

Massey University Albany

# Documentation

159.339 Assignment 3

Cai Gwatkin  
10-20-2017

## Table of Contents

Table of Figures .....	ii
Specifications .....	1
Login .....	1
Registration .....	1
Welcome .....	2
Search .....	3
Design choices .....	4
UI .....	4
Login .....	4
Registration .....	5
Welcome .....	6
Search .....	7
MVC Framework Backend .....	7
Routing .....	7
Controller .....	8
Model .....	9
View .....	10
Templates .....	11
JavaScript/AJAX/jQuery Frontend .....	11
Input validation .....	11
Dynamic data manipulation .....	11
Database schema and relations .....	12
User .....	12
Product .....	12

## Table of Figures

Figure 1 Login page .....	1
Figure 2 Registration page .....	2
Figure 3 Welcome page .....	3
Figure 4 Search page .....	4
Figure 5 Logo with grey background.....	4
Figure 6 Login page closeup .....	5
Figure 7 Registration page closeup.....	5
Figure 8 Registration page with validation closeup.....	6
Figure 9 Welcome page with highlighted menu element.....	7
Figure 10 Search page with search term and sorting on “cost” closeup.....	7
Figure 11 User relation .....	12
Figure 12 Product relation .....	12

## Specifications

Back end stock control system for Toolshed Inc.

The application is a simple look-up of products in the backend stock control system. The system consists of 4 parts: Login, Registration, Welcome, and Search screens.

### Login

*/login*

This page is used to login users to begin using the system. If a user session is not active then navigation to any other page of the website will redirect here.

If an existing username and password combination is entered the user will be logged in to a new session. To logout, the user will click a menu element “logout” or navigate to */logout*

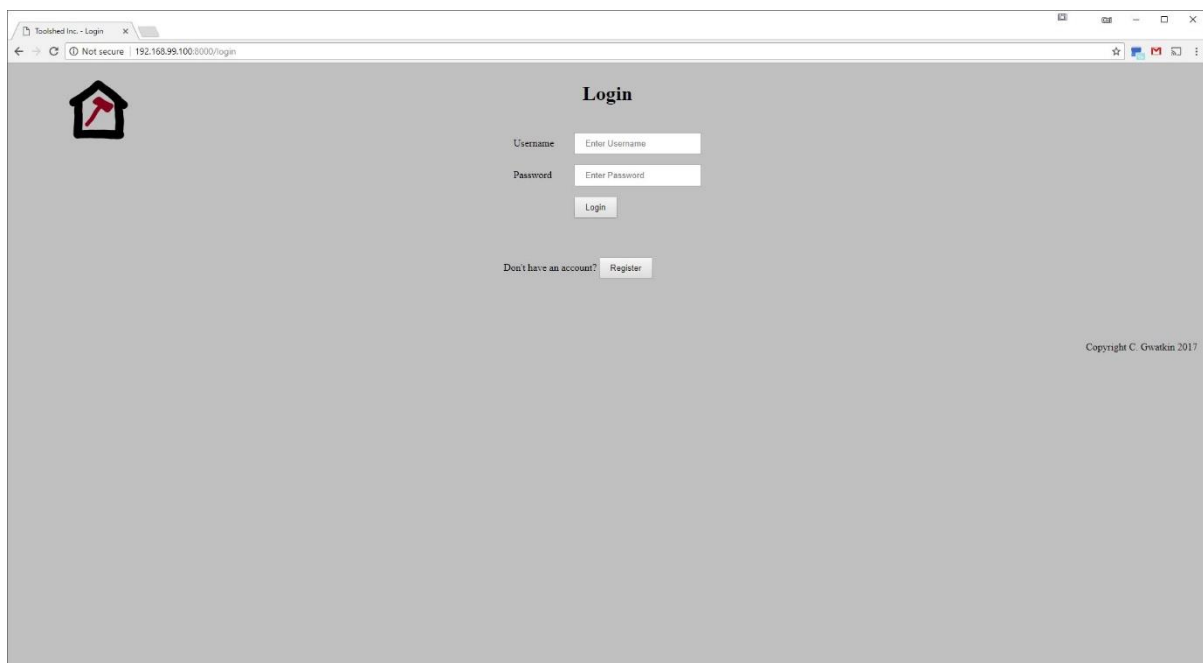


Figure 1 Login page

### Registration

*/register*

This page is for new users to register as a user. This page is navigated to via a button on the login page.

