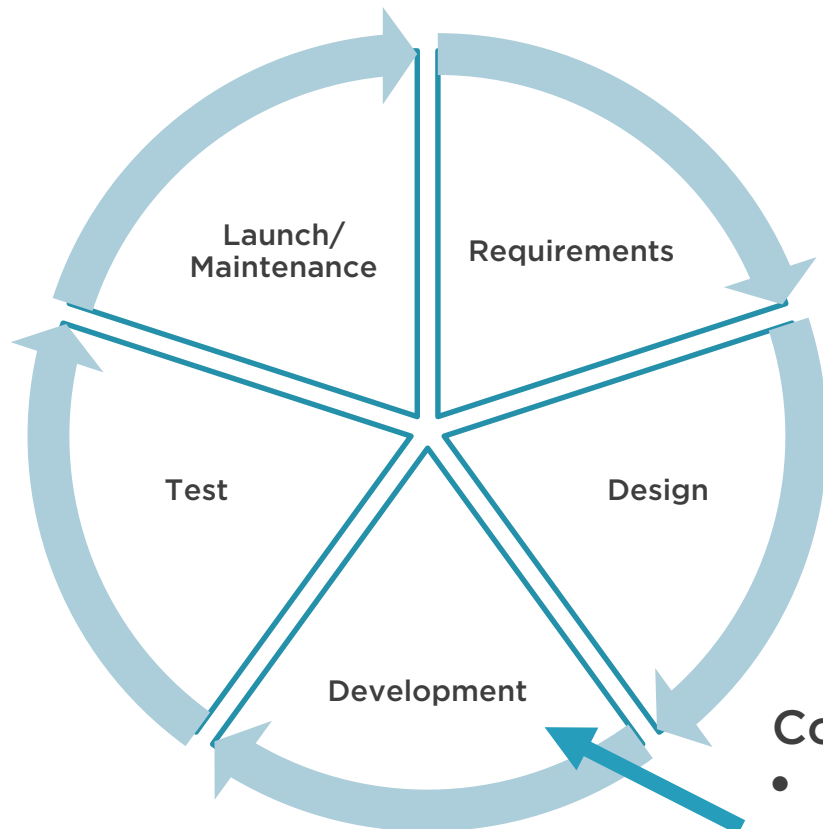
- User stories
- Traditional (text-based)
- Use cases





