Open-Source Technology Use Report

Proof of knowing your stuff in CSE312

Guidelines

Provided below is a template you must use to write your report for each of the technologies you use in your project.

Here are some things to note when working on your report, specifically about the **General Information & Licensing** section for each technology.

- **Code Repository**: Please link the code and not the documentation. If you'd like to refer to the documentation in the **Magic** section, you're more than welcome to, but we'd like to see the code you're referring to as well.
- **License Type**: Three letter acronym is fine.
- **License Description**: No need for the entire license here, just what separates it from the rest.
- **License Restrictions**: What can you *not* do as a result of using this technology in your project? Some licenses prevent you from using the project for commercial use, for example.
- **Who worked with this?**: It's not necessary for the entire team to work with every technology used, but we'd like to know who worked with what.

Also, feel free to extend the cell of any section if you feel you need more room.

If there's anything we can clarify, please don't hesitate to reach out! You can reach us using the methods outlined on the course website or see us during our office hours.

[wtforms]

General Information & Licensing

Code Repository	https://github.com/wtforms/wtforms
License Type	BSD 3-Clause "New" or "Revised" License
License Description	• A permissive license is like the BSD 2-Clause License, but with a 3rd clause that prohibits others from using the name of the project or its contributors to promote derived products without written consent. • Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: • Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. • Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. • Neither the name of the copyright holder nor the names of its

	contributors may be used to endorse or promote products derived from this software without specific prior written permission.
License Restrictions	● Liability ● Warranty
Who worked with this?	All team members had some exposure to this framework but Cooper was the primary developer on this.

Use as many of the sections below as needed, or create more, to explain every function, method, class, or object type you used from this library/framework.

[flaskform]

Purpose

Replace this text with some that answers the following questions for the above tech: ● What does this tech do for you in your project? Its a flask specific superclass and we use this class to inherit registration form , login form and chatform ● Where specifically is this tech used in your project? Give us some details like file location and line number, if applicable. If too cumbersome, a general description of where it's used for a given purpose is fine as well. It is used in auth/form.py line 6 and 19 and chat/form.py line 4

Magic ★★˚⋆｡˚ ☽ ˚｡⋆★

Dispel the magic of this technology. Replace this text with some that answers the following questions for the above tech:

- How does this technology do what it does for you in the **Purpose** section of this report? Please explain this in detail, starting from after the TCP socket is created. Remember, to be allowed to use a technology in your project, you must be able to know how it works.

It is a flask specific subclass of wtforms.
Create form class in application
Inherits class RegistrationForm from flaskform(represents the web form)
Implement the fields in template.

- Where is the specific code that does what you use the tech for? You ***must*** provide a link to the specific file in the repository for your tech with a line number or number range.
    - If there is more than one step in the chain of calls *(hint: there will be)*, you must provide links for the entire chain of calls from your code, to the library code that actually accomplishes the task for you.
    - Example: If you use an object of type HttpRequest in your code which contains the headers of the request, you must show exactly how that object parsed the original headers from the TCP socket. This will often involve tracing through multiple libraries and you must show the entire trace through all these libraries with links to all the involved code.

*This section may grow beyond the page for many features.
https://github.com/wtforms/flask-wtf/blob/main/src/flask_wtf/form.py#L27

In lines 24- 30
 Defines class form

[StringField]

Purpose

Replace this text with some that answers the following questions for the above tech:

- What does this tech do for you in your project?

The stringfield class creates placeholder with specified validators

- Where specifically is this tech used in your project? Give us some details like file location and line number, if applicable. If too cumbersome, a general description of where it's used for a given purpose is fine as well.