The new user must enter their name, desired username, password, and repeat their password. Front-end validation alerts the user of the following:

- their desired username is already in use
- their password does not conform to the website's password rules

- their repeated password does not match

The user can submit their registration form once all fields have data entered. If their input is valid, a new user is created and they are logged in. If their input is invalid, the page will reload and a warning message will appear informing the user of their error.

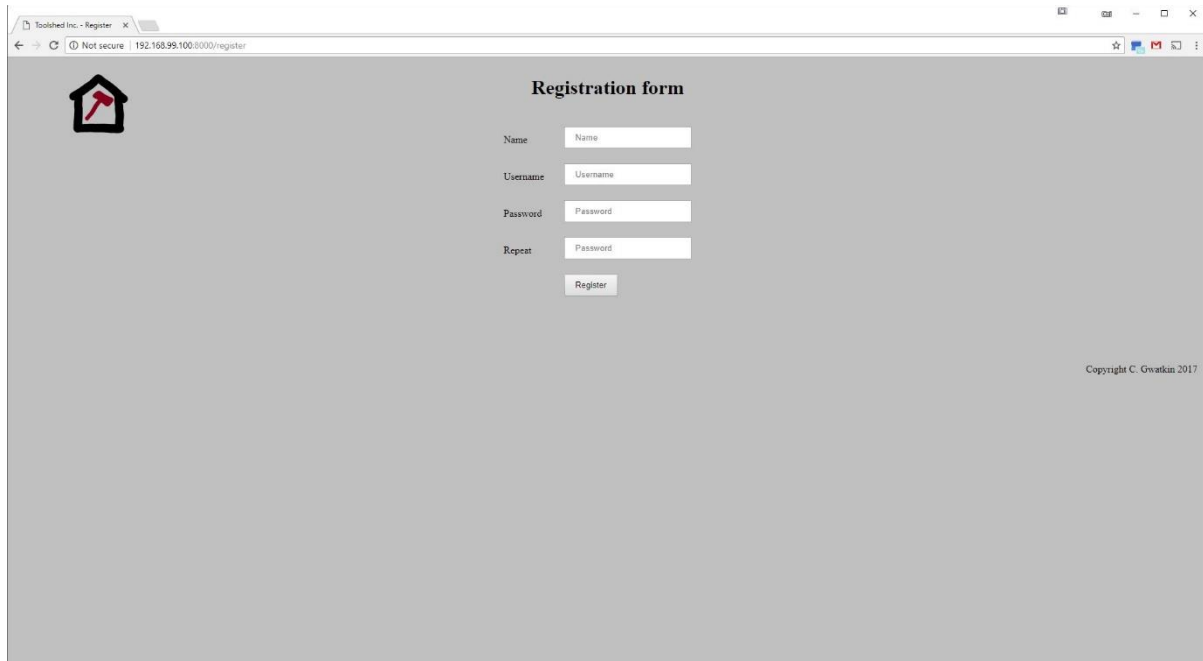
A screenshot of a web browser window displaying a registration form. The browser's address bar shows the URL '192.168.99.100:8000/register'. The page has a light gray background. On the left side, there is a small icon of a house with a red arrow pointing inside. The main content area is titled 'Registration form' in bold. Below the title, there are four input fields: 'Name', 'Username', 'Password', and 'Repeat'. Each field has a label to its left and a text input box to its right. Below the 'Repeat' field is a 'Register' button. In the bottom right corner of the page, there is a small text string: 'Copyright C. Gwosdz 2017'.

