Capstone Project–Capital Two

Technique Specification Discussion Manual

Shaokang Jiang
sjiang97@wisc.edu

October 3, 2020

Contents

1 General idea

This is a draft edition, we should still discuss any improvement

This document may not be share to other people.

This document only describe the technique specification of the web track.

The library and methods mentioned in the document is nethier final decision nor best possible solution. All information is just a brief intro. For each part, there might be other better methods to use. We can still discuss them.

Some codes in this document are not tested

1.1 Some basic agreement

- The entire project would be a nearly serverless application, except some fundamental database part
- Consistent of design: The theme for the entire project is using Material Design.
- All quizzes come from a question bank. So less collision would happen.
- User Login credential will be saved in local cache place temporary.
- Majority of repeating code works should be done by programs

1.2 Tools recommend

The percentage stands for the percentage of work required. Following percetage is measured base on my experience. Written part is a little bit heavy as I do not like writting.

Markdown(30%) This might be the core part of the entire project. It contains the knowledge passage(3 points*10 topics=30 passages) we are going to provide to user. This part might also include some part of quiz questions design and the words in the homepage(our group description). If nobody knows Markdown, then Microsoft doc would be a good substitution but the entire work will be increased as we need to convert it to markdown. And it is also not good for keeping your copyright. Google doc is not really compatible with it, so it is not recommended.

JSON(20%) Quiz questions(30 passages*80 Questions/passage = 2400 Questions) and answers should be in JSON format. Do not be shocked by the quiz questions part, we could also generate some, see in part 3. But this is still a large amount of work.

Javascript(12%) This will be needed in the stock simulation, the loan calculation, etc. It will calculate information provided and return user with the result.

HTML(7%) This part will include the page design of stock simulation, the loan calculation, etc.

CSS(1%) This part won't need a lot of time. We might customize some CSS, but majority part of CSS comes from MaterilizeCss. Or other pre-defined library.

Java/Python/Javascript/C/C++(17%) This part include the work on the worker size, including providing different types of questions. The way of providing questions will be discussed in part 3. In worker, you could select the language you want. But I would like to recommend Javascript. Because the other language will require you to use assembly to load code. You can read more at here

Ejs(6%) ejs, a.k.a effective javascript, is the code that will be used to generate our knowledge passage. We have some pre-defined structure, but we do need to modify them to meet our needs. This part might take some time. Since I am not pretty familiar with it, 6% is just a rough estimate. But it won't take too long because we only need to do some small modifications.

Microsoft Powerpoint/Google Slides(2%): We need to do some presentations throughout this semester. I prefre to use Powerpoint as it has some wonderful functions. But Google slide is also acceptable.

CSV(2%) Organize and report the data of stocks in stock simulation tool. We need 8 stocks*10 groups = 80 data pieces. See 4 for more details.

Google Spreadsheet/Microsoft Excel(1%) Organize pass code, quiz code information.

Other(2%) In case I forget anything. And we do need time to design and choose right color, picture for each part in the website.

Statistic New(29%) Old(71%)

1.3 Technique/Software recommend

Description comes from their official or wikipedia. Each paragraph title is clickable, it will bring you to a description page or download page. For most of programs, we do not need to install.

Hexo A fast, simple & powerful blog framework. Hexo is a fast, simple & powerful blog framework powered by Node.js.

Cloudflare Worker Cloudflare Workers lets developers deploy serverless JavaScript applications on Cloudflare's global cloud network, where they are seamlessly scalable and closer to end users. Based on the Service Workers API, Workers receive events for every HTTP (S) request made to an application.

Gitpages GitHub Pages are public webpages hosted and easily published through GitHub.

Typora Typora allows you to manage your files easily, providing both file tree panel and articles (file list) side panel, allows you to manage your files easily. Organize your files your way, including putting in sync services, like Dropbox or iCloud.

Microsoft Powerpoint Microsoft PowerPoint empowers you to create clean slide presentations and intricate pitch decks and gives you a powerful presentation maker to tell your story.

Cloudflare CDN The Cloudflare CDN is an content delivery network that uses next-gen tech to deliver a fast, reliable, CDN services.

Github Release Releases are deployable software iterations you can package and make available for a wider audience to download and use.

HTML+JS pages HTML is the standard markup language for Web pages. JavaScript, often abbreviated as JS, is a programming language that conforms to the ECMAScript specification.

visual studio code Visual Studio Code is a code editor redefined and optimized for building and debugging modern web and cloud applications.

Livesever Launch a development local Server with live reload feature for static & dynamic pages on visual studio code.

Git Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. Git is easy to learn and has a tiny footprint with lightning fast performance .

Notepad++ Notepad++ is a free (as in "free speech" and also as in "free beer") source code editor and Notepad replacement that supports several languages. Running in the MS Windows environment, its use is governed by GNU General Public License.

Google Spreadsheet Google Sheets is a spreadsheet program included as part of a free, web-based software office suite offered by Google within its Google Drive service.

Microsoft Excel Microsoft Excel is a spreadsheet developed by Microsoft for Windows, macOS, Android and iOS. It features calculation, graphing tools, pivot tables, and a macro programming language called Visual Basic for Applications.

Hexo is actually based on Node.js. So, Node.js technically should be included.

Different library in jvascript will be discussed in each section.

1.4 Suggest roles

This part just states we need those role. It doesn't mean each one take one job of this. We should work on each part together.

- Writer
- Worker script
- Designer
- Speaker
- Tool Implement

2 Knowledge Passage

Host This page will be hosted at a subsite, at knowledge.capitaltwo.ga. Structure will be determined by Hexo + Material-X theme.

Naming rule The name of each passage file should be in words with dash in the middle between each two words. No white space or special character is permitted. For example, `How_to_do.md` is valid, `How to do.md` is not valid.

Final format of most of knowledge passage would be in Markdown format with valid yaml formatter. Each passage will be in one file. Each file have a yaml formatter.

yaml Formatter It likes a key value pair dictionary, each passage should at least have the following fields.

```
---
parentTitle: Crypto #parent title of this page, might be useful
title: Tell us who you interviewed #Title or topic of this page, like intro to crypto, part 1
type: dev #The category, like cypto
toc: true #True means including menu
body:
  - article
  - comments
date: 2020-09-30 11:53:46
top:
keywords:
  - Design
categories: Development #The category, like cypto
tags: #Tags, like primaey level, seondary level, or other good tag.
  - Design
---
```

Notice

Keys are caase sensitive. And three dash line in the front and the end should be included. In Typora, you can just type `---` and press `enter` to enter the front matter typing mode. In this mode, dash line in the front and the end should **NOT** be included as typora will do that for you.