It is used in auth/forms.py line7 and will take text as input in username placeholder. We use this to take register username from the user

Magic ⋆★˚｡⋆୨୧˚ ☽ ˚⁀➴｡˚★彡⋆

Dispel the magic of this technology. Replace this text with some that answers the following questions for the above tech:

- How does this technology do what it does for you in the **Purpose** section of this report? Please explain this in detail, starting from after the TCP socket is created. Remember, to be allowed to use a technology in your project, you must be able to know how it works.
    - To start, a stringlField is declared with a label of “username”, as well as a requirement that the email is between 5 and 34 characters. This username field is a text field, except the input type is overridden to be “usernamel”.
    - A widget is created as an username input, which sets the input type as an eusernmae for the html as well as the validation attributes.

Where is the specific code that does what you use the tech for? You ***must*** provide a link to the specific file in the repository for your tech with a line number or number range.

- If there is more than one step in the chain of calls *(hint: there will be)*, you must provide links for the entire chain of calls from your code, to the library code that actually accomplishes the task for you.
- Example: If you use an object of type HttpRequest in your code which contains the headers of the request, you must show exactly how that object parsed the original headers from the TCP socket. This will often involve tracing through multiple libraries and you must show the entire trace through all these libraries with links to all the involved code.

*This section may grow beyond the page for many features.
In https://github.com/wtforms/wtforms/blob/master/src/wtforms/fields/simple.py

(class StringField(Field):) -> Line(s) 55 - 69 creates a string field with a text input widget and two methods - process_form data and value. Widgets are located in

In our code in auth/forms.py in our code repository it is used in line 7

[PasswordField]

Purpose

Replace this text with some that answers the following questions for the above tech:

- What does this tech do for you in your project?

PasswordField represents input type password.It is used in our register functionality to take password from a user

- Where specifically is this tech used in your project? Give us some details like file location and line number, if applicable. If too cumbersome, a general description of where it’s used for a given purpose is fine as well.

It is used in auth/forms.py page line 9
It takes input as text as user for logging the user in

Magic ⋆★˚｡⋆୨୧˚ ☽˚｡⋆ ★彡⋆ ༄

Dispel the magic of this technology. Replace this text with some that answers the following questions for the above tech:
How does this technology do what it does for you in the **Purpose** section of this report? Please explain this in detail, starting from after the TCP socket is created. Remember, to be allowed to use a technology in your project, you must be able to know how it works.

- To start, an PasswordlField is declared with a label of "password", as well as a requirement that the password must be entered(data required). This password field is a string field, except the input type is overridden to be "email".

● A widget is created as an Email input, which sets the input type as an email for the html as well as the validation attributes.

- Where is the specific code that does what you use the tech for? You ***must*** provide a link to the specific file in the repository for your tech with a line number or number range.
    - If there is more than one step in the chain of calls *(hint: there will be)*, you must provide links for the entire chain of calls from your code, to the library code that actually accomplishes the task for you.
    - Example: If you use an object of type HttpRequest in your code which contains the headers of the request, you must show exactly how that object parsed the original headers from the TCP socket. This will often involve tracing through multiple libraries and you must show the entire trace through all these libraries with links to all the involved code.

*This section may grow beyond the page for many features.

In https://github.com/wtforms/wtforms/blob/master/src/wtforms/fields/simple.py

(class passworlField(StringField):) -> Line(s) 80-90 creates a password field, and overrides the widget and input type as 'passwordl'In https://github.com/wtforms/wtforms/blob/master/src/wtforms/fields/simple.py

(class StringField(Field):) -> Line(s) 55 - 90 creates a string field with a text input widget and two methods - process_form data and value. Widgets are located in https://github.com/wtforms/wtforms/blob/master/src/wtforms/widgets/core.py

In https://github.com/wtforms/wtforms/blob/master/src/wtforms/widgets/core.py

(class password lInput(Input):) -> Line(s) 190-210 - A password input is created (extending off the base input class) and overwrites the call method. Its input is of type password and has a required data requirement.

In our code it is used in line 9 auth/forms.py

[BooleanField]

Purpose

Replace this text with some that answers the following questions for the above tech:

- What does this tech do for you in your project?

We have used the Boolean field to see if the terms and conditions have been accepted by the user while logging in.