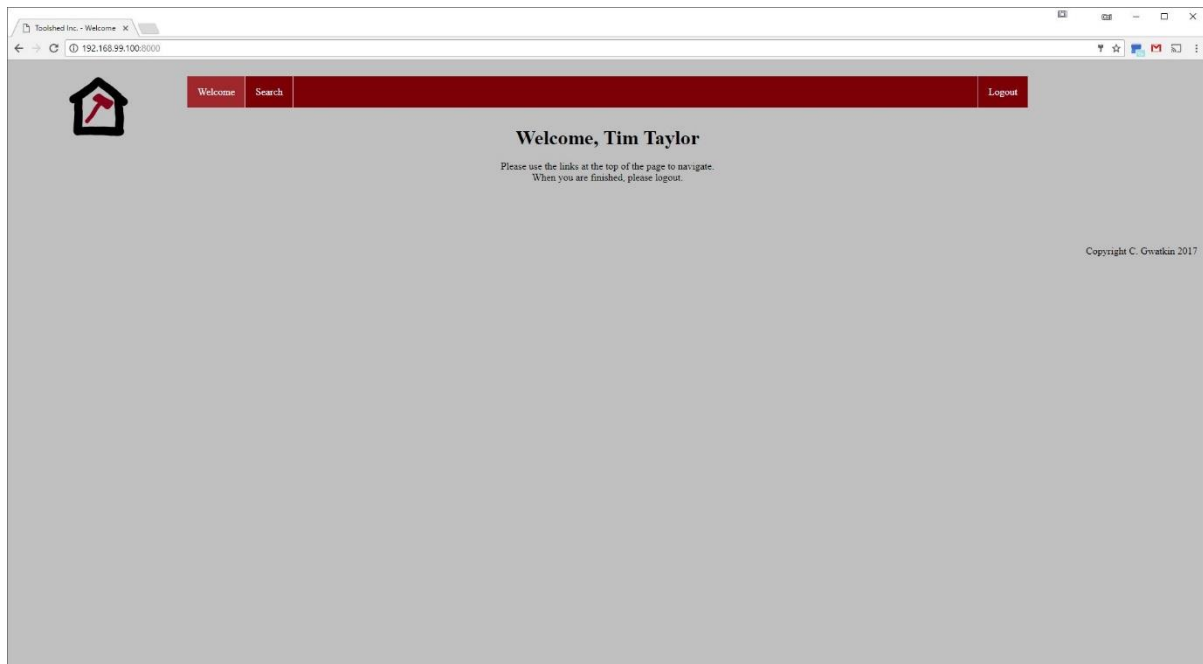
Figure 2 Registration page

## Welcome

/

This page welcomes the user and displays the menu bar. After login or registration, the user is redirected to this page.

From this page, the user can navigate to other pages using the menu bar at the top of the screen. The other pages are Search and Logout.



*Figure 3 Welcome page*

## Search

*/search*

This page allows users to search through the product catalogue.

All products are displayed by default when this screen is first navigated to. The user can begin typing any word into the search box and that partial search will be used to dynamically update the displayed data. Any product containing the partial search term in their name will be displayed.

The user can click on the header of any of the table columns to order the data in the table by that column. A second click of the same column header will order the data in reverse.