Main Passage Main Passage will be in Markdown format. For the best experience of converting and user experience, Markdown is the best. Though Microsoft word and google doc is acceptable.

A short reference to useful command:

```
### aa <!--Title, one more # means a lower level title-->
- aa <!--Unordered list-->
```

```
1. aa <!--ordered list-->
[something](link) <!--Links -->
![](link) <!--images, links required to be online source. Good imagebed would be Onedrive-->
`code` <!--Small field keyword highlighting-->
**text** <!--Make text bold-->
*text* <!--Make text in italian-->
> Reference <!--Do a reference, large quota-->
```

You can also utilize HTML code in each markdown file to create multi-media document, just put HTML code to markdown file, then it will work, some common codes:

```
<iframe src="https://docs.google.com/forms/d/e/1
    ↪ FAIpQLSeIQZNBcicL01lptHL_VDwL5u9KR6iDYFNE-ElCjMQ88VDBDQ/viewform?usp=sf_link" title="
    ↪ Questionnaires" width="100%" height="450 pixels"></iframe>
<mark></mark> <!--To highlight some words-->
```

2.1 Quiz Reference

If this passage is associated with a quiz, usually this is always true, you need also to include a unique quizpart code. It will be passed to quiz part. Each passage should have a unique code. The code doesn't necessary to make sense in their word meaning. It could be a random word. I recommend for a higher security by using aes, which will be discussed in pointing system at 2.2. But it is not required at here. Because user might want to use the quiz without finishing the reading requirement.

To include a quizepart code, add following field to previous yaml:

```
quizepart: XXX #The quiz part code
```

Except including those stuff, you also need to put the quizpart field to a shared google sheet/ Microsoft Excel.

2.2 Pointing System

If this passage require user to finish a previous chapter, you then need to include the following field:

```
password: XXX #The encryption password.
```

Except including those stuff, you also need to specify a securitynumber in a shared google sheet/ Microsoft Excel, which will be used to verificate user activity. And you also need to put the password field to this document.

Method behind this - finish one study Each passage with a previous reading requirement is associated with a securitynumber, a password, which are stored in the worker side. When a user finish a quiz, they could choose either go to next chapter directly(use local storage to store the credential text and go to the next page directly) or get a credential text directly, they could choose to copy the text or download as a txt file. To see

code of downloading, refer to after quiz page at 3.3. By saving it, they can use it and continue their study next time. The credential text is generated on worker and returned to client. It will be encrypted by a high level security code via AES-256 encryption, like Secret Passphrase. This phrase will be stored in the worker.

credential text

[name of current chapter], [the securitynumber], [Browser Identification String], [IP Address]

For sample code of this part, refer to 3.3.

The reason I choose this is that user will have the full control of their own data. So we are free of information storage risk and privacy concern. And we do not need to use cookies as well.

Method behind this - continue their study To continue their study using those credential, the main page and each page that require credential will have a window for them to upload their credential before continue. Once they upload, their text will be passed to worker to decrypt and worker will return an address for that passage for them to continue. Client will also be required to check the desired passage has a correct securitynumber, browser identification, IP address or not. Or if the worker code detect any cheating, worker will just return home page with an error message. Client js will then tell user those information.

The sample code for loading file:

```
document.getElementById('inputfile').addEventListener('change', function() {
  var fr=new FileReader();
  fr.onload=function(){
    //do something with fr.result, here is one sample
    //pass to the server using fetch and see if correct information is provided
    document.getElementById('output').textContent=fr.result;
  }

  fr.readAsText(this.files[0]);
})
```

The sample JSON data sever should return:

```
{
    securitynumber: "XXX",
    next: "absolute address to the next passage",
    browserID: "XXX",
    IP: "XXX"
}
```

The sample worker side that returns JSON: read more at https://developers.cloudflare.com/workers/learning/gett started

```
addEventListener("fetch", event => {
  const data = {
```

```
    hello: "world"
  }

  const json = JSON.stringify(data, null, 2)

  return event.respondWith(
    new Response(json, {
      headers: {
        "content-type": "application/json;charset=UTF-8"
      }
    })
  )
})
```

The client will then direct user to the page in the "next" field and also pass in those stuff as a localstorage. The page in the "next" field will finish the remaining validation stuff.

For the validation in each page, we might check the IP and BrowserID locally. And then check the securitycode with worker side by passing a JSON object, worker side will check if security code of this passage is correct or not. If result is correct, user have access, otherwise user doesn't have access.

Sample request JSON:

```
{
    passageName: "XXX",
    securityCode: "XXX"
}
```

Return is just a JSON of true or false and password for this passage.

Encrypt & Decrypt All encription and decription, no matter sever side or user side, will always be handled by using AES via crypto-js. Sample code shown below: More details could be read at https://github.com/brix/crypto-js and https://www.c-sharpcorner.com/blogs/advanced-encryption-in-javascript-using-cryptojs

```
// Encrypt
var ciphertext = CryptoJS.AES.encrypt('my message', 'secret key 123').toString();

// Decrypt
var bytes = CryptoJS.AES.decrypt(ciphertext, 'secret key 123');
var originalText = bytes.toString(CryptoJS.enc.Utf8);
```

We also need to include their script in the header:

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/cryptojs/3.1.2/rollups/aes.js">
</script>
```

In the worker side, we will still use crypto-js with another method, which is we need to require first.

