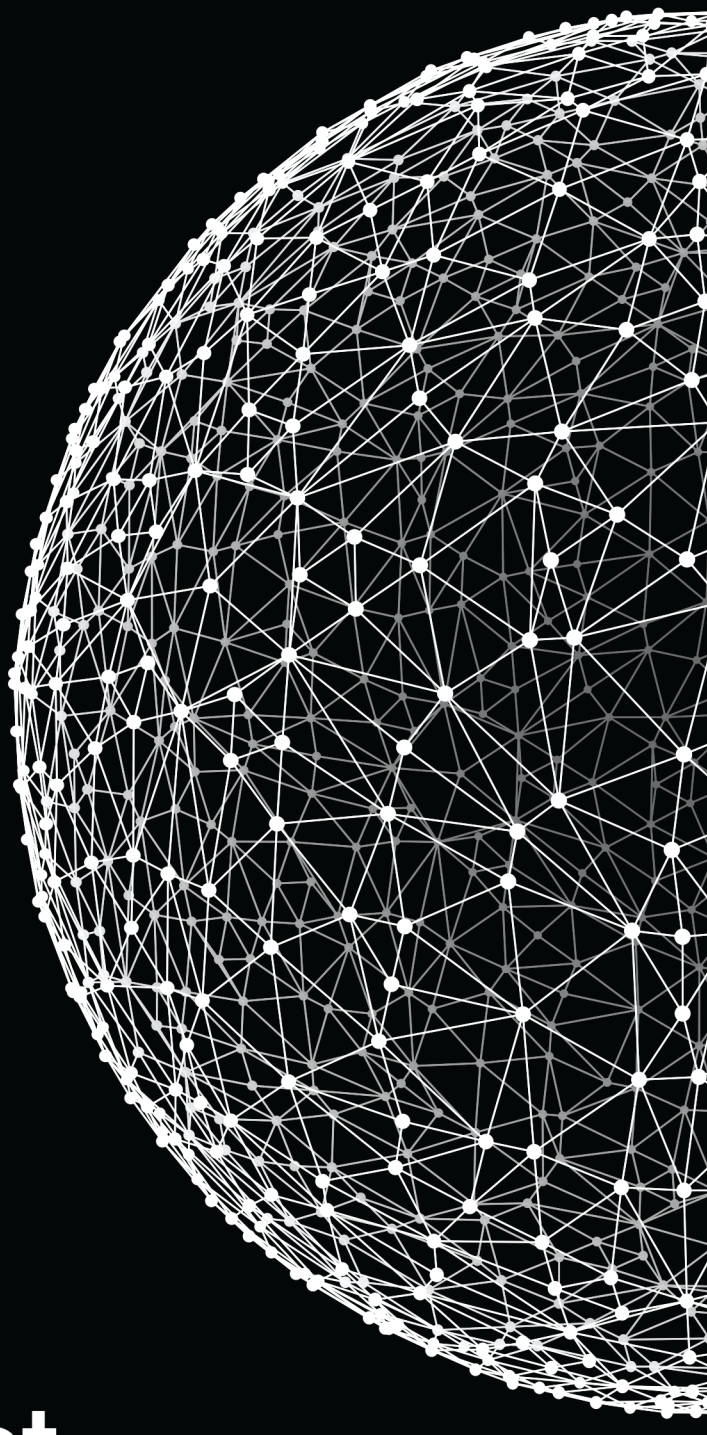
Sprint 09

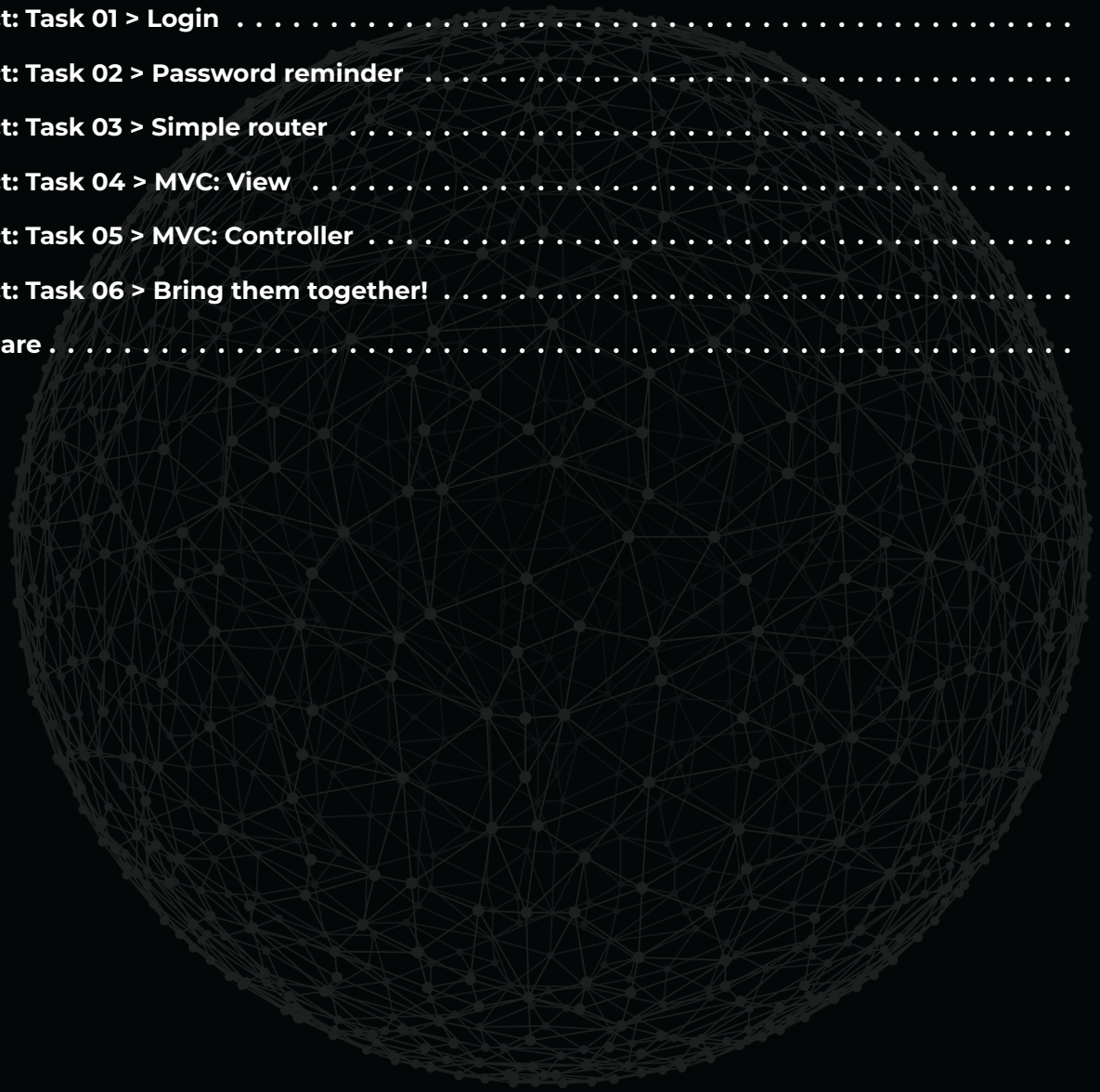
Half Marathon Web

January 13, 2021



ucode connect

Contents



ucode connect

Engage

DESCRIPTION

OH MY GOD!

Can you believe it?! It is the last Sprint in this Half Marathon!
It means that it's time to code like a strong and independent almost-junior developer that you are. Are you ready to meet real tasks? Do you want to know how to make a web service without the help of WordPress?

Today you will get acquainted with software design patterns in PHP. They form the base of backend development. They can be considered best practices for solving common problems which can occur during designing an app or system.

The key aspect of most web services is user interaction. And the most popular pattern for developing user interfaces in PHP is MVC. As it is impossible to be a good web developer without a basic understanding of how the internet works, we advise you to revise what you know about transmission of information packages and the role of the controller and router in this process.

Also, the knowledge of URL structure is obligatory in this Sprint, because you are about to parse the address bar to understand what the user of your web service wants to accomplish.

Make the last effort and you will get to the finish line!
Come on, you can definitely do it!

BIG IDEA

Principles of MVC.

ESSENTIAL QUESTION

How to handle user requests?

CHALLENGE

Create your first web service.

Investigate

GUIDING QUESTIONS

We invite you to find answers to the following questions. By researching and answering them, you will gain the knowledge necessary to complete the challenge. To find answers, ask the students around you and search the internet. We encourage you to ask as many questions as possible. Note down your findings and discuss them with your peers.