**Software Design Document**

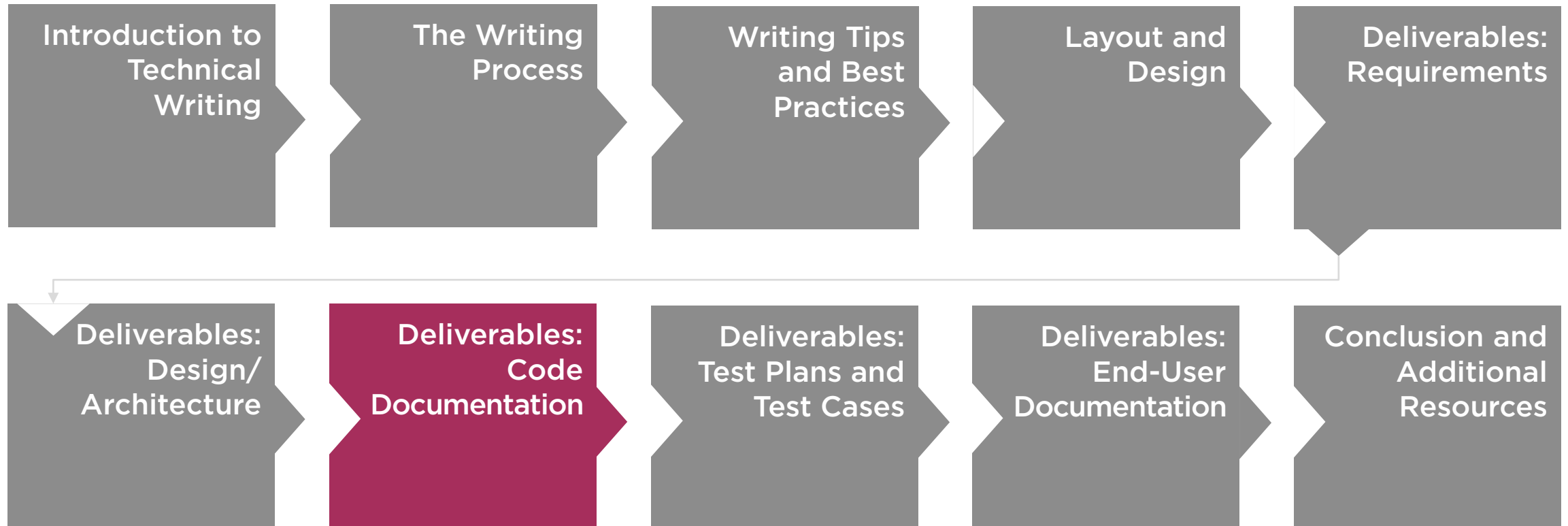




- Code Documentation**
- Code comments
  - API documentation
  - READMEs



# Course Outline



# Code Comments

---



Everything should  
be commented.



Ryan

Nothing should  
be commented.



Tim



That's what *you* think.



Ryan

My code is so good,  
it doesn't require  
comments.



Tim





That's what *you* think.



Ryan

Who's correct?

My code is so good,  
it doesn't require  
comments.



Tim



That's what *you* think.

My code is so good,  
it doesn't require  
comments.

**Both have valid points**



Ryan



Tim



```
/// <summary>
/// Gets the time zone of the prescribing pharmacy based on US state and city
/// </summary>

/// <param name="state">State where pharmacy is located</param>
/// <param name="city">City where pharmacy is located</param>
/// <returns>Time zone for the pharmacy</returns>
```

```
public string GetPharmacyTimeZone(string state, string city)
{
    string timeZoneID = TimeZoneLookup.Find(state, city);
    TimeZoneInfo tzInfo = TimeZoneInfo.FindSystemTimeZoneById(timeZoneID);
    return tzInfo.ToString();
}
```



**NEWS ALERT!**  
**PHARMALANTIS goes international!**



```
/// <summary>
/// Gets the time zone of the prescribing pharmacy based on US state and city
/// </summary>
/// <param name="state">State where pharmacy is located</param>
/// <param name="city">City where pharmacy is located</param>
/// <returns>Time zone for the pharmacy</returns>
public string GetPharmacyTimeZone(string state, string city)
{
    string timeZoneID = TimeZoneLookup.Find(state, city);
    TimeZoneInfo tzInfo = TimeZoneInfo.FindSystemTimeZoneById(timeZoneID);
    return tzInfo.ToString();
}
```



```
/// <summary>
/// Gets the time zone of the prescribing pharmacy based on US state and city
/// </summary>
/// <param name="state">State where pharmacy is located</param>
/// <param name="city">City where pharmacy is located</param>
/// <returns>Time zone for the pharmacy</returns>
public string GetPharmacyTimeZone(string country, string state, string city)
{
    string timeZoneID = TimeZoneLookup.Find(country, state, city);
    TimeZoneInfo tzInfo = TimeZoneInfo.FindSystemTimeZoneById(timeZoneID);
    return tzInfo.ToString();
}
```



```
/// <summary>
/// Gets the time zone of the prescribing pharmacy based on US state and city
/// </summary>
/// <param name="state">State where pharmacy is located</param>
/// <param name="city">City where pharmacy is located</param>
/// <returns>Time zone for the pharmacy</returns>
public string GetPharmacyTimezone(string country, string state, string city)
{
    string timeZone = TimeZoneLookup.Find(country, state, city);
    TimeZoneInfo tzInfo = TimeZoneInfo.FindSystemTimeZoneById(timeZoneID);
    return tzInfo.ToString();
}
```

