Project 2 -Wordle!

**Wordle, the Word Game**

JavaScript, React and State Management

**Overview**

For this project, you will recreate a popular single player word game called Wordle. An example of Wordle can be seen here

<https://www.nytimes.com/games/wordle/index.html>

,and rules will be discussed below. When users enter your app, they should see a welcome page that contains nothing but the title of the game and links to a “rules” page and the main “game” page. In this game, users have multiple chances to figure out a word and can use hints from previous guesses to determine the correct answer.

For this assignment, you may work with one additional student. You may post on the discussion board if you are looking for a partner. You are of course welcome to tackle this work by yourself as well.

Please note: you are welcome to come up with your own app idea, but you must contact the teaching staff a week beforehand to ensure that the challenge is sufficiently worthwhile and can reasonably meet the expectations of the rubric below.

**Rubric**

Core Functionality - 25%

Working Github and Working Website Link - 5%

Correct Views and Good Styling - 15%

Well Written Code - 15%

Stores, Actions and Reducers - 15%

Demonstrating Proper React Principles - 15%

Writeup - 10%

Bonus Points

**Core Functionality**

Wordle is a word game that challenges to figure out a word within only a certain number of guesses. You can try the game out for yourself here! <https://www.nytimes.com/games/wordle/index.html>

Please keep in mind that the style and functionality is a bit different.

When a user opens up your app, they should see a title screen with links to play the game and a link to the game rules. These pages are described in more detail below.

In Wordle, the game secretly chooses a random word that the user will try to guess within a certain number of attempts. Both the length of the word and the number of attempts are based on the difficulty selected by the user (see below.) You need to store at least 10 words for each difficulty level so that each game is different each time. On the game page, users should have a prompt to input a word that is the length defined by the difficulty level.

If the user inputs the correct word, you should display a congratulations at the top of the screen. If they chose an incorrect word, you will give them clues based on the location of the letters in the word by highlighting certain letters. For instance, say the correct word is “faces” but the user submits “eats”, then you show “EATER”. In this situation, the one E is in the word but not in the correct spot, as indicated by the yellow; the A is in the word AND in the correct spot, so this is marked by the green. Notice that the second ‘E’ is marked in gray like the other missing letters: since E only appears once in the word, we will only highlight it once (if it were to appear twice, both would be highlighted.)

If the word is too short or long, you should show a message at the top but will not deduct any attempts. Here is an example flow:

Game secretly selects the word “Fellow”. The screen prompts the user to input a 6 letter word and shows that they have 6 attempts remaining.

User submits the word “hat.”

Since “hat” is too short, the game prompts a longer word and does not deduct any attempts.

The user submits in “Allows”.

The game shows “ALLOWS” and since this is not the current word it shows the user has 1 less attempt.

The user submits “Fellow”.

The game shows “Congratulations! Would you like to try again?

The page should also show a button that allows users to reset the game and try a different word.

When the user comes to your landing page, they should be able to select one of 2 difficulties for this game. This should redirect them to a specific URL so that users The difficulties are as follows:

In a normal game, the user will have to find 6 letter words and be given only 6 opportunities. This should be found at url http://<domainname>.com/game/normal.

A hard game will use 7 letter words and only have 5 opportunities to answer. This should be found at url http://<domainname>.com/game/hard.

The difficulty should be handled in the URL, so if a user refreshes the page or shares the link with a friend, that same difficulty will be selected.

**Working Github and Server Link**

For this assignment, you are welcome to use any web hosting service you are comfortable with, but I recommend using Render. Note that many of these are paid, so if you decide to spend money know that you need to keep your app up only for a month or so and check to see if they offer any student accounts. Please follow the instructions below or contact any of the teaching staff to get this set up if you need help. Please be sure to add the TA’s as collaborators to your Github repo.