- What is a router?
- What is PHP routing?
- What is the difference between static and dynamic routing?
- What is a URL? Describe the URL structure.
- What is a domain name?
- What is a `.htaccess` file? Why do we use it and where?
- What does MVC stand for and what does each component do?
- What are the advantages of using MVC?
- Which part of the MVC pattern does the user interact with?
- What is the difference between `include` and `require` in PHP?
- What are the main error types in PHP and how do they differ?
- How can you enable error reporting in PHP?
- What is a 404 error?
- Which codes of server response do you know?

GUIDING ACTIVITIES

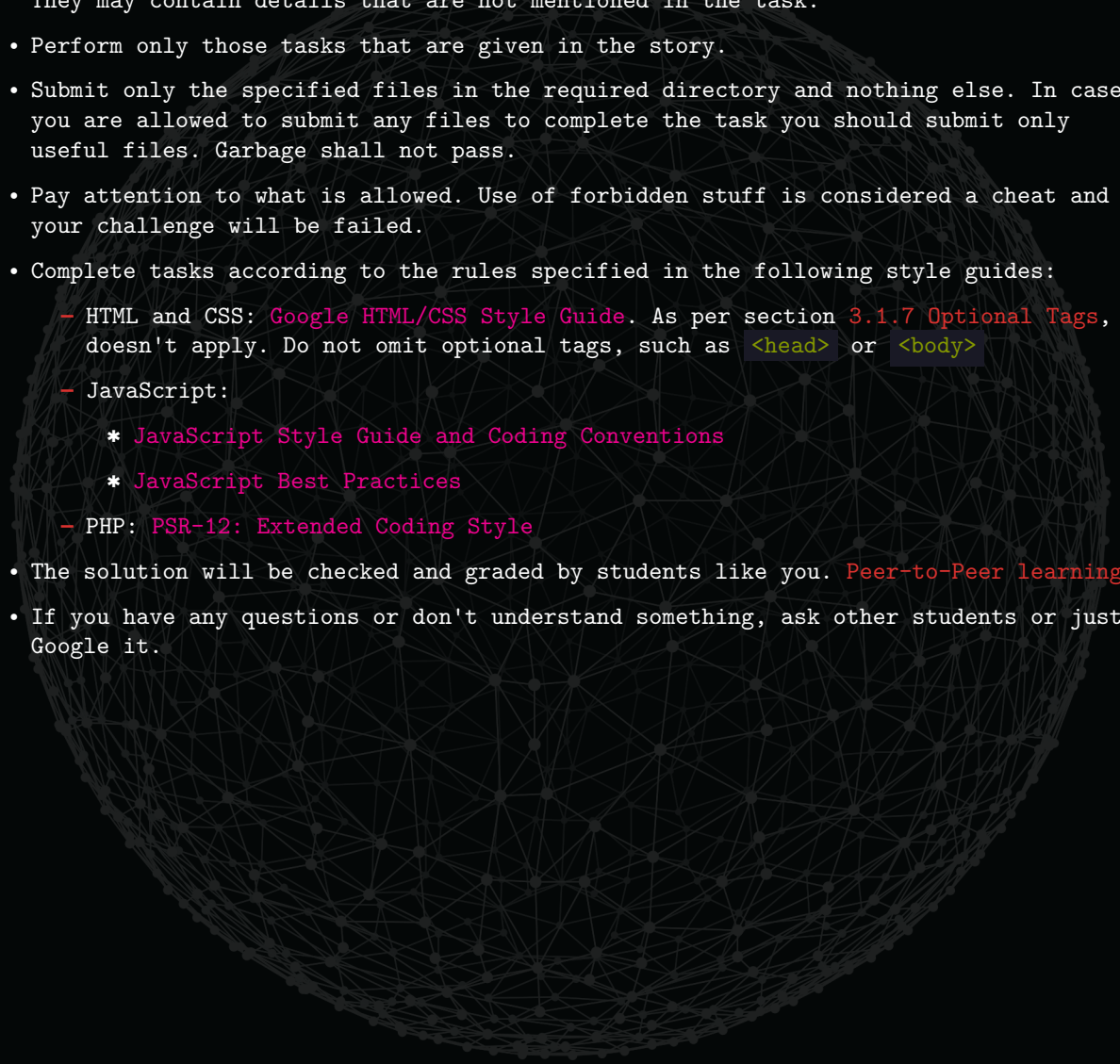
Complete the following activities. Don't forget that you have a limited time to overcome the challenge. Use it wisely. Distribute tasks correctly.