So what's the problem?



```
/// <summary>
/// Gets the time zone of the prescribing pharmacy based on US state and city
/// </summary>
/// <param name="state">State where pharmacy is located</param>
/// <param name="city">City where pharmacy is located</param>
/// <returns>Time zone for the pharmacy</returns>
```

```
public string GetPharmacyTimeZone(string country, string state, string city)
{
    string timeZoneID = TimeZoneLookup.Find(country, state, city);
    TimeZoneInfo tzInfo = TimeZoneInfo.FindSystemTimeZoneById(timeZoneID);
    return tzInfo.ToString();
}
```





There's a bug.



```
/// <summary>
/// Gets the time zone of the prescribing pharmacy based on US
/// </summary>
/// <param name="state">State where pharmacy is located</param>
/// <param name="city">City where pharmacy is located</param>
/// <returns>Time zone for the pharmacy</returns>
public string GetPharmacyTimeZone(string country, string state, string city)
{
    string timeZoneID = TimeZoneLookup.Find(country, state, city);
    TimeZoneInfo tzInfo = TimeZoneInfo.FindSystemTimeZoneById(timeZoneID);
    return tzInfo.ToString();
}
```



Let me read through the code AND the comments.



```
/// <summary>
/// Gets the time zone of the prescribing pharmacy based on US
/// </summary>
/// <param name="state">State where pharmacy is located</param>
/// <param name="city">City where pharmacy is located</param>
/// <returns>Time zone for the pharmacy</returns>
public string GetPharmacyTimeZone(string country, string state, string city)
{
    string timeZoneID = TimeZoneLookup.Find(country, state, city);
    TimeZoneInfo tzInfo = TimeZoneInfo.FindSystemTimeZoneById(timeZoneID);
    return tzInfo.ToString();
}
```



Hmmm...they don't match.  
I wonder which is correct.



```
/// <summary>
/// Gets the time zone of the prescribing pharmacy based on US
/// </summary>
/// <param name="state">State where pharmacy is located</param>
/// <param name="city">City where pharmacy is located</param>
/// <returns>Time zone for the pharmacy</returns>
public string GetPharmacyTimeZone(string country, string state, string city)
{
    string timeZoneID = TimeZoneLookup.Find(country, state, city);
    TimeZoneInfo tzInfo = TimeZoneInfo.FindSystemTimeZoneById(timeZoneID);
    return tzInfo.ToString();
}
```



Everything should  
be commented.



Ryan



Medium

Nothing should  
be commented.



Tim



Everything should  
be commented.



Ryan



Medium

Nothing should  
be commented.



Tim



```
/// <summary>  
/// Constructor  
/// </summary>  
public class Patient  
{  
}
```

How NOT to Use Comments  
To state something obvious or redundant



```
/// <summary>
/// Updates patient's address
/// </summary>
/// <param name="patient"></param>
private void UpdatePatientAddress(int patientID, AddressInfo address)
{
    //Update patient's address
    //...
    //...
}
```

How NOT to Use Comments  
To state something obvious or redundant



```
//Instantiate a new instance of a patient  
Patient patient = new Patient();
```

How NOT to Use Comments  
To state something obvious or redundant





```
//Check to see if the SMS failed by checking for 4  
if (smsMessage.Status == 4)  
{  
    //Do something  
}
```

## How NOT to Use Comments

To explain poorly-written or poorly-designed code



```
private void ABigBloatedMethod()
{
    //Do the first thing
    //...

    //Do the second thing
    //...

    //Do the third thing
    //...

    //Keep going forever
}
```

## How NOT to Use Comments

**To explain poorly-written or poorly-designed code**



```
// Check to see if the SMS failed by checking for 4
// if (smsMessage.Status == 4)
// {
//     //Do something
// }
```

## How NOT to Use Comments

To *sort of* delete code



Everything should  
be commented.



Ryan



Medium

Nothing should  
be commented.