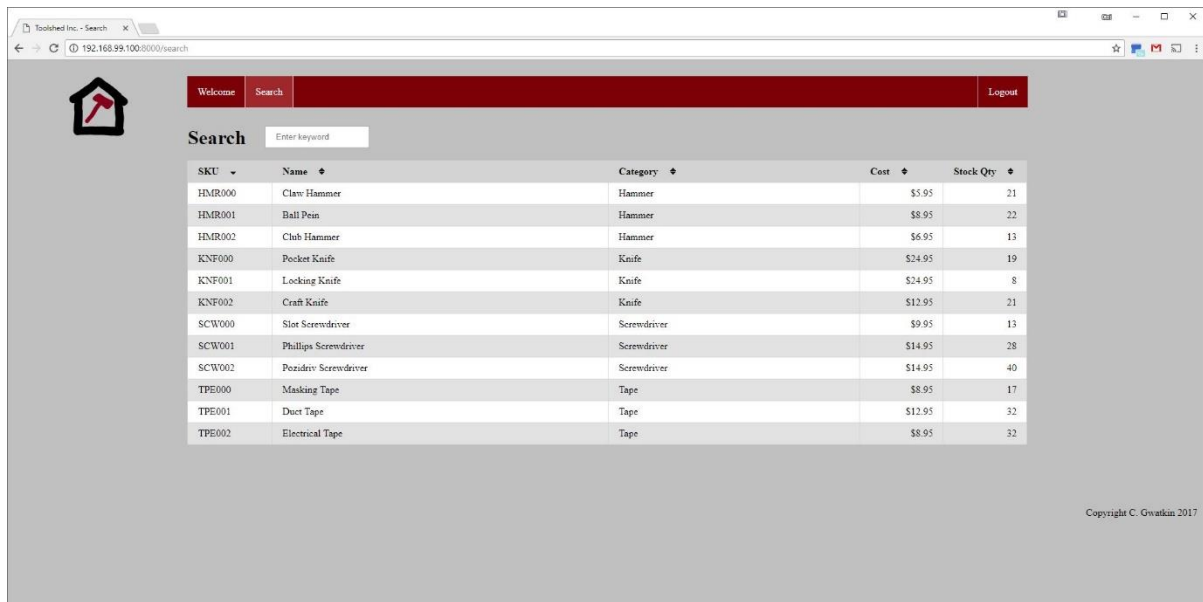


Figure 4 Search page

## Design choices

### UI

The UI is based on mock-ups provided by the client.

Each page has the same background and logo to maintain consistency in user experience. The logo also links to the root page (welcome if logged in, login if not).



Figure 5 Logo with grey background

The active area of the window is the horizontal-centre 70%. Elements in this active area are generally horizontally centred so that the user only needs to look down the centre of their screen. Most pages consist of a menu bar (if user is logged in) and title with some important content underneath. Navigation between pages requires the user to click a button or a menu element.

### Login

The login page consists of two input fields for username and password. This design is kept simple and consistent with most other login pages on the internet.

A “Login” button is used to submit the form. If the user has entered an invalid username and password, the page will refresh with an error message alerting the user. If their input was valid, they will be redirected to the welcome page.

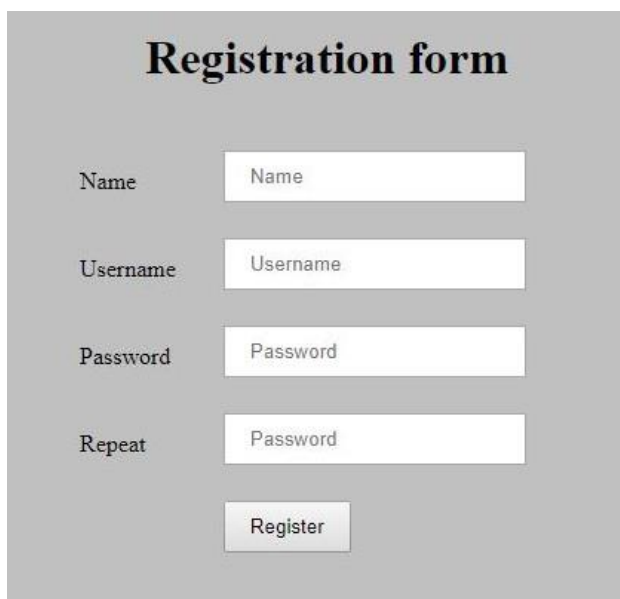
There is also a button to “Register” which redirects the user to the registration page.

A closeup of a login form on a gray background. At the top, the word "Login" is written in a large, bold, black serif font. Below it, there are two input fields. The first is labeled "Username" in a black sans-serif font, and the input field contains the placeholder text "Enter Username". The second is labeled "Password" in a black sans-serif font, and the input field contains the placeholder text "Enter Password". Below the password field is a "Login" button with a gray gradient and black text. At the bottom left, the text "Don't have an account?" is followed by a "Register" button, also with a gray gradient and black text.

*Figure 6 Login page closeup*

### Registration

This page consists of four input fields for name, username, password, and repeated password. This design is consistent with most other registration forms on the internet.