I wrote up some instructions here

<https://docs.google.com/document/u/0/d/14rjMRGlsj6YM5YoQGdQihwE0qzEQCq76TSQLE43C1AQ/mobilebasic>

**Correct Views and Good Styling**

There are at least 3 views to implement for this assignment:

A home page that allows users to select the difficulty of the game

A page that allows the user to play the game

A page with the rules for the game

You should decide what these views look like and how they behave (are they entirely different pages? Will you swap out components?). Feel free to intermingle these, but justify your decision in your writeup.

Additionally, you should have a unique and consistent style across the different pages. While the style is relatively simple here, consider adding style to buttons, on hover styling over input so that the cursor turns to a pointer, fonts, background images, etc. The screen should also present well on both mobile and desktop browsers. You should have some sort of navbar or navigation aid that allows users to go between these different views.

You are welcome to use any 3rd party styling libraries, such as Tailwind, React Bootstrap, Material UI, etc.

Finally, if these views are on different pages, consider sensical and good URL design.

**Well Written Code**

Now that we’re writing logic, you must start considering the quality of the code you’re writing. Functions should be simple, easy to read and avoid repetition. React components should be small, reusable and have a single purpose. We are not expecting you to use any more advanced JavaScript functionality, but you should be writing code that you would be happy to show to a potential employer.

Finally, every use of the “alert” function will result in the loss of one point.

**State Management**

For this assignment, consider how and where you are storing information. In web development, there are local and global states:

Local state is the data stored in a single component (this might be what kind of card to display, etc.)

Global state is all the data related to the entirety of the app (what words have been written, what the current input is, etc.)

For global state, you are expected to use some kind of state management technology (useContext, Redux, or other if you speak with the teaching staff first). Make sure that there are at least 2 state update actions with your state management tool.

Demonstrating Proper React Principles

With the introduction of state, we learned about the unidirectional flow of information and how to separate the views from the data. In this assignment you must:

Have at least 4 different React components

One of those components must receive props from a parent component

You must have nested components

Some data must be passed from the child to the parent, but you should NOT do this with a function. Rather, use your state management tool

**Bonus Points**

There are some opportunities to earn extra points for this project if you go above and beyond. Remember that every ten points you earn is 1 point increase for your final grade. Attempts for each of these are worth partial credit.

**Early Submission - 3pts**

If you submit this project 48 hours before the due time (start of class), you gain 3 bonus points.

**Saved Data - 4pts**

Sometimes users have a busy life and are not able to finish the puzzle before they find the solution. If a user closes their browser or tab, you should have additional functionality that “remembers” where they are as if they never closed the screen at all.

**Valid Word Check - 2pts**

To make the game a bit more difficult, the game should check that a valid English word is used. Every time users submit a word, it should be checked that it is valid English and if not, the user cannot submit and a warning should be displayed on the page to ask them to submit a new word. You should be able to ensure that at least 1000 words exist for each difficulty.

**Deliverables**

Include all the following in your submission on Canvas:

A link to your Github repo. If you are working with a partner, you may submit a link to the same repo (for grading purposes, the TA’s will likely only look at a single repo so make sure they are identical.) Please note that your Github repo should be named: {firstname}-{lastname}-project2, and if you’re working with a second person both names should appear on the repo.

A link to your app hosted on a web service (such as Firebase, Heroku, Render, AWS, etc.) As above, if you are working with a collaborator, please submit the same link.

Your writeup. If you are working with a partner, you may each write this together or individually, but please indicate this with your submission.

The name of your collaborator, if any

Indicate what bonus points you attempted, if any.