```
var AES = require("crypto-js/aes");

// Encrypt
var ciphertext = CryptoJS.AES.encrypt('my message', 'secret key 123').toString();

// Decrypt
var bytes = CryptoJS.AES.decrypt(ciphertext, 'secret key 123');
var originalText = bytes.toString(CryptoJS.enc.Utf8);
```

3 Quizzes

Host This page will be hosted at main site in a sub directory, at capitaltwo.ga/quiz/. Structure will be described in Client Side at 3.2.2.

Quiz part will utilize the Cloudflare Worker functions. Quiz question will be stored in each script. Each script has a size limit of 1 MB, it is enough to store 160-240 questions. (Assume 200 chars per question, 200*240 = 48000 < 1048576)

Each generated quiz would contain some of Pre-defined Quiz and some Generated quiz, the proportion of each part could be pre-defined or not pre-defined. We could get the quiz questions from quizlet.

3.1 Quiz Type

3.1.1 Pre-defined Quiz

All Pre-defined Quiz question will be pre-defined in JSON field.

Sample pre-defined quiz:

Pre-defined Quiz

Is 401K a tax deferred retirement plan funded by employees of profit seeking business?
A. Yes
B. No

Sample JSON format:

```
[
   {
      "QuestionText": <The text of question>,
      "Choices": <[option1,option2,...]>,
      "Answer": <Choice>,
      "Explanation": <word description>,
   }, ...
]
```

3.1.2 Generated Quiz

Generated quiz is a quiz that have some variables able to be changed. Each quiz question means a pair of generation and validation method in worker. Sample generated quiz: The 1000 and 3.6 in the following question could be changed.

Generated Quiz

You rent 1000 from bank at an APR of 3.6% for a year, how much interest you need to pay for a year?

_____

Because we will not have a lot of calculation in the quiz part, I guess we could just define each question seperatly. It would be enough.

3.2 Preparing Quiz

3.2.1 Worker side

Worker will generate the quiz and return a JSON to a request. Because the original question is in JSON file, this place is just to randomly select some question from the bank. The question bank is just an object variable in script. We might add more verification in the request header in the future. For the code of returning JSON, see 2.2. Here is the sample JSON file returned from worker:

```
[
    {
        "QNumber": <The index of number, the same as the order of problem set, for indexing purpose
            ↪ >,
        "QuestionText": <The text of question>,
        "QuestionType": <Either pre-defined or generated>,
        "QuestionSubType": <none if multiple choice, otherwise an identification code of validation
            ↪ method>,
        "Choices": <none or [option1,option2,...]>, //none means it is a calculation question.
    }, ...
]
```

Some data describing requiesting detail, such as quizcode, will be passed into worker in the data part. And then worker could use it to navigate to the correct question.

Whether generate quiz in one file (use header to distinguish the function) or in seperate files (client call different address on worker) on worker is up to designer.

Worker side will also have script to check the correctness of the quiz by using the QNumber or calculation method and return correctness for each question. Client will pass in JSON as body in a fetch request, sample passed in JSON:

```
[
    {
```

```
        "QNumber": <The index of number, the same as the order of problem set, for indexing purpose
            ↪ >,
        "Selection": <Choice>
    }, ...
]
```

Sample return JSON for validation process: The order of each question should remain the same as the order in the pass in data.

```
[
    {
        "QNumber": <The index of number, the same as the order of problem set, for indexing purpose
            ↪ >,
        "Answer": <Choice>,
        "Explanation": <word description>,
    }, ...
]
```

3.2.2 Client Side

Client side is just to formulate the data as described in the pass in side in the above part. And it will loop through the data passed back to either formulate questions or generate a report, like check answer and showing explanation. If a user pass the quiz, user will be guided to the next section After finishing quiz at 3.3.

In the main page of Client Side, which is located at capitaltwo.ga/quiz/index.html. The default page will have a input window asking user for quizcode. Then it will guide user to capitaltwo.ga/quiz/index.html?quizcode=XXX. This page will be generated to be the page for this quiz. Other page could navigate user to the quiz page by using capitaltwo.ga/quiz/index.html?quizcode=XXX. Then they will not see the default page in the case.

The method behind this is to analyze the parameter in the javascript part, sample code for getting parameters as shown below:

```
var request = {
       QueryString: function(val) {
      var uri = window.location.search;
          var re = new RegExp("" + val + "=([^&?]*)", "ig");
          return ((uri.match(re)) ? (uri.match(re)[0].substr(val.length + 1)) : null);
       }
}

request.QueryString("wind") == null;//check no parameters called wind
request.QueryString("wind");//get the value of parameter of wind
```

By using those information get, this javascript will then write corresponding elements by using DOM Manipulation Methods in jQuery. Sample:

```
// Creating a new div element
var newDiv = document.createElement("div");

// Creating a text node
var newContent = document.createTextNode("Hi, how are you doing?");

// Adding the text node to the newly created div
newDiv.appendChild(newContent);

// Adding the newly created element and its content into the DOM
var currentDiv = document.getElementById("main");
document.body.appendChild(newDiv, currentDiv);
```

Jquery can also do this by using append():

```
$('#div1').after('<div style="background-color:yellow"> New div </div>');
```

Sample HTML:

```
<div id="main">
    <h1 id="title">Hello World!</h1>
    <p id="hint">This is a simple paragraph.</p>
</div>
```

3.3 After finishing quiz

If user passed this quiz and checked all explanations, user will get choices of getting certificate or go to next passage directly.

If user choose to go to the next page directly, the client will then ask worker provide related JSON object, and then direct user to the page in the "next" field and also pass in those stuff as a localstorage. The sample received stuff is described in previous part at 2.2. The page in the "next" field will finish the remaining validation stuff. The request JSON is slightly different than the request JSON discussed before. And worker should have a seperate function for it, the sample request JSON.

```
{
    quizcode: "XXX",
    browserID: "XXX",
    IP: "XXX"
}
```

If they choose to get certificate, the client side script will pass in some information, including [quizcode], [Browser Identification String], [IP Address] to worker and will receive an encrypted text. See later part for how this works. And a small window will go out with certificate information. The small window sample code is:

```
function OpenDetial(id, width, height) {
    var _srcWidth = parseInt(window.innerWidth || document.documentElement.clientWidth || document.
        ↪ body.clientWidth);
    var _srcHeight = parseInt(window.innerHeight || document.documentElement.clientHeight ||
        ↪ document.body.clientHeight);
    _width = parseInt(width) || _srcWidth;
    _height = parseInt(height) || _srcHeight;

    var cssWidth = width == undefined ? "100%" : _width;
    var cssHeight = height == undefined ? "100%" : _height;

    var left = (_srcWidth - _width) == 0 ? 0 : ((_srcWidth - _width) / 2);
    var top = (_srcHeight - _height) == 0 ? 0 : ((_srcHeight - _height) / 2);



    var _id = "#" + id;
    var detial;
        detial =
        '<div id="panel"> <div class="card white z-depth-3"><div class="card-content black-text"> <
            ↪ div class="row">' +
        '<div class="file-field input-field col s12"> <div class="btn"><span>File</span><input id="
            ↪ history_data" name="history_data" type="file" onchange="loadFile(this)" accept=".csv
            ↪ "></div><div class="file-path-wrapper"><input placeholder="Input Stream" id="
            ↪ Input_Stream" class="file-path validate" type="text"></div></div><p class="col s12"><
            ↪ label><input type="checkbox" onchange="switchExport(this);" /><span>I don\'t need to
            ↪ export usage data(Choose when source is intended to be large).</span></label></p><p
            ↪ class="col s8" id="notice"><p><button id="usethis" class="waves-effect waves-light btn
            ↪  col s3" disabled onclick="Start()">Use This</button>' +
        '</div></div></div></div>';

    $(_id).html(detial);
    $(_id).css({
        "display": "none",
        "position": "fixed",
        "bottom": "0px",
        "width": "100%",
        "top": "0px",
        "height": "100%",
        "background-color": "rgba(223, 236, 236, 0.36)"
    });
    $("#panel").css({
```

```
        "width": cssWidth,
        "height": cssHeight,
        "left": left,
        "top": top,
        "position": "fixed"
    });
    $(_id).slideDown();
}

OpenDetial("<div id>", 400, 200);//open this window
```

If user choose to download it, Then client could generate a download like this:

```
var funDownload = function(content, filename) {
    var eleLink = document.createElement('a');
    eleLink.download = filename;
    eleLink.style.display = 'none';
    var blob = new Blob([content]);
    eleLink.href = URL.createObjectURL(blob);
    document.body.appendChild(eleLink);
    eleLink.click();
    document.body.removeChild(eleLink);
};

funDownload(text, 'credential.txt');
```

We will use body to pass information to worker, Sample JS to request information, read more at https://developer.
US/docs/Web/API/Fetch_API/Using_Fetch

```
fetch('https://XXX.1442334619.workers.dev', {
  headers: {
    'User-agent': 'Mozilla/4.0 Custom User Agent',
    'Content-Type': 'application/json'
    // 'Content-Type': 'application/x-www-form-urlencoded',
  },
  body: JSON.stringify(data)
})
.then(response => response.json())
.then(data => {
  console.log(data)
})
.catch(error => console.error(error))
```

sample JSON request body

```
{
    quizcode: "XXX",
    IP: "XXX"
}
```

Sample code for returning text from worker side, this step will need to install worker local environment. It will need to use quizcode information to get the securitycode of next passage and other information to finish the encryption.

```
var AES = require("crypto-js/aes");

addEventListener("fetch", event => {
  event.respondWith(handleRequest(event.request))
})

async function handleRequest(request) {
  //use request.body and request.header and handle encryption using AES
  return new Response("Hello worker!", {
    headers: { "content-type": "text/plain" }
  })
}
```

4 Stock Simulation Tool

Host This page will be hosted at main site in a sub directory, at capitaltwo.ga/stock/. Structure could be determined by user implementing it. Data part of this section may be put in the same directory or using Guthub release/raw.github.com in another repo if it is too big.

This tool will get some history data and let user have a simulated stock trade experience. User will could have an initial amount of money and trade in and trade out at some time. We will compress a day's trend chart to 10 minutes.

Data Preparation Data source could be stored in Github, it might also be held in other cloud storage. We will provide user with 5 stocks to play with in each game. As required of randomization, we will prepare 10 pairs of data. Each data will include the following information in csv like this:

```
Date,Open,High,Low,Close,Volume,Adj Close
11:59:16,43.23,43.32,42.91,43.20,28942700,43.20
11:59:16,42.84,43.45,42.65,43.23,30314900,43.23
11:59:17,42.74,43.17,42.21,42.74,24634000,42.74
11:59:18,43.31,43.46,42.83,43.08,26266400,43.08
11:59:19,42.97,43.47,42.81,43.37,34244200,43.37
```

```
11:59:20,43.21,43.25,42.60,42.86,31170300,42.86
11:59:20,43.38,43.69,43.08,43.16,31537500,43.16
11:59:20,44.07,44.10,43.29,43.58,31921400,43.58
11:59:21,43.91,44.09,43.64,43.89,27763100,43.89
```

Data Reading Because this is a large amount of data, it might cause a high traffic and memory load would be high, in extreme case, it might also cause IO choke. We will use Papa Parse to use stream to fix this issue.

```
Papa.parse("http://example.com/file.csv", {
   header: true,
        skipEmptyLines: true,
        dynamicTyping: true,
   download: true,
   step: function(results, parser) {
          console.log("Row data:", results.data["Time"]);
   complete: function(results) {
      console.log(results);
   }
});
```

As always, we will need to include their js file in our HTML header.

Data Representation For the data representation, I think amcharts. A sample representation code like below: Refer to https://www.amcharts.com/demos/stock-chart/ for more details.

```
am4core.ready(function() {

// Themes begin
am4core.useTheme(am4themes_animated);
// Themes end

// Create chart
var chart = am4core.create("chartdiv", am4charts.XYChart);
chart.padding(0, 15, 0, 15);

// Load external data
chart.dataSource.url = "https://www.amcharts.com/wp-content/uploads/assets/stock/MSFT.csv";
chart.dataSource.parser = new am4core.CSVParser();
chart.dataSource.parser.options.useColumnNames = true;
chart.dataSource.parser.options.reverse = true;

// the following line makes value axes to be arranged vertically.
```

```
chart.leftAxesContainer.layout = "vertical";

// uncomment this line if you want to change order of axes
//chart.bottomAxesContainer.reverseOrder = true;

var dateAxis = chart.xAxes.push(new am4charts.DateAxis());
dateAxis.renderer.grid.template.location = 0;
dateAxis.renderer.ticks.template.length = 8;
dateAxis.renderer.ticks.template.strokeOpacity = 0.1;
dateAxis.renderer.grid.template.disabled = true;
dateAxis.renderer.ticks.template.disabled = false;
dateAxis.renderer.ticks.template.strokeOpacity = 0.2;
dateAxis.renderer.minLabelPosition = 0.01;
dateAxis.renderer.maxLabelPosition = 0.99;
dateAxis.keepSelection = true;
dateAxis.minHeight = 30;

dateAxis.groupData = true;
dateAxis.minZoomCount = 5;