A closeup of a registration form on a gray background. At the top, the words "Registration form" are written in a large, bold, black serif font. Below it, there are four input fields. The first is labeled "Name" in a black sans-serif font, and the input field contains the placeholder text "Name". The second is labeled "Username" in a black sans-serif font, and the input field contains the placeholder text "Username". The third is labeled "Password" in a black sans-serif font, and the input field contains the placeholder text "Password". The fourth is labeled "Repeat" in a black sans-serif font, and the input field contains the placeholder text "Password". Below the "Repeat" field is a "Register" button with a gray gradient and black text.

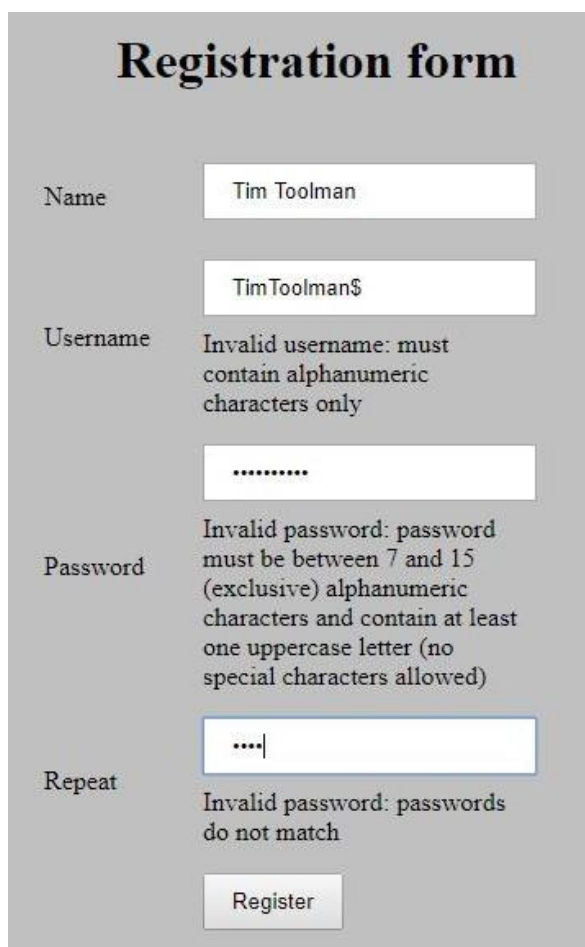
*Figure 7 Registration page closeup*



As the user types in the username field, their input is dynamically checked to be valid. If the username has any non-alphanumeric characters or is the same as an existing username, an alert will be displayed below the input field.

The password and repeated password fields are similar to the username field, except checked on password validation rules. The password must be between 7 and 15 alphanumeric characters, with at least one uppercase letter, or an alert is displayed below this field. The repeated password must match the password.

A “Register” button is used to submit the form. If the user entered invalid inputs, the page will be refreshed and an alert displayed notifying the user of the error. If their inputs were valid, a new user will be generated, their account logged in, and they are redirected to the welcome page.



The image shows a registration form titled "Registration form" on a light gray background. The form contains four input fields with labels to their left. The "Name" field contains "Tim Toolman". The "Username" field contains "TimToolman\$". Below the "Username" field is a red error message: "Invalid username: must contain alphanumeric characters only". The "Password" field contains "\*\*\*\*\*". Below it is a red error message: "Invalid password: password must be between 7 and 15 (exclusive) alphanumeric characters and contain at least one uppercase letter (no special characters allowed)". The "Repeat" field contains "\*\*\*\*|". Below it is a red error message: "Invalid password: passwords do not match". At the bottom of the form is a "Register" button.

Name	<input type="text" value="Tim Toolman"/>
Username	<input type="text" value="TimToolman\$"/> Invalid username: must contain alphanumeric characters only
Password	<input type="password" value="*****"/> Invalid password: password must be between 7 and 15 (exclusive) alphanumeric characters and contain at least one uppercase letter (no special characters allowed)
Repeat	<input type="password" value="**** "/> Invalid password: passwords do not match
<input type="button" value="Register"/>	

Figure 8 Registration page with validation closeup

## Welcome

This page consists of a menu bar and welcome message.