- Remember all the material you have learned during the previous Sprints.
- Discover how to submit a form without a submit button.
- Explore how to check the code of the page response.
- Find out what a perfect 404 page should look like and what it should contain.
- Find information about the `rewrite engine`.
- Think about which PHP functions can help you to parse the address bar.
- Find the most informative classes about MVC and google everything you don't understand.
- Try to explain the MVC pattern to another student so that they understand it.
- Understand the information received, decide what is paramount for you and what is not.
- Clone your git repository issued on the challenge page in the LMS.
- Brainstorm with other students and find more than one solution.
- Try to implement your thoughts in code.

ANALYSIS

Analyze your findings. What conclusions have you made after completing guiding questions and activities? In addition to your thoughts and conclusions, here are some more analysis results.

- Be attentive to all statements of the story. Examine the given examples carefully. They may contain details that are not mentioned in the task.
- Perform only those tasks that are given in the story.
- Submit only the specified files in the required directory and nothing else. In case you are allowed to submit any files to complete the task you should submit only useful files. Garbage shall not pass.
- Pay attention to what is allowed. Use of forbidden stuff is considered a cheat and your challenge will be failed.
- Complete tasks according to the rules specified in the following style guides:
  - HTML and CSS: Google HTML/CSS Style Guide. As per section 3.1.7 Optional Tags, it doesn't apply. Do not omit optional tags, such as `<head>` or `<body>`
  - JavaScript:
    * JavaScript Style Guide and Coding Conventions
    * JavaScript Best Practices
  - PHP: PSR-12: Extended Coding Style
- The solution will be checked and graded by students like you. Peer-to-Peer learning.
- If you have any questions or don't understand something, ask other students or just Google it.



Act: Task 00

NAME

Registration

DIRECTORY

`t00/`

SUBMIT

`db.sql, index.php, connection/*.php, models/*.php, *.html, *.css, *.js`

ALLOWED

PHP, PDO, HTML, CSS, JS

LEGEND

The global automatization process does not step over the S.H.I.E.L.D. A new innovation database named S.W.O.R.D. (Simple Wide-Optimized Relational Database) has been under development for the last three years and now it's time to move forward. Director Fury needs to improve the registration of new agents in the system. Now they should create their profiles by themselves and not waste the time of the slave Director.

DESCRIPTION

Create a simple form to register new users:

- you must create the sword database with a table called users
- the form must contain the following required fields:
  - login
  - password
  - confirm password
  - full name
  - email address
- the fields (except confirm password) must exist in the database as well
- don't forget to handle errors and show a success message if a user was created

You can use the `DatabaseConnector` from the previous Sprint to manage your database. In that case, you should place the `Connector` in the `connection/` folder and all the models in the `model/` folder.

All your database manipulations should be written in 'db.sql' file.

Act: Task 01

NAME

Login

DIRECTORY

`t01/`

SUBMIT

`index.php, connection/*.php, model/*.php, *.html, *.css, *.js`

ALLOWED

`PHP, PDO, HTML, CSS, JS`

LEGEND

The process of registration of new agents is going well, but the agent still can't understand what to do next. Do not let them share rumors but let them get access to their accounts.

DESCRIPTION

Create a simple form to allow login for users:

- the form must contain the following required fields:
  - login
  - password
- a logged-in user must not see the login form, but must see the status: admin or user, which corresponds to the user's field in the database (create this field if you don't have it)
- a corresponding message must appear after a successful or failed login
- all errors must be managed with this process

Make sure the user can log out at any time.

SEE ALSO

`$_SESSION`

Act: Task 02

NAME

Password reminder

DIRECTORY

`t02/`

SUBMIT

`index.php, *.php, connection/*.php, model/*.php, *.html, *.css, *.js`

ALLOWED

PHP, PDO, HTML, CSS, mail, JS

LEGEND

Moving all agents to S.W.O.R.D. is going well. All units have access to the system. Recently, a new agent, Paulson, had missed his target and got a lot of injury from an enemy. The rehabilitation has not taken a lot of time, but the agent had partially lost his memory. Agent Paulson is back to the S.H.I.E.L.D. but cannot access the S.W.O.R.D. system. You need to do something about it.

DESCRIPTION

Create a simple form for the user to get a reminder about their password.
Only registered users can send a request to get a password reminder. You should get a password and send it to the user's email.

Remember! It's not secure to send passwords by email, so don't do it in real life!