- Where specifically is this tech used in your project? Give us some details like file location and line number, if applicable. If too cumbersome, a general description of where it’s used for a given purpose is fine as well.

It is implemented in auth/forms.py line 15
Here it is used as an input type checkbox to check if terms and conditions have been checked during register a user

Magic ⋆★˚⋆｡˚ ☽˚｡⋆ ˚★彡⋆

Dispel the magic of this technology. Replace this text with some that answers the following questions for the above tech:

- How does this technology do what it does for you in the **Purpose** section of this report? Please explain this in detail, starting from after the TCP socket is created. Remember, to be allowed to use a technology in your project, you must be able to know how it works.
    - To start, the boolean field is initialized with a label of ''I accept the Terms Of Service' and a 1 element list that means this data must be submitted.
    - In the initializer field, a widget is created with a checkbox input, which extends off of the basic input class of wtforms.
        - In this basic input, any validations attributes are set as required, and the function html_params is converted to a static method of the Input Class
    - A default "false" value is then set as a Tuple of 3 elements: a literal False, the string "false" and an empty string.
    - The boolean field defines two functions; process data and process form data.
        - process data takes a boolean value and assigns it to self.data
        - process form data takes a value list and assigns the truth value to the first element in that value list.
- Where is the specific code that does what you use the tech for? You ***must*** provide a link to the specific file in the repository for your tech with a line number or number range.

In https://github.com/wtforms/wtforms/blob/master/src/wtforms/fields/simple.py

(class BooleanField(Field):) -> Line(s) 20 - 52 - a widgets variable is created with a checkbox input. This comes from https://github.com/wtforms/wtforms/blob/master/src/wtforms/widgets/core.py

In https://github.com/wtforms/wtforms/blob/master/src/wtforms/widgets/core.py

(class CheckboxInput(Input):) -> Line(s) 224 - 236 - A Checkbox input is created (extending off the base input class) and overwrites the call method.

In https://github.com/wtforms/wtforms/blob/master/src/wtforms/widgets/core.py

(class Input:) -> Line(s) 153 - 179 - The base input class for the checkbox input. It creates the html input tags for various text inputs.

Back in In https://github.com/wtforms/wtforms/blob/master/src/wtforms/fields/simple.py
(class BooleanField(Field):) -> Line(s) 20 - 52 - default false_values are created and an init function. Three functions are created - process_data, process_formdata and _value, which fetch and process values and set the internal state of the field.

In https://github.com/wtforms/wtforms/blob/master/src/wtforms/fields/core.py

(class Field:) -> Line(s) 15 - 420 - the default field for all input fields (in other sections as well). It provides the skeleton functionality that the rest of the fields implement.

** MARKUP CLASS DESCRIBED IN TEXTAREAFIELD

[EmailField]

Purpose

Replace this text with some that answers the following questions for the above tech:

- What does this tech do for you in your project?
    - It is an Html5 field that represent the email input type as is used
- Where specifically is this tech used in your project? Give us some details like file location and line number, if applicable. If too cumbersome, a general description of where it's used for a given purpose is fine as well.
    - It is implemented in auth/forms.py lines 8 and 20
    - This is an email validator. It basically accepts input text, and verifies if the input is an email address.

Magic ⋆★˚｡⋆ ☽ ˚⁀➴｡˚★彡⋆ ༄

Dispel the magic of this technology. Replace this text with some that answers the following questions for the above tech:

- How does this technology do what it does for you in the **Purpose** section of this report? Please explain this in detail, starting from after the TCP socket is created. Remember, to be allowed to use a technology in your project, you must be able to know how it works.
    - To start, an EmailField is declared with a label of “Email Address”, as well as a requirement that the email is between 6 and 35 characters. This email field is a string field, except the input type is overridden to be “email”.
    - A widget is created as an Email input, which sets the input type as an email for the html as well as the validation attributes.
- Where is the specific code that does what you use the tech for? You ***must*** provide a link to the specific file in the repository for your tech with a line number or number range.