// these two lines makes the axis to be initially zoomed-in
// dateAxis.start = 0.7;
// dateAxis.keepSelection = true;

var valueAxis = chart.yAxes.push(new am4charts.ValueAxis());
valueAxis.tooltip.disabled = true;
valueAxis.zIndex = 1;
valueAxis.renderer.baseGrid.disabled = true;
// height of axis
valueAxis.height = am4core.percent(65);

valueAxis.renderer.gridContainer.background.fill = am4core.color("#000000");
valueAxis.renderer.gridContainer.background.fillOpacity = 0.05;
valueAxis.renderer.inside = true;
valueAxis.renderer.labels.template.verticalCenter = "bottom";
valueAxis.renderer.labels.template.padding(2, 2, 2, 2);

//valueAxis.renderer.maxLabelPosition = 0.95;
valueAxis.renderer.fontSize = "0.8em"

var series = chart.series.push(new am4charts.LineSeries());
series.dataFields.dateX = "Date";
```

```
series.dataFields.valueY = "Adj Close";
series.tooltipText = "{valueY.value}";
series.name = "MSFT: Value";
series.defaultState.transitionDuration = 0;

var valueAxis2 = chart.yAxes.push(new am4charts.ValueAxis());
valueAxis2.tooltip.disabled = true;
// height of axis
valueAxis2.height = am4core.percent(35);
valueAxis2.zIndex = 3
// this makes gap between panels
valueAxis2.marginTop = 30;
valueAxis2.renderer.baseGrid.disabled = true;
valueAxis2.renderer.inside = true;
valueAxis2.renderer.labels.template.verticalCenter = "bottom";
valueAxis2.renderer.labels.template.padding(2, 2, 2, 2);
//valueAxis.renderer.maxLabelPosition = 0.95;
valueAxis2.renderer.fontSize = "0.8em"

valueAxis2.renderer.gridContainer.background.fill = am4core.color("#000000");
valueAxis2.renderer.gridContainer.background.fillOpacity = 0.05;

var series2 = chart.series.push(new am4charts.ColumnSeries());
series2.dataFields.dateX = "Date";
series2.dataFields.valueY = "Volume";
series2.yAxis = valueAxis2;
series2.tooltipText = "{valueY.value}";
series2.name = "MSFT: Volume";
// volume should be summed
series2.groupFields.valueY = "sum";
series2.defaultState.transitionDuration = 0;

chart.cursor = new am4charts.XYCursor();

var scrollbarX = new am4charts.XYChartScrollbar();
scrollbarX.series.push(series);
scrollbarX.marginBottom = 20;
scrollbarX.scrollbarChart.xAxes.getIndex(0).minHeight = undefined;
chart.scrollbarX = scrollbarX;


// Add range selector
```

```
var selector = new am4plugins_rangeSelector.DateAxisRangeSelector();
selector.container = document.getElementById("controls");
selector.axis = dateAxis;

}); // end am4core.ready()
```

To use it, we need to include their js library:

```
<style>
#chartdiv {
  width: 100%;
  height: 500px;
  max-width: 100%;
}

#controls {
  overflow: hidden;
  padding-bottom: 3px;
}
</style>
<script src="https://cdn.amcharts.com/lib/4/core.js"></script>
<script src="https://cdn.amcharts.com/lib/4/charts.js"></script>
<script src="https://cdn.amcharts.com/lib/4/plugins/rangeSelector.js"></script>
<script src="https://cdn.amcharts.com/lib/4/themes/animated.js"></script>
<div id="controls"></div>
<div id="chartdiv"></div>
```

Notice

We will draw the graph animated, so it is not just using csv file directly as code above. We will pass data into the graph asa time goes.

User Interaction For the user interaction, user could trade in and trade out by put number or press button. The key solution at here is to refresh the page every 100 ms(or other rational value). Sample refresh code would like this:

If asynchronous, not really recommend as js is synchronous.

```
var hoverId = setInterval(function() {

    //Do whatever you want at here, below is just a sample
    data.push({
        name: history.getLast().getKey(),
        value: [
```

```
            history.getLast().getKey(),
            history.getLast().getValue()
        ]
    });

}, freq);

clearInterval(hoverId); //Use this when we no longer want to refresh the page
```

Synchronous method could be used to parallel with file reading:

```
setTimeout(function() {
    var delay = function() {
        if (Model_Status == 1) {
            //do some things at here
            parser.resume();
        } else {
            setTimeout(function() {
                delay();
            }, 200)
        }
    }
    delay();
}, freq);
```

Further Notice

User's purchasing might influence stock market, and we can calculate this asa well. But I do not suggest this as JavaScript will have error in those detailed floating calculation. And user's influence is pretty low.

5 Retirement Calculation Tool

Host This page will be hosted at main site in a sub directory, at capitaltwo.ga/retire/. Structure could be determined by user implementing it.

This part doesn't have any special stuff. We will collect some information and let user select some common retirement plan, or customize their own plan. And click calculate to get the calculation result and see the amount of money they could receive in the next few months.

This part will only use some normal javascript function with some calculation logic. When representing result, we could use another page or rewrite current page using javascript. We might represent some pie chart for some data, code is represented in **??**, like money from social security and money from deposite.

6 Loan Calculation Tool

Host This page will be hosted at main site in a sub directory, at capitaltwo.ga/loan/. Structure could be determined by user implementing it.

This part doesn't have any special stuff. We will collect some information and let user select some common loan plan and the duration of loan, or customize their own plan. And click calculate to get the calculation result. We will represent result in a table to show how much they need to pay in the next few months.

This part will only use some normal javascript function with some calculation logic. When representing result, we could use another page or rewrite current page using javascript. We might represent some pie chart for some data, code is represented in **??**, like the interest and your base money.

7 Budgeting Tool

Host This page will be hosted at main site in a sub directory, at capitaltwo.ga/budget/. Structure could be determined by user implementing it.

This will be in a collapsibles list and user could choose between those two.

7.1 Monthly Budgeting Tool

This is a normal one month calculation. People will input their money in and money out category, then we just generate a report of money usage of current month. This is just a simple calculation to let money in - money out. And we could also draw some spending category chart using echarts. (Or amchart) A pie chart would be great.

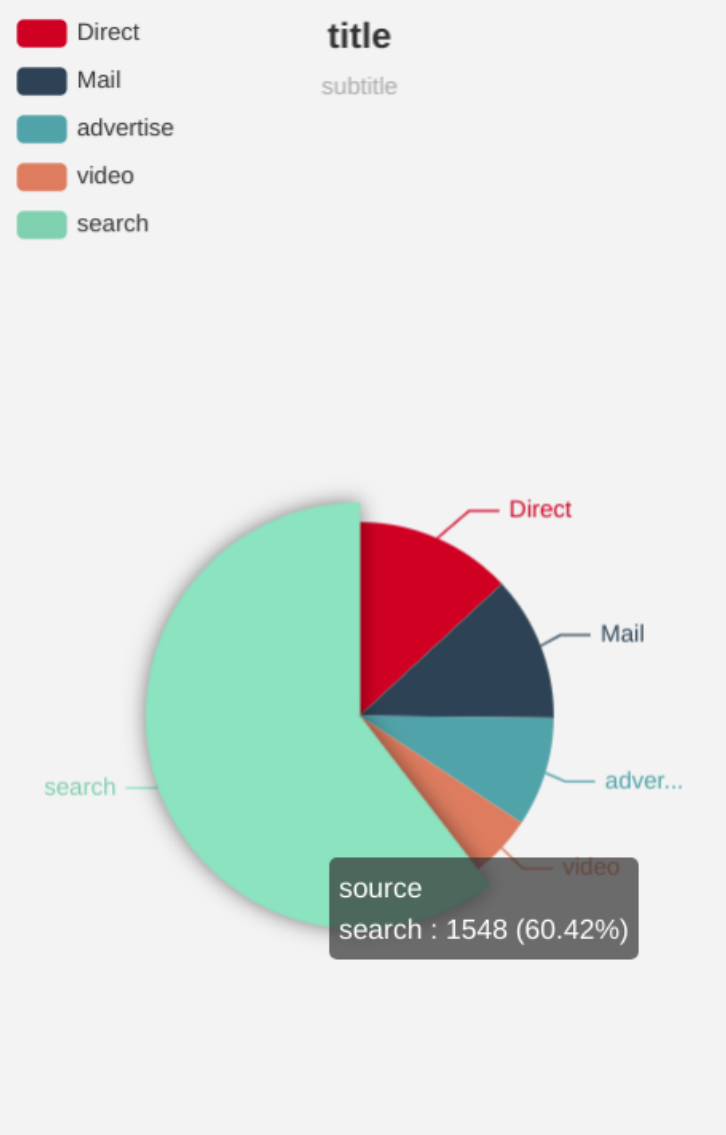
A sample pie chart using echarts at **??**, in this sample a div have `main` should be defined, like in **??**:

```
var myChart = echarts.init(document.
    ↪ getElementById('main'));
var options = {
    title: {
        text: 'title',
        subtext: 'subtitle',
        left: 'center'
    },
    tooltip: {
        trigger: 'item',
        formatter: '{a} <br/>{b} : {c} ({d}%)'
    },
    legend: {
        orient: 'vertical',
        left: 'left',
        data: ['Direct', 'Mail', 'advertise', '
            ↪ video', 'search']
    },
    series: [
        {
            name: 'source',
            type: 'pie',
            radius: '55%',
            center: ['50%', '60%'],
            data: [
                {value: 335, name: 'Direct'},
                {value: 310, name: 'Mail'},
                {value: 234, name: 'advertise'},
                {value: 135, name: 'video'},
                {value: 1548, name: 'search'}
            ],
            emphasis: {
                itemStyle: {
                    shadowBlur: 10,
                    shadowOffsetX: 0,
                    shadowColor: 'rgba(0, 0, 0, 0.5)
                        ↪ '
                }
            }
        }
    ]
};

myChart.setOption(options);
```

Need to import script:

```
<script src="https://cdn.staticfile.org/echarts
    ↪ /4.3.0/echarts.min.js"></script>
```



We could also analyze the data and made some suggestion to their spending.

7.2 Year Budgeting Tool

User could type their money on each category or item with an importance level associated to it. With those information, we will solve it by using an optimization rules. User could choose either optimizing saving or optimizing consuming (maybe another word at here).

In this part, we will utilize jsLPSolver and solve the problem. Refer to 7.3 to see the sample of using it.

User will have choices to indicate if this is a one time purchase or not, like wishlist. Then we will make a plan to tell user when would be the best time to buy it. And in order to achieve this goal, what they should do in each month.

In this category, we will consider the saved money will be used in some low-risk investment, like 0.036 APR. The report generated would be the item should be bought at that month.

We could support up to 100 level of priority, but priority of 1 is the essential stuff, like housing, foods. Snacks might be in a lower priority.

Optimize saving This process doesn't require optimization, thought it could be used to reach this purpose. But it might be as easy as loop through the list and select all components that have a priority level of 1. Sample methods as below:

Algorithm 1 Normal method for yearly budgeting

```
1: procedure YEAR(L, P)                    ▷ Variable definition is shown in the next part
2:     for j < 12 do
3:         for i ∈ {O ∪ R} do
4:             if L_ij == 1 then
5:                 S_ij ← true
6:             end if
7:         end for
8:     end for
9:     return S                            ▷ The selection result of each month
10: end procedure
```

Optimize consume Define variables: P_{ij} – The amount of money that needs to be spent in different categories i in month j.

L_{ij} – The priority level associate with each spending category, P_{ij}. It is the reverse of the original list done by user. Like for high priority stuff, L_{ij} will be high

I_j – The anticipated money in in each month.

T_j – The money available at the end of month J.

S_{ij} – The binary selector to indicate this category should be selected or not in the month j.

A – The one month rate of buying some low-risk investment product.

O – Set including one time purchase, like wait list.

R – Set including repeating purchase.

Formula:

$$\begin{aligned}
\text{Maxmize} \quad & \sum_{i}^{R\cup O}\sum_{j}^{12} L_{ij}S_{ij} \\
\text{Subject to} \quad & \sum_{i}^{n} P_{ij}S_{ij} \leq AT_{j-1} + I_j \\
& T_j = AT_{j-1} - \sum_{i}^{n} P_{ij}S_{ij} + I_j \qquad (1) \\
& \sum_{i}^{O}\sum_{j}^{12} S_{ij} = 1 \\
& \sum_{i}^{R} S_{ij} = 1 \quad \forall j \in \{1...12\}
\end{aligned}$$

Though optimization might give us the optimal solution in complex cases, but it is also acceptable to use normal methods as shown below.

Algorithm 2 Normal method for yearly budgeting