翻译-项目 2 - Wordle！

**Wordle，文字游戏**

JavaScript、React 和状态管理

**概述**

对于这个项目，您将重新创建一个名为 Wordle 的流行单人文字游戏。可以在这里看到 Wordle 的示例<https://www.nytimes.com/games/wordle/index.html>，规则将在下面讨论。当用户进入您的应用程序时，他们应该会看到一个欢迎页面，其中仅包含游戏名称以及指向“规则”页面和“游戏”主页面的链接。在这个游戏中，用户有多次机会找出一个单词，并可以使用之前猜测的提示来确定正确答案。

对于这项作业，您可以与一名额外的学生一起工作。如果您正在寻找合作伙伴，您可以在讨论区发帖。当然也欢迎您自己完成这项工作。

请注意：欢迎您提出自己的应用创意，但您必须提前一周联系教学人员，以确保挑战足够值得，并且能够合理地满足以下评分标准的期望。

**说明**

核心功能 - 25%

工作 Github 和工作网站链接 - 5%

正确的观点和良好的造型 - 15%

编写良好的代码 - 15%

Stores、Actions 和 Reducers - 15%

展示正确的 React 原则 - 15%

写作 - 10%

奖励积分

**核心功能**

Wordle 是一种文字游戏，挑战仅在一定数量的猜测中找出一个单词。您可以在这里亲自尝试游戏！<https://www.nytimes.com/games/wordle/index.html>请记住，样式和功能略有不同。

当用户打开您的应用程序时，他们应该会看到一个标题屏幕，其中包含玩游戏的链接和游戏规则的链接。这些页面在下面有更详细的描述。

在 Wordle 中，游戏秘密地选择一个随机单词，用户将在一定的尝试次数内尝试猜测。单词的长度和尝试次数都取决于用户选择的难度（见下文。）您需要为每个难度级别至少存储 10 个单词，以便每次游戏都不同。在游戏页面上，应该提示用户输入一个由难度级别定义的长度的单词。

如果用户输入了正确的单词，您应该在屏幕顶部显示祝贺。如果他们选择了一个不正确的单词，您将通过突出显示某些字母，根据字母在单词中的位置为他们提供线索。例如，假设正确的词是“faces”，但用户提交的是“eats”，那么你会显示“ E A TER ” 。在这种情况下，一个 E 在单词中但不在正确的位置，如黄色所示；A 在单词 AND 中的正确位置，因此用绿色标记。请注意，第二个“E”像其他缺失的字母一样标记为灰色：由于 E 在单词中只出现一次，我们将只突出显示它一次（如果它出现两次，则两者都会突出显示。）

如果单词太短或太长，您应该在顶部显示一条消息，但不会扣除任何尝试。这是一个示例流程：

游戏暗中选择了“Fellow”这个词。屏幕提示用户输入 6 个字母的单词并显示他们还有 6 次尝试。

用户提交单词“帽子”。

由于“帽子”太短，游戏提示一个较长的单词并且不扣除任何尝试。

用户在“允许”中提交。

游戏显示“ A L L OW S ”，因为这不是当前单词，它表明用户少了 1 次尝试。

用户提交“Fellow”。

游戏显示“恭喜！你想再试一次吗？

该页面还应显示一个按钮，允许用户重置游戏并尝试不同的单词。

当用户来到您的着陆页时，他们应该能够为该游戏选择 2 个难度之一。这应该将它们重定向到特定的 URL，以便用户 困难如下：

在正常游戏中，用户必须找到 6 个字母的单词，并且只有 6 次机会。这应该可以在 url **http://<domainname>.com/game/normal**找到。

困难游戏将使用 7 个字母的单词，并且只有 5 次回答机会。这应该可以在 url **http://<domainname>.com/game/hard**找到。

难度应在 URL 中处理，因此如果用户刷新页面或与朋友分享链接，将选择相同的难度。

**工作 Github 和服务器链接**

对于此作业，欢迎您使用您熟悉的任何网络托管服务，但我建议使用 Render。请注意，其中许多是付费的，因此如果您决定花钱，请知道您只需要将您的应用程序保持一个月左右，并查看他们是否提供任何学生帐户。如果您需要帮助，请按照以下说明或联系任何教职员工进行设置。请务必将助教作为合作者添加到您的 Github 存储库中。

我在这里写了一些说明

<https://docs.google.com/document/u/0/d/14rjMRGlsj6YM5YoQGdQihwE0qzEQCq76TSQLE43C1AQ/mobilebasic>