In https://github.com/wtforms/wtforms/blob/master/src/wtforms/fields/simple.py

(class EmailField(StringField):) -> Line(s) 159 - 164 creates an email field, and overrides the widget and input type as ‘email’
In https://github.com/wtforms/wtforms/blob/master/src/wtforms/fields/simple.py

(class StringField(Field):) -> Line(s) 55 - 69 creates a string field with a text input widget and two methods - process_form data and value. Widgets are located in https://github.com/wtforms/wtforms/blob/master/src/wtforms/widgets/core.py

In https://github.com/wtforms/wtforms/blob/master/src/wtforms/widgets/core.py

(class EmailInput(Input):) -> Line(s) 224 - 236 - An Email input is created (extending off the base input class) and overwrites the call method. Its input is of type email and has a required, max length, min length and pattern requirement.

In https://github.com/wtforms/wtforms/blob/master/src/wtforms/widgets/core.py

(class Input:) -> Line(s) 153 - 179 - The base input class for the email input. It creates the html input tags for various text inputs. In this case, it’s email.

In https://github.com/wtforms/wtforms/blob/master/src/wtforms/fields/core.py

(class Field:) -> Line(s) 15 - 420 - the default field for all input fields (in other sections as well). It provides the skeleton functionality that the rest of the fields implement.

** MARKUP CLASS DESCRIBED IN TEXTAREAFIELD

[validators]

Purpose

Replace this text with some that answers the following questions for the above tech:

- What does this tech do for you in your project?
    - Validators are used to provide custom validation to input fields. Here we have used to validate input fields like username, length of password, accept_tos, email,
- Where specifically is this tech used in your project? Give us some details like file location and line number, if applicable. If too cumbersome, a general description of where it's used for a given purpose is fine as well.
    - We use this is line of input fields.
        - An example is in username stringfield is required to have length of min 4 and max 25 . this can be seen in auth/forms.py line 7.

Magic ⋆★˚｡⋆୨୧˚ ☽˚⁀➷｡˚★彡⋆⭒˚

Dispel the magic of this technology. Replace this text with some that answers the following questions for the above tech:

- How does this technology do what it does for you in the **Purpose** section of this report? Please explain this in detail, starting from after the TCP socket is created. Remember, to be allowed to use a technology in your project, you must be able to know how it works.
    - In our case, we are only using validator to check for usernames length,email length, etc. Basically the stuff we listed above. It does a check to see if the check passes or not. If it does not pass, it sends an error from the class ValidationError(ValueError)
    - We also pass in messages if case the consumer of the product fails the validation check. This message is passed in the form of a message provided by the class ValidationError(ValueError)
- Where is the specific code that does what you use the tech for? You ***must*** provide a link to the specific file in the repository for your tech with a line number or number range.

In src/wtforms/validators.py
The whole file is involved with creating the functionality of validators. The whole file has different functions to provide checks on different things. In our case, we are only using it to validate if the for the things listed above. When the validation fails, it returns a message provided by the class ValidationError(ValueError). This class is in the same file and the message is what ever the user/programmer wants it to say.

*This section may grow beyond the page for many features.

[SubmitField]

Purpose

Replace this text with some that answers the following questions for the above tech:

- What does this tech do for you in your project?
    - This tech allows for the user to render a submit button onto the application.

It represents and input type submit and can take multi-line input. We use for register after a user puts in their username and password and same for login and chat form as well

- Where specifically is this tech used in your project? Give us some details like file location and line number, if applicable. If too cumbersome, a general description of where it's used for a given purpose is fine as well.
    - This tech is used on files that have the name of form.py. It is primarily used to render a submit button for our webpages.
        - In /auth/forms.py line 16
        - In /chat/forms.py line 8

It is used to create a submit input type and checks if a given submit button is pressed.