Tim



```
//Encapsulates logic for sending SMS messages and taking  
//action based on a patient's response
```

## How TO Use Comments

### High-level comments

Generally useful at the class level



//TODO Refactor into two methods

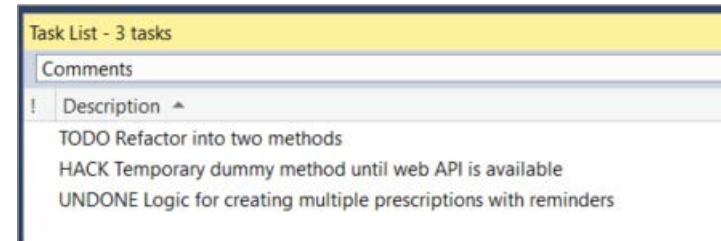
//HACK Temporary dummy method until web API is available

//UNDONE Logic for creating multiple prescriptions with reminders

## How TO Use Comments

**TODO, HACK and UNDONE**

**Use the Visual Studio Task List**



# Clean Code: Writing Code for Humans

By Cory House

Anyone can write code a computer can understand, but professional developers write code \*humans\* can understand. Clean code is a reader-focused development style that produces software that's easy to write, read and maintain.

Start free trial now

Table of contents

Description

Exercise files

Transcript

Discussion

Learning check

Expand all

|                |         |   |
|----------------|---------|---|
| ▶ Introduction | 12m 25s | ▼ |
| ▶ Principles   | 19m 48s | ▼ |
| ▶ Naming       | 16m 5s  | ▼ |
| ▶ Conditionals | 26m 35s | ▼ |
| ▶ Functions    | 26m 52s | ▼ |

## Course info

Level Intermediate

Rating ★★★★★ (1508)

Duration 3h 10m

Released 7 Oct 2013

## Course authors



Cory House

Cory is an independent consultant with over 15 years of experience in software development. He is a Microsoft MVP, ASP Insider, and a member of the Telerik developer experts program.

Start free trial now

## Share course



# API Documentation

---





# API Documentation

Instructions for how to effectively use APIs of hardware or software



The best API in the world is  
useless without good  
documentation



```
string pharmacyTimeZone = pharmacy.get
```

```
string Pharmacy.GetPharmacyTimeZone(string state, string city)
```

GetHashCode

GetPharmacyTimeZone

GetType



```
string pharmacyTimeZone = pharmacy.g
```

```
string Pharmacy.GetPharmacyTimeZone(string state, string city)
```

```
Gets the time zone of the prescribing pharmacy based on US state and city
```