```
procedure YEAR(L, P)                                   ▷ Variable definition is shown above
    T_i ← 0 ∀i ∈ {1...12}
    R ← 0                                              ▷ The remains money of the current month
    for j < 12 do
        M_i ← Sort current P_ij in importance order L_ij and amount of money    ▷ less goes first
        Remove finished M_i ∀i ∈ O                     ▷ O stands for the group of one time purchase
        R ← I_j + AT_{j-1}
        i ← 0
        while R > 0 do
            R ← R − M_i
            S_ij ← true
            i ← next available i
        end while
        T_j ← R
    end for
    return S                                           ▷ The selection result of each month
end procedure
```

7.3 Sample codes

Sample Question Each week you're limited to 300 square feet of wood, 110 hours of labor, and 400 square feet of storage.

A table uses 30sf of wood, 5 hours of labor, requires 30sf of storage and has a gross profit of $1,200. A dresser uses 20sf of wood, 10 hours of work to put together, requires 50 square feet to store and has a gross profit of $1,600.

How much of each do you produce to maximize profit, given that partial furniture aren't allowed in this dumb world problem?

For the model Definition of model:
W: wood provided
L: labor provided
S: storage provided
T: amount of table made
D: amount of dresser produced

Math representation:

$$\begin{aligned} \text{Maxmize} \quad & 1200T + 1600D \\ \text{Subject to} \quad & 30T + 20D \leq 300 \\ & 5T + 10D \leq 110 \\ & 30T + 50D \leq 400 \end{aligned} \tag{2}$$

For the code Code as shown below:

```
var model = {
       "optimize": "profit",
       "opType": "max",
       "constraints": {
           "wood": {"max": 300},
           "labor": {"max": 110},
           "storage": {"max": 400}
       },
       "variables": {
           "table": {"wood": 30, "labor": 5, "profit": 1200, "table": 1, "storage": 30},
           "dresser": {"wood": 20, "labor": 10, "profit": 1600, "dresser": 1, "storage": 50}
       },
       "ints": {"table": 1, "dresser": 1}
   }