Magic ⋆★˚｡⋆°⋆☽˚⁀➷｡˚★彡⋆༄

Dispel the magic of this technology. Replace this text with some that answers the following questions for the above tech:

- How does this technology do what it does for you in the **Purpose** section of this report? Please explain this in detail, starting from after the TCP socket is created. Remember, to be allowed to use a technology in your project, you must be able to know how it works.
    - This tech uses a class called “SumbitInput()”. It renders a button with the parameters provided by the user. If the programmer passes along a parameter of ‘Submit’, The button will have a text saying “Submit”.
    - This button represents <input type=”submit”> in HTML.
- Where is the specific code that does what you use the tech for? You ***must*** provide a link to the specific file in the repository for your tech with a line number or number range.

In /src/wtforms/fields/simple.py
Lines 126 to 132 is involved with creating the functionality of SubmitField(). It takes in the parameter of “BooleanField” which is a class defined in the same file. The report for this is above. It also uses a class called SubmitInput(Input) which has more definitions of the structure of how this was created.

In /src/wtforms/widgets/core.py
Lines 278 to 290 is involved with creating the class for SubmitInput(). Throughout these lines, it is returning the value set by the user. This value will be represented as a text on the button.

*This section may grow beyond the page for many features.

[TextAreaField]

Purpose

Replace this text with some that answers the following questions for the above tech:

- What does this tech do for you in your project?
    - This tech allows for us to create a textbox such as the one given to us in the html template in homework 2.
- Where specifically is this tech used in your project? Give us some details like file location and line number, if applicable. If too cumbersome, a general description of where it’s used for a given purpose is fine as well.
    - This tech is used in /chat/forms.py
        - On line 6 is where it is used.

Magic ⋆★˚｡⋆ ˚ · ˚ ☽ ˚⁀➴｡˚★彡⋆༄

Dispel the magic of this technology. Replace this text with some that answers the following questions for the above tech:

- How does this technology do what it does for you in the **Purpose** section of this report? Please explain this in detail, starting from after the TCP socket is created. Remember, to be allowed to use a technology in your project, you must be able to know how it works.
    - This tech uses a class called “TextArea()”. It renders a chatbox with the parameters passed on by the user. To leave it blank, we can pass in an empty string.
    - This is represented as <textarea> in html. It can be used to take multi-line inputs.
- Where is the specific code that does what you use the tech for? You ***must*** provide a link to the specific file in the repository for your tech with a line number or number range.

In /src/wtforms/fields/simple.py
Lines 71 to 77 is involved with the creating the functionality of TextAreaField(). It takes in the parameter of “StringField” which is a class and the report for it is written above. It also uses a class called TextArea() which has more definition of the structure of how this was created.

In /src/wtforms/widgets/core.py
Lines 293 to 311 is involved with creating the class, “TextArea”. It uses the markupsafe library to return the chatbox. The chatbox is literally <textarea> in HTML and is escaping any HTML written by anyone writing in the textbox.

In the markupsafe library in /src/markupsafe/__init__.py
Lines 32 to 220 is involved with creating the functionality of Markup(). On these lines, it has a bunch of defined functions to help aid in creating more functions such as “escape” which is talked about on the next part below. Markup() returns the parameter passed into it as text. Example of what is return is below:

```
 >>> Markup("Hello, <em>World</em>!")
Markup('Hello, <em>World</em>!')
>>> Markup(42)
Markup('42')
```

In /src/markupsafe/_native.py
Lines 6 to lines 27 involved with creating the functionality of escape(). Just like what we do in our homeworks, we such replace the html values with different values. We replace “&” with “&amp”, “>”, with “&gt”, “<”, with “&lt” “ ‘ “, with “&#39” and ‘ “ ‘ with “&#34”. In escape(), it uses the Markup() function which is talked about above and replaces each instance of the character with the escaped value.

```
 >>> Markup.escape("Hello, <em>World</em>!")
Markup('Hello &lt;em&gt;World&lt;/em&gt;!')
```

*This section may grow beyond the page for many features.