- ✱ Equals
- ✱ GetHashCode
- ✱ **GetPharmacyTimeZone**
- ✱ GetType
- ✱ ToString



# API Audiences

**Newbie  
developers**

**Developers  
needing to debug**

**Businesspeople or  
developers  
evaluating the API**



# What Should Be Included?

**Reference documentation**

**Overview and concepts**

**Tutorials/training**

**Installation/getting started/troubleshooting documentation**

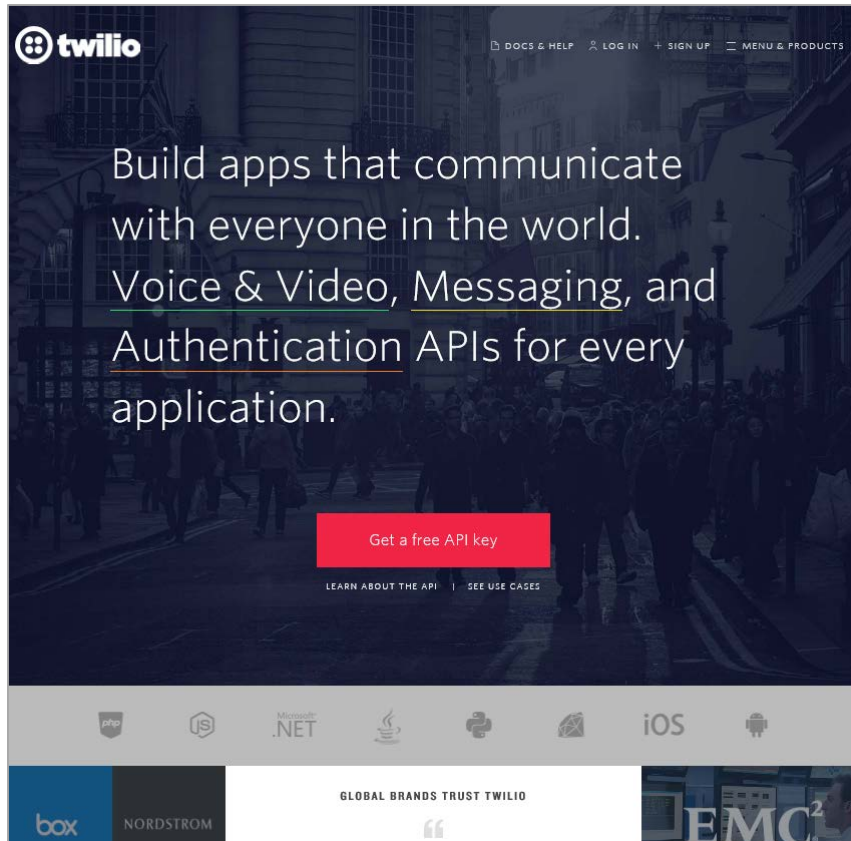
**SDK tools documentation**

**License information**

For more, visit

[https://en.wikipedia.org/wiki/Application\\_programming\\_interface](https://en.wikipedia.org/wiki/Application_programming_interface)

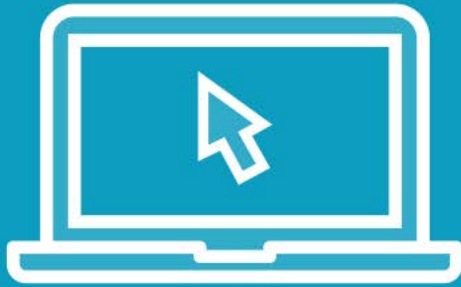




Send SMS



# Demo



**TWILIO: An example of awesome  
API documentation**





# Popular Tools for Generating API Documentation

[swagger.io](https://swagger.io)

[mashery.com](https://mashery.com)

[apiary.io](https://apiary.io)

[raml.org](https://raml.org)

**ASP.NET Web API**



# README Files



---

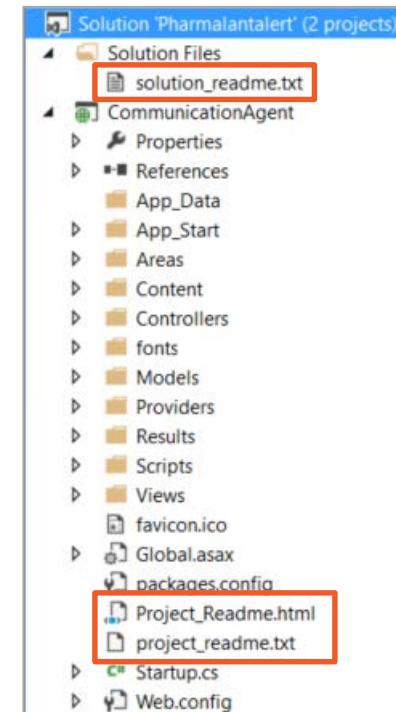


# README File

A file (usually .txt) that helps users/other developers know how to do things with your software



|  | Name       | Date modified    | Type          |
|---|------------|------------------|---------------|
|  | README.txt | 6/4/2016 3:25 PM | Text Document |



“Until you’ve written about your software, you have no idea what you’ll be coding.”

**Tom Preston-Werner, Co-Founder of Github**



Write your README before  
your code



Bad README files







# Goood README files



## REQUIRED

Date

Software name and version number

Short description of the software

Installation requirements and instructions

Copyright and licensing information

Contact information for the developer or distributor

## IF IT MAKES SENSE

A file manifest (list of files included)

Configuration instructions

Operating instructions

Known bugs

Troubleshooting

Credits and acknowledgments

Change log (usually for developers)

News/updates (usually for users)



# Summary

---



# Summary



**Code comments should be used to summarize information and document things that are not obvious**

- Do not use them as a crutch

**Good API documentation will make everyone happier**

- Be thorough and think of all audiences

**README files are your opportunity to make a good (or bad) first impression with developers and users**



Up next