SEE ALSO

mail

Act: Task 03

NAME

Simple router

DIRECTORY

`t03/`

SUBMIT

`index.php, Router.php, .htaccess`

ALLOWED

PHP

LEGEND

S.W.O.R.D. shows itself as a great system. No one knows about its goals and functions, but everyone can register and login. What a success! Now you get a really great challenge. The S.W.O.R.D. routing system needs to be improved with MVC, and your mission is to find the way to implement it.

DESCRIPTION

It's time to begin the creation of your first MVC project. Didn't hear about it? Let's start with a simple router to manage all the requests on the site.

Create a class `Router`. This class must manage all the requests on your site.
The main goal of this class is to print a path passed in a URL string without a domain name in the constructor. Your router must be initialized like `$router = new Router();`

Your class must have a public `$params` parameter. `$params` must contain all query parameters from the URL string.
For example, after executing a URL like `/test/?data=somedata&filter=somefilter`
`$params` must looks like `array('data' => 'somedata', 'filter' => 'somefilter')`

Print `$params` too. Like this: `{'data': 'somedata', 'filter': 'somefilter'}`

By the way, from now the entry point of your site must be `index.php`. This means that every request will be passed through 'index.php', and it's not allowed to print any data in this file. Find out how to make your 'index.php' file call the Router.

SEE ALSO

MVC
.htaccess

Act: Task 04

NAME

MVC: View

DIRECTORY

`t04/`

SUBMIT

`view/View.php`

ALLOWED

PHP

LEGEND

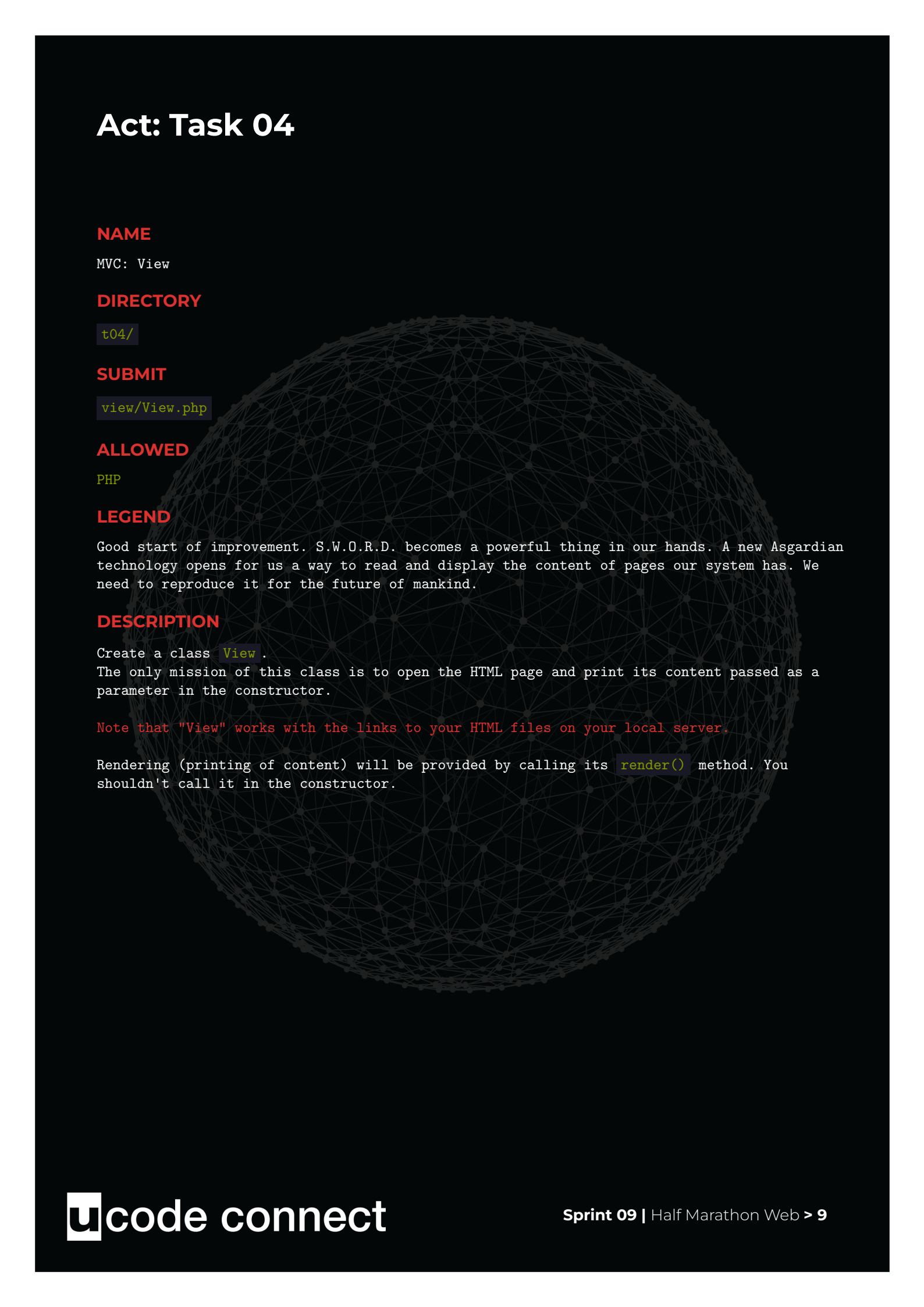
Good start of improvement. S.W.O.R.D. becomes a powerful thing in our hands. A new Asgardian technology opens for us a way to read and display the content of pages our system has. We need to reproduce it for the future of mankind.