console.log(solver.Solve(model));
```

8 Home Page

Host This page will be hosted at main site in a sub directory, at capitaltwo.ga/index.html.

This is a pure HTML + JS page. I will suggest to use Materialcss. By filling the data and do a little edit to their Parallax Template, then we could build a home page as template. The home page would include our information and purpose of this website, as well as our functions, special stuff.

9 Copyright info and Disclaimer

Host This page will be hosted at main site in a sub directory, at capitaltwo.ga/legal.html. Or in a defaault place created by Github.

The whole project will use multiple libraries and multiple source codes. Our code could be licensed through MIT, usually saw as:

MIT License

MIT License

Copyright (c) 2020 Capitaltwo

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

We should also include license of other library we use. Sample as shown below:

We will need to put those information in a place easy to see. For page that are hard to see, we could provide a bar and guide user to go to the correct page to see it.

10 Color and Style choosing

Host Most part of this section will be hold in other website.

We need to define one the major color for all page. And up to 5 supplement colors. Also we need to define the color for reference section, links, etc. Some color palette as shown below:

Color Palette

#ffebee red lighten-5	#fce4ec pink lighten-5	#f3e5f5 purple lighten-5
#ffcdd2 red lighten-4	#f8bbd0 pink lighten-4	#e1bee7 purple lighten-4
#ef9a9a red lighten-3	#f48fb1 pink lighten-3	#ce93d8 purple lighten-3
#e57373 red lighten-2	#f06292 pink lighten-2	#ba68c8 purple lighten-2
#ef5350 red lighten-1	#ec407a pink lighten-1	#ab47bc purple lighten-1
#f44336 red	#e91e63 pink	#9c27b0 purple
#e53935 red darken-1	#d81b60 pink darken-1	#8e24aa purple darken-1
#d32f2f red darken-2	#c2185b pink darken-2	#7b1fa2 purple darken-2
#c62828 red darken-3	#ad1457 pink darken-3	#6a1b9a purple darken-3
#b71c1c red darken-4	#880e4f pink darken-4	#4a148c purple darken-4
#ff8a80 red accent-1	#ff80ab pink accent-1	#ea80fc purple accent-1
#ff5252 red accent-2	#ff4081 pink accent-2	#e040fb purple accent-2
#ff1744 red accent-3	#f50057 pink accent-3	#d500f9 purple accent-3
#d50000 red accent-4	#c51162 pink accent-4	#aa00ff purple accent-4

#ede7f6 deep-purple lighten-5	#e8eaf6 indigo lighten-5	#e3f2fd blue lighten-5
#d1c4e9 deep-purple lighten-4	#c5cae9 indigo lighten-4	#bbdefb blue lighten-4
#b39ddb deep-purple lighten-3	#9fa8da indigo lighten-3	#90caf9 blue lighten-3
#9575cd deep-purple lighten-2	#7986cb indigo lighten-2	#64b5f6 blue lighten-2
#7e57c2 deep-purple lighten-1	#5c6bc0 indigo lighten-1	#42a5f5 blue lighten-1
#673ab7 deep-purple	#3f51b5 indigo	#2196f3 blue
#5e35b1 deep-purple darken-1	#3949ab indigo darken-1	#1e88e5 blue darken-1
#512da8 deep-purple darken-2	#303f9f indigo darken-2	#1976d2 blue darken-2
#4527a0 deep-purple darken-3	#283593 indigo darken-3	#1565c0 blue darken-3
#311b92 deep-purple darken-4	#1a237e indigo darken-4	#0d47a1 blue darken-4
#b388ff deep-purple accent-1	#8c9eff indigo accent-1	#82b1ff blue accent-1
#7c4dff deep-purple accent-2	#536dfe indigo accent-2	#448aff blue accent-2
#651fff deep-purple accent-3	#3d5afe indigo accent-3	#2979ff blue accent-3
#6200ea deep-purple accent-4	#304ffe indigo accent-4	#2962ff blue accent-4

#e1f5fe light-blue lighten-5	#e0f7fa cyan lighten-5	#e0f2f1 teal lighten-5
#b3e5fc light-blue lighten-4	#b2ebf2 cyan lighten-4	#b2dfdb teal lighten-4
#81d4fa light-blue lighten-3	#80deea cyan lighten-3	#80cbc4 teal lighten-3
#4fc3f7 light-blue lighten-2	#4dd0e1 cyan lighten-2	#4db6ac teal lighten-2
#29b6f6 light-blue lighten-1	#26c6da cyan lighten-1	#26a69a teal lighten-1
#03a9f4 light-blue	#00bcd4 cyan	#009688 teal
#039be5 light-blue darken-1	#00acc1 cyan darken-1	#00897b teal darken-1
#0288d1 light-blue darken-2	#0097a7 cyan darken-2	#00796b teal darken-2
#0277bd light-blue darken-3	#00838f cyan darken-3	#00695c teal darken-3
#01579b light-blue darken-4	#006064 cyan darken-4	#004d40 teal darken-4
#80d8ff light-blue accent-1	#84ffff cyan accent-1	#a7ffeb teal accent-1
#40c4ff light-blue accent-2	#18ffff cyan accent-2	#64ffda teal accent-2
#00b0ff light-blue accent-3	#00e5ff cyan accent-3	#1de9b6 teal accent-3
#0091ea light-blue accent-4	#00b8d4 cyan accent-4	#00bfa5 teal accent-4

#e8f5e9 green lighten-5	#f1f8e9 light-green lighten-5	#f9fbe7 lime lighten-5
#c8e6c9 green lighten-4	#dcedc8 light-green lighten-4	#f0f4c3 lime lighten-4
#a5d6a7 green lighten-3	#c5e1a5 light-green lighten-3	#e6ee9c lime lighten-3
#81c784 green lighten-2	#aed581 light-green lighten-2	#dce775 lime lighten-2
#66bb6a green lighten-1	#9ccc65 light-green lighten-1	#d4e157 lime lighten-1
#4caf50 green	#8bc34a light-green	#cddc39 lime
#43a047 green darken-1	#7cb342 light-green darken-1	#c0ca33 lime darken-1
#388e3c green darken-2	#689f38 light-green darken-2	#afb42b lime darken-2
#2e7d32 green darken-3	#558b2f light-green darken-3	#9e9d24 lime darken-3
#1b5e20 green darken-4	#33691e light-green darken-4	#827717 lime darken-4
#b9f6ca green accent-1	#ccff90 light-green accent-1	#f4ff81 lime accent-1
#69f0ae green accent-2	#b2ff59 light-green accent-2	#eeff41 lime accent-2
#00e676 green accent-3	#76ff03 light-green accent-3	#c6ff00 lime accent-3
#00c853 green accent-4	#64dd17 light-green accent-4	#aeea00 lime accent-4

#fffde7 yellow lighten-5	#fff8e1 amber lighten-5	#fff3e0 orange lighten-5
#fff9c4 yellow lighten-4	#ffecb3 amber lighten-4	#ffe0b2 orange lighten-4
#fff59d yellow lighten-3	#ffe082 amber lighten-3	#ffcc80 orange lighten-3
#fff176 yellow lighten-2	#ffd54f amber lighten-2	#ffb74d orange lighten-2
#ffee58 yellow lighten-1	#ffca28 amber lighten-1	#ffa726 orange lighten-1
#ffeb3b yellow	#ffc107 amber	#ff9800 orange
#fdd835 yellow darken-1	#ffb300 amber darken-1	#fb8c00 orange darken-1
#fbc02d yellow darken-2	#ffa000 amber darken-2	#f57c00 orange darken-2
#f9a825 yellow darken-3	#ff8f00 amber darken-3	#ef6c00 orange darken-3
#f57f17 yellow darken-4	#ff6f00 amber darken-4	#e65100 orange darken-4
#ffff8d yellow accent-1	#ffe57f amber accent-1	#ffd180 orange accent-1
#ffff00 yellow accent-2	#ffd740 amber accent-2	#ffab40 orange accent-2
#ffea00 yellow accent-3	#ffc400 amber accent-3	#ff9100 orange accent-3
#ffd600 yellow accent-4	#ffab00 amber accent-4	#ff6d00 orange accent-4

#fbe9e7 deep-orange lighten-5	#efebe9 brown lighten-5	#fafafa grey lighten-5
#ffccbc deep-orange lighten-4	#d7ccc8 brown lighten-4	#f5f5f5 grey lighten-4
#ffab91 deep-orange lighten-3	#bcaaa4 brown lighten-3	#eeeeee grey lighten-3
#ff8a65 deep-orange lighten-2	#a1887f brown lighten-2	#e0e0e0 grey lighten-2
#ff7043 deep-orange lighten-1	#8d6e63 brown lighten-1	#bdbdbd grey lighten-1
#ff5722 deep-orange	#795548 brown	#9e9e9e grey
#f4511e deep-orange darken-1	#6d4c41 brown darken-1	#757575 grey darken-1
#e64a19 deep-orange darken-2	#5d4037 brown darken-2	#616161 grey darken-2
#d84315 deep-orange darken-3	#4e342e brown darken-3	#424242 grey darken-3
#bf360c deep-orange darken-4	#3e2723 brown darken-4	#212121 grey darken-4
#ff9e80 deep-orange accent-1		
#ff6e40 deep-orange accent-2		
#ff3d00 deep-orange accent-3		
#dd2c00 deep-orange accent-4		

#eceff1 blue-grey lighten-5

#cfd8dc blue-grey lighten-4

#b0bec5 blue-grey lighten-3

#90a4ae blue-grey lighten-2

#78909c blue-grey lighten-1

#607d8b blue-grey

#546e7a blue-grey darken-1

#455a64 blue-grey darken-2

#37474f blue-grey darken-3

#263238 blue-grey darken-4

#000000 black

#ffffff white

N/A transparent