**正确的观点和良好的造型**

这个任务至少有 3 个视图要实现：

允许用户选择游戏难度的主页

允许用户玩游戏的页面

包含游戏规则的页面

您应该决定这些视图的外观以及它们的行为方式（它们是完全不同的页面吗？您会更换组件吗？）。随意混合这些，但在你的文章中证明你的决定是正确的。

此外，您应该在不同页面上拥有独特且一致的风格。虽然这里的样式相对简单，但可以考虑为按钮添加样式，在输入上悬停样式，使光标变为指针、字体、背景图像等。屏幕也应该在移动和桌面浏览器上显示良好。您应该有某种导航栏或导航辅助工具，允许用户在这些不同的视图之间切换。

欢迎您使用任何第 3 方样式库，例如 Tailwind、React Bootstrap、Material UI 等。

最后，如果这些视图位于不同的页面上，请考虑合理且良好的 URL 设计。

**写得好的代码**

现在我们正在编写逻辑，您必须开始考虑您正在编写的代码的质量。功能应该简单，易于阅读，避免重复。React 组件应该是小的、可重用的并且有一个单一的目的。我们不希望您使用任何更高级的 JavaScript 功能，但您应该编写您乐意向潜在雇主展示的代码。

最后，每次使用“提醒”功能都会扣一分。

**状态管理**

对于此作业，请考虑存储信息的方式和位置。在 web 开发中，有局部状态和全局状态：

本地状态是存储在单个组件中的数据（这可能是显示哪种卡片等）

全局状态是与整个应用程序相关的所有数据（写了什么字，当前输入是什么，等等）

对于全局状态，您应该使用某种状态管理技术（useContext、Redux 或其他，如果您先与教学人员交谈）。确保状态管理工具至少有 2 个状态更新操作。

**展示正确的 React 原则**

随着状态的引入，我们了解了信息的单向流动以及如何将视图与数据分开。在此作业中，您必须：

至少有 4 个不同的 React 组件

这些组件之一必须从父组件接收道具

你必须有嵌套的组件

一些数据必须从子级传递给父级，但你不应该用函数来做这件事。相反，使用你的状态管理工具

**奖励积分**

如果您超越自我，就有机会为该项目赚取额外积分。请记住，您每获得 10 分，您的期末成绩就会增加 1 分。对这些中的每一个的尝试都值得部分信任。

**提前提交 - 3 分**

如果您在截止时间（开课）前 48 小时提交此项目，您将获得 3 分加分。

**保存的数据 - 4pts**

有时用户生活忙碌，无法在找到解决方案之前完成拼图。如果用户关闭了他们的浏览器或选项卡，你应该有额外的功能来“记住”他们所在的位置，就好像他们根本没有关闭过屏幕一样。

**有效单词检查 - 2pts**

为了让游戏更难一点，游戏应该检查是否使用了有效的英文单词。用户每次提交一个词，都应该检查它是有效的英文，如果不是，用户不能提交，页面上应该显示一个警告，要求他们提交一个新词。您应该能够确保每个难度至少有 1000 个单词。

**可交付成果**

在 Canvas 上提交的内容中包括以下所有内容：

指向您的 Github 存储库的链接。如果您正在与合作伙伴合作，您可以提交指向同一存储库的链接（出于评分目的，助教可能只会查看一个存储库，因此请确保它们是相同的。）请注意，您的 Github 存储库应命名为： {firstname}-{lastname}-project2，如果你和第二个人一起工作，两个名字都应该出现在 repo 上。

指向托管在 Web 服务（例如 Firebase、Heroku、Render、AWS 等）上的应用程序的链接。如上所述，如果您与合作者合作，请提交相同的链接。

你的文章。如果您与合作伙伴一起工作，你们可以一起或单独写这篇文章，但请在提交时注明。

您的合作者的姓名（如果有）

指出您尝试了哪些奖励积分（如果有）。