PWS 236369 - HW1 [WIP]

Winter 22\23

1. General Notes

- Release Date - 31.10.22
- Submission Deadline - 15.11.22 at 23:50
- Submission in pairs only.
- Assistance with the assignment will be given by workshops and course forum on Moodle only.
- Files supplied by the staff can be found on this repository - pws236369 / HW1

2. Main Goal

In this assignment you will build an interactive web page with HTML, CSS, and Javascript. This is a “front-end only” task. No server calls, only a static web page.

3. Setup

For recommended development environment setup, please refer to *Appendix 1 - Setup, Prepare & Deploy*, at the end of this document.

4. Background story

We want to create a “pricing page” for the new gaming company, “TeamTaub”.

TeamTaub gives its users great game streaming services and lots of cool options and features. Our goal is to sell those plans - and match each user with the package that suits them.

In a nutshell, your pricing page is the landing page on your website that lists the different pricing options or tiers available to customers and the benefits and features included in each tier. (For pricing page examples, please refer to *Appendix 2 - Examples*).

5. User Story [70 points]

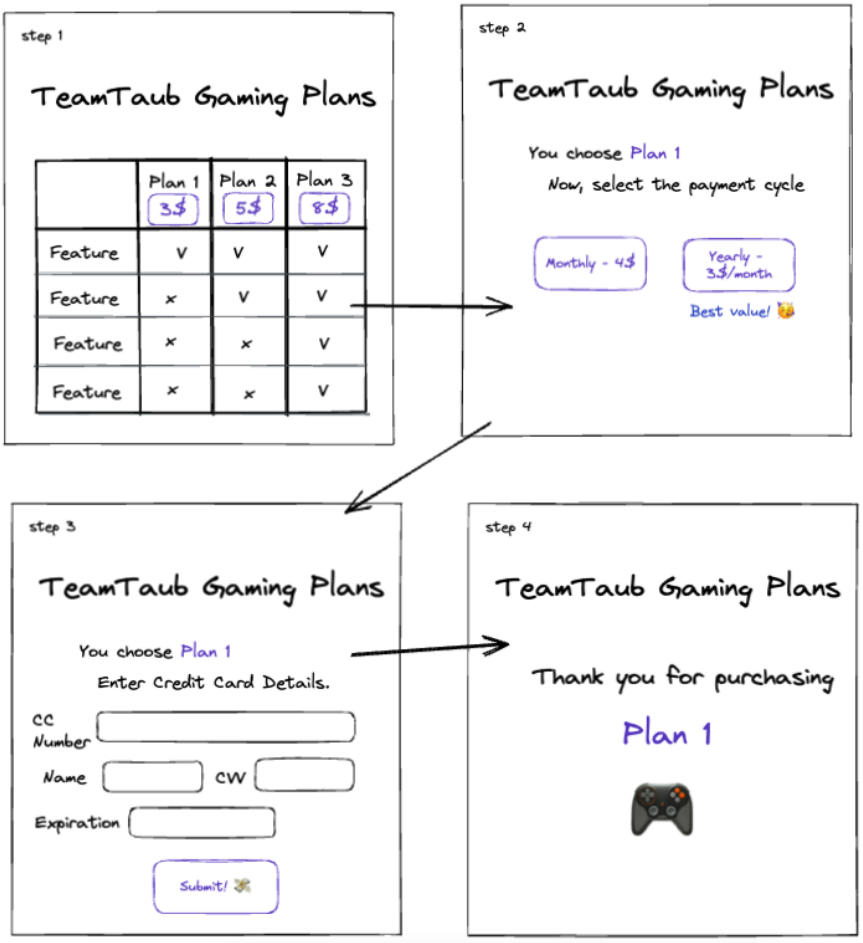
First, read about user story.

Our webpage is going to have 4 steps, each one has its own page. Our goal is to make the user choose a package, a payment cycle, and pay with the checkout form. The experience should be nice and clear so that we promote the user in the purchase flow.