DESCRIPTION

Create a class `View`.
The only mission of this class is to open the HTML page and print its content passed as a parameter in the constructor.

Note that "View" works with the links to your HTML files on your local server.

Rendering (printing of content) will be provided by calling its `render()` method. You shouldn't call it in the constructor.



Act: Task 05

NAME

MVC: Controller

DIRECTORY

`t05/`

SUBMIT

`controller/Main.php, view/View.php, view/templates/main.html`

ALLOWED

PHP, HTML

LEGEND

Control...
This is what Tony is fighting against. This is the way to restrain chaos and the reason for the separation of the Avengers. But we have no choice, S.W.O.R.D. should be able to manage all the routes through controllers. We hope it will not be a start of a new Civil War.

DESCRIPTION

Create a simple 'html' page named 'main.html' and save it in `view/templates/` path. It must contain this code:
`<html><div>Hello, world!</div></html>`

Create a class `Main` that implements `ControllerInterface`. Your class must have a `$view` property that is initialized in the constructor. `$view` is an instance of the `View` class you created before.

The `execute()` method calls the `render()` method of `$view`. As a result, the page 'main.html' must be displayed.

Act: Task 06

NAME

Bring them together!

DIRECTORY

`t06/`

SUBMIT

`model/*.php, view/*.php, controller/*.php, *.*`

ALLOWED

`PHP, PDO, HTML, CSS, JS, mail`

LEGEND

Nick Fury is in a fury! S.W.O.R.D. had to become a great weapon to fight all unautomated processes, but for now it's just a bunch of useless stuff and no one can understand the purpose of each part.
We have some secret information from agent Paulson from Wakanda. It contains notes about how to join together all the stuff we've made before and can explain a lot of things. Use it to join all the parts and run the system.

DESCRIPTION

It's time to join everything you created today in one beautiful MVC app.
Compose all the files together and create a powerful app.
The main requirements are:

- the router must call the corresponding controller class depending on the URL passed in the address string
- the 404 page must be displayed if the controller does not exist
- store all your models in the `model/` folder
- each model must be named after the corresponding table in the database
- you must check if the user logged in before displaying a page for them
- if the user is not logged in, the user must see only the login page
- your controllers must be available to make the changes in the database if the user wants to do so

There are some required pages that you must implement:

- registration page
- login page
- password reminder page
- main page

Share

PUBLISHING

Last but not least, the final stage of your work is to publish it. This allows you to share your challenges, solutions, and reflections with local and global audiences. During this stage, you will discover ways of getting external evaluation and feedback on your work. As a result, you will get the most out of the challenge, and get a better understanding of both your achievements and missteps.

To share your work, you can create:

- a text post, as a summary of your reflection
- charts, infographics or other ways to visualize your information
- a video, either of your work, or a reflection video
- an audio podcast. Record a story about your experience
- a photo report with a small post

Helpful tools:

- Canva - a good way to visualize your data
- QuickTime - an easy way to capture your screen, record video or audio

Examples of ways to share your experience:

- Facebook - create and share a post that will inspire your friends
- YouTube - upload an exciting video
- GitHub - share and describe your solution
- Telegraph - create a post that you can easily share on Telegram
- Instagram - share photos and stories from ucode. Don't forget to tag us :)

Share what you've learned and accomplished with your local community and the world. Use #ucode and #CBLWorld on social media.