The menu bar design is similar to those on other websites. This menu bar displays three elements: welcome, search, and logout. These, if clicked on, redirect the user to the pages corresponding to their name. The current page, welcome, is highlighted to show the user what page they are on.

When another element is hovered-over with the cursor, that element is highlighted brighter than the current page to show that the user can interact with it.

The content of the page consists of a greeting to the current user, using their name, and instructions on how to use the website.



Figure 9 Welcome page with highlighted menu element

## Search

The search page consists of the menu bar, with the “search” element highlighted, a search input, and a table of products. The table design is consistent with other data tables on the internet.

The table rows are of alternating colour to help the user scan across the columns of a single row. The products displayed in the table are dynamically updated while they type a search term into the search input. Products can be re-ordered by clicking on the column headers. This will update the arrows beside the header text to show the direction of sorting.

SKU	Name	Category	Cost	Stock Qty
HMR000	Claw Hammer	Hammer	\$5.95	21
HMR002	Club Hammer	Hammer	\$6.95	13
HMR001	Ball Pein	Hammer	\$8.95	22
TPE000	Masking Tape	Tape	\$8.95	17
TPE002	Electrical Tape	Tape	\$8.95	32
KNF002	Craft Knife	Knife	\$12.95	21
TPE001	Duct Tape	Tape	\$12.95	32

Figure 10 Search page with search term and sorting on “cost” closeup

## MVC Framework Backend

At the backend, the server uses PHP in an MVC framework. This provides reasonable flexibility and expandability whilst being easy to read and maintain.

## Routing

All traffic is handled by the index.php file in the root web folder. This is the root controller which handles routing and static resources.

Route data is setup for expected URLs to handle expected use cases. The router is then initialised with these routes. If the incoming request is one of the expected use cases, the router will call the necessary controller class defined in the route data setup earlier.

If the incoming request is not routable, it is then checked to be a request for a static resource. If so, and the resource exists, the requested resource is sent back via HTTP.

If the request matches none of the expected use cases and is not for a static resource, the server returns an HTTP packet with the HTTP error 404 to alert the browser.

## Controller

The base “Controller” class is extended from by the “UserController” and “ProductController” classes. This class contains the following:

- redirect action
  - allowing the server to redirect the browser to a given page
- welcome action
  - loading the welcome view if the user is logged in or redirecting to the login page if not
- error action
  - called by child controller classes in the case of an exception
- function to check if a user is logged in
  - which checks session data to verify if a user is logged in

## User controller

The user controller class extends the controller class and manages the user-centric actions:

- login action
  - if user logged in → loads, renders, and echoes welcome view
  - if request is POST → generates a new user model and checks the login details
    - if login valid → starts session and redirects to root/welcome page
    - if login invalid → loads, renders, and echoes the user login view with error message
  - if no user is logged in → loads, renders, and echoes the user login view
- logout action
  - destroys session and redirects to login page
- register action
  - if user logged in → loads, renders, and echoes welcome view
  - if request is POST → checks registration form data
    - if data valid → creates new user model, saves, starts session and redirects to welcome page
    - if data invalid → loads, renders, and echoes registration view with error message
- verify registration form action (called from AJAX POST request on registration page)
  - returns if username exists in database

### *Product controller*

The product controller class extends the controller class and manages the product-centric actions:

- search action
  - if user not logged in → redirects to user login page
  - if user logged in → loads, renders, and echoes product search view
- update search results (called from AJAX GET request on search page)
  - creates product collection model based on GET parameters, JSON encodes each product model in the product collection, and returns the JSON encoded product models

### *Model*

The base “Model” class is extended from by “UserModel”, “ProductModel”, and “CollectionModel” classes. This class sets up the connection with the database and loads sample data if no data exists.

### *User model*

This class extends the model class and is used to interface with the user table in the database.

Properties:

- ID (get; set)
- username (get; set)
- password (set)
- name (get; set)

Functions:

- load
  - loads user data from database from user ID
- save
  - inserts current user model into database
- check login
  - loads user by username
  - returns true if hashed password matches loaded user’s hashed password
- username exists
  - returns true if a username exists in the database

### *Product model*

This class extends the model class and is used to interface with the product table in the database.