The 4 steps of the purchase process are as follows:

	User Story	Details	Points
1	**Main pricing page** Choose a package.	• A table that will show all tiers (different packages). • Each package will have a name, price, and features. • The price shown in this page will be the yearly price.	50
2	**Cycle page** Choice of payment cycle - either monthly or yearly	• The user will have the option to choose between a monthly and annual payment. • We will tell the user that choosing an annual payment is cheaper	5
3	**Checkout form**	• Form for inserting credit card information	20
4	**“Thank you” page**	• Show the user a “Thank you” message, and a summary of the purchase.	5

In general, the purchase flow should look like this:



5.1. User Story Notes

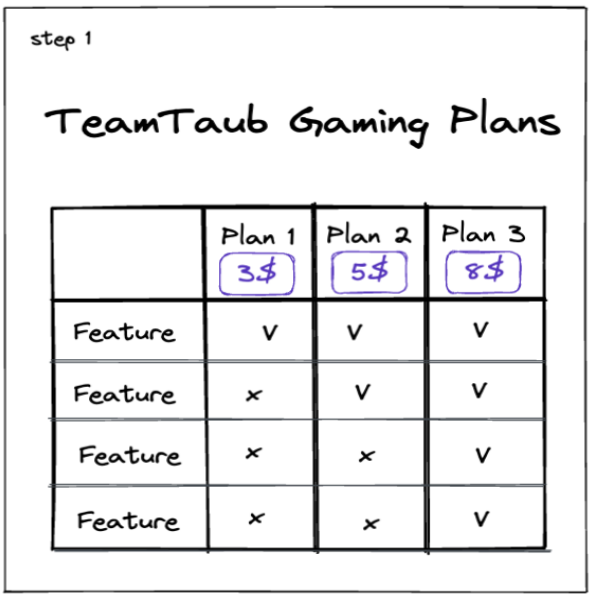
- Stick to the mock page that we give you here - it is a good skeleton for your site and the bare minimum you must implement in this assignment. However, feel free to get inspired by the examples in the appendix.
- Each page must have a title and step number.
- You don’t need to implement “go back” functionality on each page.

5.2. Technical hints & tips

- Start with the HTML markup for each page and use the best HTML tags for each element.
- After that, try to add some Javascript - first with simple tags, to navigate between pages.
- Think how to model the project - HTML, CSS & JS files.
- Think about how you should pass data between pages and how to change them. There might be more than one way to achieve that.

6. Website Stages

6.1. Main pricing page



The main page of our website. In this page, the user will choose the right subscription plan for him. Different plans will be displayed in a table, so our user will be able to compare plans and see which features are included in each plan. In the top row, we will present the name of the plan and its price on an annual basis. Clicking on the price will lead to the next step.

Notes:

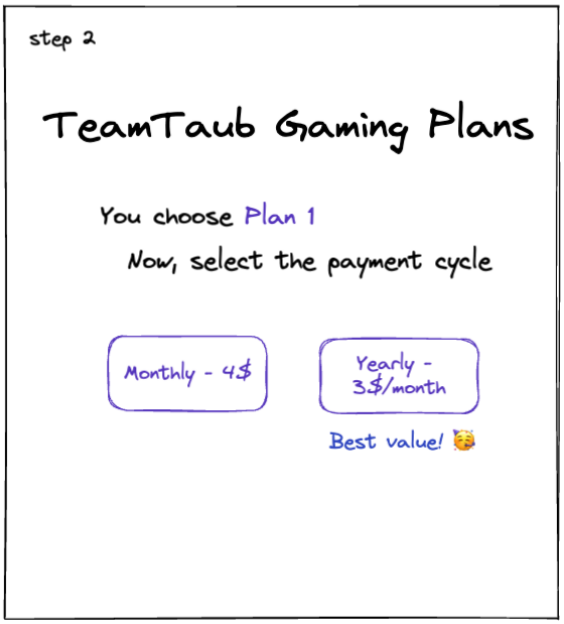
- The design should look good. Table must be aligned properly with the same margins for each column & row.
- Information and details for each plan must be loaded from a JS file (as if it is fetched from a server). A data file (data.js) containing all plan details is supplied through the assignment's repository. You must not create the table only with html tags, usage of JS to generate the table is required.
- Feature rows of the table will be - Number of Games, TaubCoins, Special Avatar(yes\no), Cool T-Shirt Included(yes\no), Customer Care
- Try using CSS grid & CSS flexbox, you might find it useful for viewing the table.

6.2. Billing Cycle page

The user will have to choose between two billing options - annual or monthly. We will show the “best value” badge for the annual cycle to note that it is cheaper than the other.

Notes:

- Try using CSS flexbox, you might find it helpful.



6.3. Checkout form

The user will have to insert credit card info to make his purchase. Payment method info includes card number, holder's name, expiration date and CVV (3 digits on the back of the card).

Credit card info will be validated for the following constraints:

- CC Number - Exactly 16 digits.
- Name - consists of English-only characters.
- Expiration Date - MM/YY format. Only future dates are valid.
- CVV - 3 digits only.
- All fields are required (not empty).

If one or more validations fail, You must give an indication of failure, and not proceed to the next step. The user will be able to correct its info in case of a submission failure.