Properties:

- ID (get; set)
- SKU (get; set)
- name (get; set)
- cost (get; set)
- cost string (get; set)

- category (get; set)
- stock (get; set)

Functions:

- load
  - loads product data from database from product ID
- expose variables
  - returns an array of all product model class variables

### *Collection model*

This class extends the model class and is used to load a collection of models from the database.

On construction, the collection model is passed parameters to be used to query the database. Those include:

- class
  - specifies what model class the collection objects should be loaded from
- table
  - the table to query
- needle/haystack
  - the needle is a substring used in the WHERE clause to search the haystack column of the table
- order by
  - to order the results
- sort
  - for either ascending or descending sorting

The function “getObjects” is used to get the model objects once the collection model object has been constructed.

### *Product collection model*

This class extends the collection model class and is used to load a collection of product models from the database.

On construction, this class creates and passes the necessary parameters to the parent collection model class to load a collection of product models.

### *View*

The view class loads a PHTML file from the template library so that it can be rendered. This class allows the following:

- construction
  - uses the filename of the desired template to load a PHTML template file
- add data
  - takes a key and value pair which can then be used within the PHTML file to add data to the resulting HTML

- render
  - outputs the formed template HTML DOM

## Templates

The template files are PHTML files used to dynamically create HTML DOMs from the server.

Each main PHTML file consists of multiple components and the HTML specific to its purpose. The components allow code reuse as, for example, the same basic header can be used on multiple pages by including the header component. As these files are PHTML, data can be passed to them dynamically by the server.

## JavaScript/AJAX/jQuery Frontend

At the frontend, the web app uses JavaScript, AJAX, and jQuery to perform client-side input validation and data manipulation. This allows for dynamic and responsive design which the user can interact with easily. Pages don't reload as often, offering a more seamless user experience.

JavaScript forms the backbone of the client-side code. AJAX is used to communicate with the server and jQuery is used to find elements in the DOM.

## Input validation

The registration page uses client-side input validation.

Each input field on the registration page, except for the name field, has business rules that dictate what the user can enter. The inputs are dynamically checked against these business rules as the user modifies them. This allows the page to display warnings to the user if the inputs do not conform to the business rules.

AJAX is used on the registration page to send the inputted username to the server to check that it is unique.

## Dynamic data manipulation

The search page uses client-side data manipulation.

The search results table displays products with name substrings matching the search term input by the user. When the user types in the search input, the URL is updated with the search. AJAX is used to send a GET request to the server for new products matching the search term and sorting. The server then sends back a JSON object containing a collection of product data which is then dynamically loaded into the search results table.

If a user clicks the header of any of the table columns, the triangle images beside the header names are updated to reflect a new sort order. A new AJAX request is sent with the new sort order and search term to update the sort order of the products.

## Database schema and relations

The MySQL database consists of two relations. These relations have no foreign keys and only contain the necessary columns for the user sessions and product searches.

### User

- id - User ID number, primary key
- username - Username
- pwd - User password
- name - User's name

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id 	int(10)		UNSIGNED	No	None		AUTO_INCREMENT
2	username 	varchar(256)	latin1_swedish_ci		No	None		
3	pwd	varchar(256)	latin1_swedish_ci		No	None		
4	name	varchar(256)	latin1_swedish_ci		No	None		

Figure 11 User relation

### Product

- id - Product ID number
- sku - Stop Keeping Unit (SKU) identifier
- name - Product name
- cost - Product cost
- category - Product category
- stock - Stock quantity

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id 	int(10)		UNSIGNED	No	None		AUTO_INCREMENT
2	sku 	char(6)	latin1_swedish_ci		No	None		
3	name	varchar(256)	latin1_swedish_ci		No	None		
4	cost	decimal(19,2)		UNSIGNED	No	None		
5	category 	varchar(40)	latin1_swedish_ci		No	None		
6	stock	mediumint(8)		UNSIGNED	No	None		

Figure 12 Product relation