Notes:

- This is a mock checkout! We don't send any real data anywhere. When coding, don't use any real CC information.
- Try using the html form tag - <form>: The Form element.

Fun fact

- About CC validations - We are not really implementing CC validations - read here about a part of real validation. There are a lot of operations that are performed - the majority on the server side. Try to think about what actions need to be performed before requesting a transaction transfer 🤓

6.4. Thank You page

After submitting payment information (in case the CC info is valid) we will show the user a “Thank you” page with information about his purchase. The thank you note should include the plan and billing cycle chosen by the user, in addition to the amount that the user will be charged for (depending on the plan and billing cycle).

7. **Coding & Tech [25 points]**

Your submission will be graded for code quality (as well as for other parameters) as follows:

- Structure [15 points] - You must separate your modules and logic in a sensible manner. Different modules should be separated into different files or directories.
- General Programming Principles [10 points] - Your code must be readable, short and smart (please refer to chapters 4-6 of this blog post)
  Important Note - In this assignment we won't check your different git commits. You will be checked only for your last commit on master branch and the deployed website (more on submission in chapter 9 - Submission & Grading)

8. **UI & Design [5 points]**

As most of this course students are not designers, you are not expected to create stunning designs. However, we do expect your website to offer a comfortable, simple user experience that doesn't break. Make sure there is a constant margin (especially in the table), and clear and appropriate font & text (not too big nor too small).
Try to check out the examples in the second appendix to get inspired.

9. Submission & Grading

User Story	Coding & Tech	UI	
70	25	5	100

9.1. Grading

In each of the assignments parts you will be graded as follows:

- User story grade is for the core functionality - Did you implement what was expected of you? Are there bugs? Does everything work as expected? Think about edge cases and where things can go wrong.
- Coding & Tech grade is for best practices, as we learn in lectures. We also expect you to stick to the principles of basic coding concepts (like variable names, split code to functions and models).
- UI grade is for a minimal comfortable user experience.

Note: You can consult us in the course forum, as well as in the workshops. At the end of the exercise, you will receive the list of the exact parameters on which you were measured and graded with personal feedback.

9.2. Submission

Submission of the assignment consists of three parts:

- GitHub Repository - You must create a git repository containing your submission and upload it to GitHub as a private repository. In order for us to check your source files, you are required to add the staff's account pws236369 as a collaborator (via the repository settings).
- Deployed Site - Your website must be deployed, i.e. available via URL. This can be achieved by using GitHub Pages (more on deployment in *Appendix 1 - Setup, Prepare & Deploy*)
- Moodle - In the class website you will find an assignment item called "Homework 1". In the text field of this assignment you are required to fill in the following:
    - URL of your private GitHub repo, with permission to the course user.
    - URL for the static web page (GitHub Pages deployment).
    - IDs and Names of both you and your partner for this submission.

Important Note - The following conditions will cause an automatic failure of the assignment:

- Pushing commits to the GitHub repository after the submission deadline.
- Making the GitHub repository public instead of private.
- The website is not deployed or the URL submitted is not valid.

Appendix 1 - Setup, Prepare & Deploy

1. Work Environment :

1.1. Visual Studio Code

In order to write your solutions to this assignment (and all other assignments throughout this class), we recommend using Visual Studio Code. Visual Studio Code (or VS Code) is a free to use cross platform IDE with a built in debugger, git support and many more useful features, and it is very well suited for web development.

To install VS Code please refer to - Setting up Visual Studio Code.

Important Note (!) - VS Code will be the **only** IDE or code editor the staff will offer support for in the workshops and the class forum. You are free to use whatever development tools you wish, however we won't be able to offer any help if needed regarding these tools.

1.2. Live Server

Live Server is an extension for VS Code that enables viewing your website while developing. When Live Server is on and your website is open on browser, it will be re-rendered whenever you make changes to your files.

To install Live Server please refer to the VS Code extensions marketplace - Live Server

2. Deployment

2.1. GitHub Pages

As all future assignments, your solution is required to be deployed (i.e. - hosted and reachable via URL). In this assignment there is no server logic required, so a good solution for hosting our static site is GitHub Pages.

GitHub Pages is a hosting platform that lets us turn out GitHub repositories into static websites. You can create a repository especially for deploying a new website or configure a deployment of an existing repository.

To learn how to deploy your solution using GitHub Pages - Creating a GitHub Pages Site

Appendix 2 - Examples

Here are few examples of pricing pages you can get inspired from:

GitHub

Atlassian - Jira Software

Monday

Connecteam